An Experimental Evaluation of Anomaly Detection in Time Series

Aoqian Zhang
Beijing Institute of Technology
aoqian.zhang@bit.edu.cn

Shuqing Deng
Beijing Institute of Technology
shuqing.deng@bit.edu.cn

Dongping Cui
Beijing Institute of Technology
dongping.cui@bit.edu.cn

Ye Yuan
Beijing Institute of Technology
yuan-ye@bit.edu.cn

Guoren Wang
Beijing Institute of Technology
wanggr@bit.edu.cn

ABSTRACT

Anomaly detection in time series data has been studied for decades in both statistics and computer science. Various algorithms have been proposed for different scenarios, such as fraud detection, environmental monitoring, manufacturing, and healthcare. However, there is a lack of comparative evaluation of these state-of-the-art approaches, especially in the same test environment and with the same benchmark, making it difficult for users to select an appropriate method for real-world applications. In this paper, we present a taxonomy of anomaly detection methods based on the main features, i.e., data dimension, processing technique, and anomaly type and six inner classes. We perform systematic intra- and inter-class comparisons of seventeen state-of-the-art algorithms on real and synthetic datasets with a point metric commonly used in classification problems and a range metric specifically designed for subsequence anomalies in time series data. We analyze the properties of these algorithms and test them in terms of effectiveness, efficiency, and robustness to anomaly rates, data sizes, number of dimensions, anomaly patterns, and threshold settings. We also test their performance in different use cases. Finally, we provide a practical guide for detecting anomalies in time series and discussions.

PVLDB Reference Format:
Aoqian Zhang, Shuqing Deng, Dongping Cui, Ye Yuan, and Guoren Wang. An Experimental Evaluation of Anomaly Detection in Time Series. PVLDB, 17(3): 483-496, 2023.
doi:10.14778/3632093.3632110

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zaqthss/experiment-tsad.

1 INTRODUCTION

Time series is one of the most commonly used data types in recent decades. Time series analysis involves methods of data analysis to gain valuable insights. Anomaly detection aims to find rare observations that deviate significantly from the majority of data [23] and is the most important part of time series analysis (mining). It has been studied in various applications, such as fraud detection [9], environmental monitoring alerting [18] and industrial manufacturing [10], cyber attack identification [36], etc. Unfortunately, an anomaly is rather challenging to define, especially in the context of time series data [54]. For the following reasons, it is difficult to evaluate the performance of an anomaly detection method and to select the suitable one (with appropriate parameters) for handling anomalies in real-world scenarios.

The first reason is the complexity and diversity of time series data [29]. A single timeseries data may be univariate or multivariate, and the latter may have a number of dimensions. Moreover, the anomalies encountered may be of different types [22]. Several algorithms have been proposed to detect anomalous behavior using different techniques, but they often focus on a specific case. For instance, [25] is only able to find anomalous data points in univariate time series. [7] can handle univariate data streams, but can only detect anomalous patterns of a certain length. Therefore, after analyzing the state-of-the-art anomaly detection algorithms, we classify them into three facets representing the three main features [5], i.e., data dimension, processing technique, and anomaly type. We can further divide them into classes under a particular facet, as shown in Figure 1, to make more detailed comparisons.

The second reason is the lack of thorough testing with the same datasets on the same platform and with the same metrics. Different works provide comparisons with different scores, even with the same basic precision-recall-F1 metric [2]. For example, [52] calculates F1 score using average precision and average recall, whereas [1] considers F1 score after threshold adjustment. In addition, the baselines being compared may be written in a different language or use different data structures. Therefore, for a fair performance comparison, systematic experiments should be performed between methods of the same class in the same test environment, which is called *intra-class* comparison.

The third reason arises from the observation that some recent papers [1, 3, 52, 58] evaluate their methods under point metrics on datasets with subsequence anomaly. These are essentially point methods, as an anomaly score is assigned to each data point and those points whose score is above a threshold are reported as an anomaly. In addition, to interpret the detection on a subsequence, they modify the prediction results before evaluation using the point-adjust [62] method. However, a comprehensive experiment should be employed to analyze the effects of these measures. In terms of efficiency, especially in big data applications where time series data have high dimensions (more than 50) and large scales (more than

Proceedings of the VLDB Endowment, Vol. 17, No. 3 ISSN 2150-8097.
doi:10.14778/3632093.3632110

100k), the trade-off between effectiveness and efficiency should be considered. In particular, the possibility of implementing univariate models on each data dimension [28] and finding out to what extent online detectors approximate the performance of batch methods [12, 44] could receive special attention. Therefore, comparisons between classes in the same facet, which we call *inter-class* comparisons, are also useful. Finally, the essential features of anomaly detection algorithms, including robustness in terms of threshold setting, should be captured in terms of practical applications. Systematic experiments with different real cases and analysis of the behavior of these algorithms will be helpful for users.

1.1 Contributions

To the best of our knowledge, this is the first in-depth experimental study to provide both *intra-class* and *inter-class* evaluations. Our main contributions in this paper are summarized as follows.

(1) We propose a taxonomy of methods for detecting anomalies in time series based on their essential features in three facets and six inner classes in Section 2.
(2) We present brief descriptions of 17 state-of-the-art methods in Section 3. Furthermore, we re-implement 10 of them with JAVA, refactor all methods with the same code structure, and build a testing framework to avoid the potential impact.
(3) We provide systematic intra-class comparisons under different scenarios with real-world and synthetic datasets in Section 4.2, including (a) effectiveness with point- and range-based metrics, (b) anomaly rates, (c) data sizes (scalability), (d) dimensions, (e) anomaly patterns.
(4) We employ inter-class comparisons and draw interesting findings in Section 4.3, like (a) point anomaly methods can also perform well under subsequence anomaly cases, (b) using range metrics on subsequence data can lead to more reasonable and robust results, (c) point-adjust method tends to report 'false' higher accuracy results and may mislead analysis.
(5) We analyze the capability of methods under different application cases, consisting of (a) the false positives and negatives they generate, (b) whether they can detect anomalies as early as possible and provide recommendations in Section 5.

1.2 Related Work

Table 1: Comparison of TAD Experimental Surveys

Exp Survey	*Dimension* <Uni, Mul, Inter>	*Processing* <Batch, Online, Inter>	*Anomaly Type* <Point, Range, Inter>	*Threshold Robustness*	*Application Guideline*
DODDS [57]	<✗, ✓, ✗>	<✗, ✓, ✗>	<✓, ✗, ✗>	✗	✗
MD [12]	<✓, ✗, ✗>	<✗, ✓, ✗>	<✓, ✓, ✗>	✗	✓
TUD [8]	<✓, ✗, ✗>	<✓, ✗, ✗>	<✓, ✗, ✗>	✗	✓
TODS [31]	<✗, ✓, ✗>	<✓, ✗, ✗>	<✓, ✓, ✓>	✗	✗
Exathlon [29]	<✗, ✓, ✗>	<✓, ✗, ✗>	<✗, ✓, ✗>	✓	✓
TSB-UAD [47]	<✓, ✗, ✗>	<✓, ✗, ✗>	<✓, ✓, ✗>	✗	✗
Meta [44]	<✗, ✓, ✗>	<✓, ✓, ✓>	<✓, ✓, ✗>	✗	✓
HPI [49]	<✓, ✓, ✗>	<✓, ✗, ✗>	<✓, ✓, ✗>	✗	✓
Our work	<✓, ✓, ✓>	<✓, ✓, ✓>	<✓, ✓, ✓>	✓	✓

We summarize 8 previous experimental studies on time series anomaly detection and our work in Table 1. The first three columns represent the fundamental facets covered in the study, e.g., in the first column, Uni indicates that the work includes univariate detection methods, Mul stands for multivariate detection methods, while Inter represents the consideration of comparisons between univariate and multivariate methods. Similarly, <Batch, Online, Inter> refers to the processing technique, while <Point, Range, Inter> focuses on the type of target anomaly. Thresholds Robustness indicates whether the work examines the robustness of the compared methods to thresholding. Finally, Application Guideline indicates whether the work provides practical guidelines for method selection in real-world scenarios with different requirements.

In [57], distance-based online detection methods of point outliers in data streams are evaluated. However, only efficiency factors such as CPU time and leakage memory are tested under different parameters. MD [12] compares runtime and efficacy of different detection methods for four types of anomalies in real streaming cases. It serves as a guide for selecting the 'best' univariate technique in terms of real-time and accuracy, but lacks considerations for the characteristics of the datasets and the type of application. TUD [8] focuses on supervised methods for univariate data. TODS [31] revises outlier definitions and proposes a new taxonomy for anomaly types. It tests batch methods under different anomaly rates for different types of anomalies, but omits evaluations under other factors such as data dimensions. Exathlon [29] generates a novel benchmarking platform. Three deep learning methods are tested on the proposed datasets with range anomalies and evaluated by range metrics [53]. Its major difference to others is the capability of evaluating explanation discovery results, which is not covered by our work. Nevertheless, it fails to compare more types of anomaly detection methods and provide more comprehensive analysis. TSB-UAD [47] establishes a new benchmark to facilitate the evaluation of univariate methods and evaluates several batch methods to demonstrate the robustness of the proposed benchmark, but does not address multivariate time series cases. The aforementioned TODS, Exathlon, and TSB-UAD provide novel benchmarks based on different techniques that present the main contributions and differences from other studies. Meta [44] presents a qualitative review of online detectors from different families and proposes a fair evaluation environment for online and offline detectors over multivariate data. The goal is to find out to what extent online detectors approximate the performance of offline detectors. However, it focuses only on unsupervised methods for detecting point anomalies (including range anomalies with point metrics) and leaves out (semi)supervised methods.

HPI [49] makes an exhaustive evaluation of 71 anomaly detection methods and evaluates them on various datasets with different types of anomalies. Although it provides a comprehensive evaluation, it still has some untouched areas, which we extend as follows: (1) They optimize the parameters of all methods globally over the same well-labeled synthetic dataset, but we tune these parameters per dataset to get a fairer evaluation with their best performance; (2) They do not consider the translation of anomaly scores to anomaly labels via thresholding, however, we analyze the robustness of using different thresholding techniques since anomaly identification is crucial in most real-world scenarios; (3) They simply implement point and range metrics for point and subsequence anomalies, respectively, without deeper analysis. We also perform the inter-class comparison using a point metric for subsequence anomalies, which has been widely used in previous studies, and find insightful results; (4) Their research insights can help users

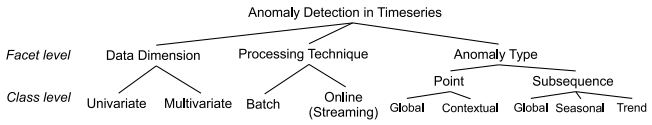

Figure 1: Facets and classes of anomaly detection algorithms

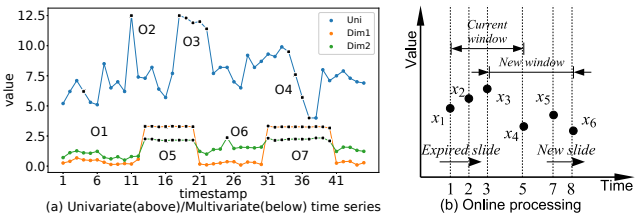

Figure 2: (a) time series and (b) online processing technique

select the optimal algorithm for specific anomaly types, such as extremum anomalies. We also analyze real-world applications that are more interested in positive, negative, and early detection cases, and provide a practical guide for method selection.

The above papers investigate problems of time series anomaly detection (TAD) from various aspects, but leave room for further analysis. Theoretical surveys [5, 11, 14, 22] provide a structured overview of research methods with different emphases. In contrast, we consider the state-of-the-art methods not covered in these studies, perform a comprehensive analysis on wider ranges as shown in Table 1, and obtain contributions as indicated in Section 1.1.

The original papers proposing anomaly detection algorithms that we compare in our work [1, 3, 6, 7, 17, 19, 24–26, 39, 42, 50, 52, 55, 58, 66, 67] also consist of empirical comparisons. However, these experiments are limited in terms of datasets, competitors, and thorough analyzes of precision/recall, efficiency, and sensitivity compared to our study. We use datasets from these works and benchmarks from Numenta [34], Exathlon [29], and TODS [31].

2 PRELIMINARIES

2.1 Problem Definition

Anomaly detection in time series data can be defined as follows.

Definition 2.1 (Time series). A time series is a series of data points indexed in time order. Consider a time series of n observations, $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. The i-th observation (data point) $\boldsymbol{x}_i \in \mathbb{R}^D$ consists of D dimensions $\{x_i^1, \ldots, x_i^D\}$ and is observed at timestamp t_i. A subsequence $\boldsymbol{X}_{i,\ell} = \{\boldsymbol{x}_i, \ldots, \boldsymbol{x}_{\ell-i+1}\}$ is a subset of consecutive time points from time series $\boldsymbol{X}$ starting at position i with length ℓ.

Definition 2.2 (Anomaly Detection in Time Series). An anomaly is an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data [4]. Anomaly detection in time series involves finding the abnormal data points $\boldsymbol{x}_i$ or subsequences $\boldsymbol{X}_{i,\ell}$ in the given time series.

2.2 Analysis Facets

The anomaly analysis problems can be categorized in various ways [22]. In this work, we propose a comprehensive taxonomy consisting of the following three facets, i.e., data dimension, processing technique, and anomaly type, as shown in Figure 1. For each facet, we further divide detection methods into classes. The details of these facets and classes are presented in this section.

2.2.1 Data dimension. The number of time-dependent variables that an anomaly detection method can consider simultaneously is its essential feature [5]. Following Definition 2.1, $\boldsymbol{X}$ is called as a **univariate** time series if $D = 1$ while $\boldsymbol{X}$ is a **multivariate** time series if $D > 1$.

Scenario 1: High dimension. Theoretically, multivariate methods can exploit correlations between dimensions, resulting in better accuracy than univariate methods, which can only treat each dimension separately. However, multivariate methods require much more time, as univariate methods can achieve high efficiency by parallelizing each dimension. The test on Swat (51 dimensions) shows that, the univariate method IDK costs 16.50s, while the multivariate method PBAD takes 125.17s, almost 10 times the time. On the other hand, [28] implements the LSTM model on each dimension of the satellite dataset, since LSTMs have difficulty predicting m-dimensional results accurately when m is large. Therefore, a systematic inter-class comparisons should be performed to test the possibility of using univariate methods on multivariate data.

2.2.2 Processing Techniques. A time series can also have an infinite number of data points (stream). Methods that can handle infinite time series are called **online** methods, while those that cannot are called **batch** methods. The sliding window is the technique most commonly used by online methods [57].

Scenario 2: Large Scale. Theoretically, batch methods have higher accuracy because they can compare all data simultaneously. In contrast, online methods can run faster because they process only the data points in the current window.On the one hand, online methods can continuously update their models for incremental anomaly detection to account for the fact that time series characteristics evolve over time [21, 44]. On the other hand, the efficiency gap, especially for large datasets, should also be taken into account[40]. Therefore, like [44], we are interested in the extent to which online methods approximate the performance of batch methods and what the trade-off between effectiveness and efficiency is.

2.2.3 Anomaly Types. In a time series, data points that deviate significantly from other observations are called **point** anomalies. Subsequences of the time series with less similarity compared to others are called as **subsequence** anomalies. The capability of a method must be evaluated by an appropriate metric related to the type of anomaly, otherwise misleading results may be obtained [45, 53]. To compare methods in different cases, we further divide point and subsequence anomalies into the following patterns [31].

Point Anomaly. Given a time series $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. Let $|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t| > \delta$, where δ is a threshold and $\hat{\boldsymbol{x}}_t$ is the expected value. *Global* anomaly indicates that $\delta \sim \sigma(\boldsymbol{X})$, where $\sigma(\boldsymbol{X})$ is the standard deviation of the whole time series. *Contextual* anomaly with $\delta \sim$

Table 2: Anomaly detection methods considered here

	Algorithm	Mul	Process	Anomaly	Threshold	Code	Speedup
Distance	NETS [66]	✓	online	point	θ_k	JAVA	-
	STARE [67]	✓	online	point	top-θ_K	JAVA	-
	NP [24]	-	batch	subsequence	top-θ_K	python	1.5
	MERLIN [42]	-	batch	subsequence	top-θ_K	matlab	60
Pattern	PBAD [19]	✓	batch	subsequence	top-θ_K	python	2.5
	LRRDS [26]	✓	batch	subsequence	cluster	r	1
	SAND [7]	-	online	subsequence	top-θ_K	C & python	0.3
	NormA [6]	-	batch	subsequence	top-θ_K	C & python	-
	GrammarViz [50]	-	batch	subsequence	top-θ_K	JAVA	-
	IDK [55]	-	batch	subsequence	top-θ_K	python	20
	SHESD [25]	-	batch	point	θ_k, top-θ_K	JAVA	
Deep Learning	BeatGAN [39]	✓	batch	subsequence	top-θ_K	python	-
	Omni [52]	✓	batch	point	θ_τ	python	-
	USAD [3]	✓	batch	point	θ_τ	python	-
	GDN [17]	✓	batch	point	θ_τ	python	-
	RCoder [1]	✓	batch	point	θ_τ	python	-
	TranAD [58]	✓	batch	point	θ_τ	python	-

$\sigma(X_{t-k,t+k})$ denotes that the such anomaly point is different from its neighbors in a specific range (window size k).

Subsequence Anomaly. Here, we assume that the subsequence $X_{i,j} = \rho(2\pi\omega T_{i,j}) + \tau(T_{i,j})$, where ρ, ω and τ represents the basic shape, trend and seasonality, respectively. *Global*(shapelet) anomaly refers to the subsequences with dissimilar basic shapes, which can be defined as $s(\rho, \hat{\rho}) > \delta$, where s is the similarity function [31]. *Seasonal* anomaly with $s(\omega, \hat{\omega}) > \delta$ means that those subsequences have unusual seasonality compared to the whole series. *Trend* anomaly indicates the subsequences that significantly alter the trend of the time series, leading to a permanent shift in the mean of the data, which can be denoted as $s(\tau, \hat{\tau}) > \delta$.

Scenario 3: Parameter relaxation. Nevertheless, it is difficult to find a specific pattern of an anomaly in real datasets. Current subsequence methods require the *length* of the target subsequence as input [5–7]. However, it is rather difficult to determine such a parameter without sufficient prior knowledge. In contrast, the point methods do not require such additional input (the length is 1). Hence, we wonder (1) how powerful point methods are in the presence of anomalies in subsequences; (2) how well the methods can detect different patterns of anomalies.

Example 2.3. Point and subsequence anomalies can be univariate or multivariate. Figure 2(a) shows two univariate point anomalies, $O1$ (*global*) and $O5$ (*contextual*). In terms of patterns, $O3$ is a *trend* anomaly as it leads to a permanent shift with a lower mean, $O2$ is a *seasonal* anomaly and $O4, O6$ are *global* anomalies. Data points in univariate time series $X = \{x_1, \ldots, x_6\}$ are observed at time 1, 2, 3, 5, 7, 8 in Figure 2(b).Let us set the window size $\theta_W = 4$, slide size $\theta_S = 2$. The online method will first process the data points $\{x_1, \ldots, x_4\}$ in the current window. After the processing, $\{x_1, x_2\}$ in the first slide will be expired and $\{x_5, x_6\}$ in the new slide will come in, forming a new window containing $\{x_3, \ldots, x_6\}$.

3 ALGORITHMS

Following the proposed taxonomy, we choose the latest representatives for each class: 11/17 of these methods are published since 2020 and 12/17 are not compared in prior works in Section 1.2. We will briefly introduce them in the class of detection techniques, i.e., distance, pattern, and deep learning-based and extract their critical factors. Table 2 lists the properties of each algorithm. The 'threshold' column indicates how the threshold for anomalies is set.

3.1 Distance-based Methods

Distance-based algorithms [57] in the literature can detect both point and subsequence anomalies, but for simplicity we only refer to point anomalies in the following definitions.

Definition 3.1 (Neighbor). Given a distance threshold θ_R, a data point x_i is a neighbor of data point $x_j (x_i \neq x_j)$ if the distance $d(x_i, x_j) \leq \theta_R$.

Definition 3.2 (Distance-based Outlier). Given a time series X, a count threshold θ_k and a distance threshold θ_R, distance-based outliers are set of data points that have less than θ_k neighbors.

The key factors are the distance threshold θ_R and the count threshold θ_k, determining the neighbors for each data point and the anomalies. For online methods, the factors window size θ_W and the slide size θ_S affect the performance and are usually set depending on the size of the dataset. Distance methods for point anomalies often focus on the scalability problem [57] and utilize different structures to save running times. NETS [66] employs a set-based approach following net effect, that data points in a short period of time are likely to be concentrated in a set of small regions in the data space. STARE [67] observes that data distributions in many regions of the data space change little across window slides and thus skips updating densities in local regions that do not change significantly. NP [24] uses bagging to robustly discover frequent and rare subsequences. A nearest neighbor ball technique that replaces the nearest neighbor distance which is too small with the radius of such a ball is proposed to provide a robust estimation and solve the twin freak problem [24], which is not solved in *matrix profile* [65]. DRAG [64] creates a candidate discord set C and then searches it for a list of discords whose nearest neighbor distance is greater than a hyper-parameter r. MERLIN [42] provides a structured search to set such r. Since NETS and MERLIN dominates DRAG, respectively, in almost all the datasets and hence we only report the former two here.

3.2 Pattern-based Methods

The so-called pattern represents the regularities in the data, such as an ordered set of data points that occur frequently in the data, or a particular distribution. Pattern-based methods [14] are usually proposed to find subsequence anomalies, which we use in the following definitions. Point anomalies can be found if the pattern indicates a distribution.

Definition 3.3 (Pattern-based Outlier). Given a time series X, an anomaly length ℓ and a count threshold θ_K, pattern-based outliers are the set of top-θ_K subsequences with length ℓ and the lowest similarity to the patterns extracted from X.

The factors for pattern methods vary depending on how they define patterns, support, and threshold for anomalies. In general, the key factors are the support threshold θ_{sup}, the anomaly threshold θ_τ, or the count threshold θ_K, which represent the minimum occurrence of a pattern, the border of an anomaly, and the rank of scores that an anomaly achieves, respectively.

ESD [48] computes the Extreme Studentized Deviate test statistic to find anomaly points. SHESD [25] extends ESD by using

STL decomposition [13] and replacing the mean and standard deviation with more robust median and median absolute deviation. Considering the seasonality, SHESD outperforms ESD and hence we only report SHESD. NormA [6], SAND [7] and IDK [55] learn a normal model (clustering the distribution of normal patterns) and the similarity distance to the normal model is used as the anomaly scores for target subsequences. SAND further extends the k-Shape clustering [46] so that the normal model can be updated from one batch to the next, making itself an online method. IDK uses Isolation Distributional Kernel [56] to compute the similarity. PBAD [19] and GrammarViz [50] use rules to extract the normal patterns. The former presents frequent pattern mining research [68] to find frequent patterns (whose support is greater than θ_{sup}) while the latter implements two grammar induction methods [33, 43] to generate a context-free grammar. LRRDS [26] uses recurrence plot [69] and extracts the subsequence set from the local recurrence rates (LREC) [63]. The anomaly score is computed by comparing similarities between statistics in the LREC curve.

3.3 Deep Learning-based Methods

Recently, several deep learning-based methods have been proposed to detect anomalies in time series [15]. Various architectures have been developed to capture latent information from temporal and dimensional aspects.

For reconstruction based methods, after learning a model of the normal data in the latent space, the test time series is first transformed to the latent space and then reconstructed to the data space. Finally, the reconstructed values that have a larger distance from observations will be identified as anomalies. OmniAnomaly(Omni) [52] proposes a stochastic recurrent neural network by using GRU to capture complex temporal dependencies between multivariate observations in data space, and a variational autoencoder (VAE) to map observations to stochastic variables. USAD [3] presents an encoder-decoder architecture within a two-phase adversarial training system (GAN). BeatGAN [39] also provides an interpretable method that combines autoencoders and GAN to detect anomalous subsequences. RCoder [1] introduces an encoder-decoder framework to learn the bounds of reconstructed signals. The key difference is that its size of latent space is 1 and a spectral analysis with Fast Fourier Transform is then applied. TranAD [58] combines transformer-based encoder-decoder networks with adversarial training. Prediction base methods learn a model from normal data and then predicts the target value based on the model and observations before the target time. Finally, the difference between the predicted value and observation is used as the anomaly score. GDN [17] employs the Graph Neural Network to learn the dependency relationship between different dimensions and use graph attention mechanism to make predictions.

3.4 Implementation Notes

Some widely used systems, such as Apache IoTDB [59], can only support JAVA as built-in functions, and hence we rewrite all non-deep learning methods in JAVA (except NormA, the existing toolkit runs differently). We also refactor NETS, STARE, GrammarViz, and SHESD (in JAVA) and deep learning methods (in Python) with our data structures to avoid potential impact on efficiency. We remove

Table 3: Real-world and synthetic datasets summary

	Real-world	Size	#Dim	Rate%	Pattern	Avg Length
Point	Yahoo [35]	1.5k	1	0.7	Contextual	-
	Twitter [25]	14k	1	0.7	Global	-
	Stock [57]	10k	1	5 - 25	Mixed	-
	Tao [57]	568k	3	5 - 25	Mixed	-
	SMTP [38]	95k	3	0.03	Mixed	-
	DLR [67]	23k	9	2.2	Contextual	-
	ECG [67]	112k	32	16.3	Global	-
Subsequence	Power [30]	35k	1	8.6	Seasonal	750
	Sed [7]	100k	1	3.0	Global	64
	Taxi [53]	10k	1	10	Global+Seasonal	207
	Machine [53]	22k	1	10	Global+Seasonal	567
	Exercise [19]	10k	3	15.1	Mixed	140.2
	Exathlon [29]	3k	19	17.4	Mixed	64.1
	Swat [41]	90k	51	12	Mixed	317.2
	Smd [52]	28k	38	9.5	Mixed	336.8
	Synthetic	Size	#Dim	Rate%	Pattern	Avg Length
Point	Uni-point-g	1k-100k	1	5 - 25	Global	-
	Uni-point-c	1k	1	5 - 25	Contextual	-
	Mul-point	1k-100k	32	5 - 25	Global	-
Subsequence	Uni-sub-g	1k-100k	1	5 - 25	Global	50
	Uni-sub-s	1k-100k	1	5 - 25	Seasonal	50
	Uni-sub-t	1k-100k	1	5 - 25	Trend	50
	Mul-sub-g	1k-100k	3/50	5 - 25	Global	50
	Mul-sub-s	1k-100k	3/50	5 - 25	Seasonal	50
	Mul-cor-g	5k	3	10	Global	50
	Mul-ncor-g	5k	3	10	Global	50

POT [51] (selecting threshold automatically) and point-adjust [62] (modifying predictions before evaluation) to make a fair comparison. Analysis on these changes can be found in Section 4.3.4 and 4.3.5, respectively. The 'Code' column in Table 2 shows the original language in which each method was written, while the 'Speedup' column shows the increase in efficiency after our implementation (except SAND, the decomposition tools runs far more slowly). We verify that the reported anomalies of the methods before and after our refactor are identical and guarantee reproducibility.

4 EXPERIMENT

This section will first show the experimental settings. Then, insights found over intra-class and inter-class comparisons are explained. Similar to [49], we try our best to describe findings from experimental results. However, if a method performs poorly in our evaluation, it does not necessarily mean that its theory is bad, because the metric and the situation are quite different.

4.1 Settings

We run experiments on a Windows 10 server with a 3.79GHz 12 Core CPU and 128GB RAM. Deep-learning methods are employed with GPU without comparing the efficiency. Others are tested under a single-core environment to compare running times.

4.1.1 Datasets. We employ widely used *real-world* datasets with labels as benchmarks, 7 with point and 8 with subsequence anomalies with various sizes, dimensions, anomaly rates (lengths) and patterns. We also generate *synthetic* datasets whose base type is sine, with different anomaly patterns (See Section 2.2.3) according to the guideline in [31] for the case of the unreliability of anomaly labels [61]. For point anomaly, we inject *global*(g) and *contextual*(c) outliers. For subsequence anomaly, we add *global*(g), *seasonal*(s) and *trend*(t) outliers, respectively. The average length is set to 50 by default, so any algorithm is applicable. In order to avoid the effect of randomness, we run experiments on synthetic datasets 10 times

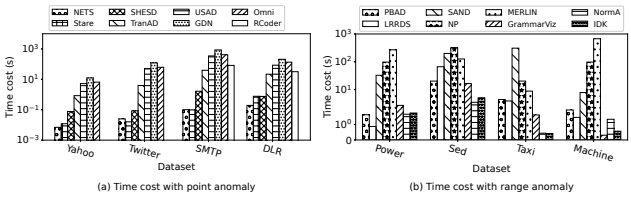

Figure 3: Time cost over various datasets

with different generating seeds and report the average results. The summary of all datasets can be found in Table 3.

4.1.2 *Evaluation Metrics.*

Accuracy. Different metrics are used with respect to anomaly types. $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$ [2] are utilized as **point metric**s, where TP, FP, FN are the number of true positives, false positives, and false negatives, respectively. We choose a recent metric proposed for subsequence anomaly which focuses on the overlap of predicted anomaly and true anomaly [53] as **range (or subsequence) metric**. Specifically, given a set of real anomaly ranges $R = \{R_1, \ldots, R_{N_r}\}$ and a set of predicted anomaly ranges $P = \{P_1, \ldots, P_{N_p}\}$, we have $Recall_T(R, P) = \frac{\sum_{i=1}^{N_r} Recall_T(R_i, P)}{N_r}$ and $Precision_T(R, P) = \frac{\sum_{i=1}^{N_p} Precision_T(R, P_i)}{N_p}$. In general, the default setting *Flat bias* [53] is employed. $F-measure = \frac{2*Precision*Recall}{Precision+Recall}$ applies for both metrics.

Efficiency. Apart from loading the data from the file and result evaluation, the total time of all the other procedures will be added and reported as the time cost of the target method.

4.1.3 *Parameter Search.*

In order to compare methods fairly and precisely (under best configurations), though all of them are unsupervised methods, we still employ a systematic (hyper)-parameter search process. Specifically, we design a search space for each parameter (including threshold) according to the suggestions in their paper, e.g., for USAD we tune the latent size with {2, 5, 10, 20} [3] or based on the distributions of the target dataset. We then split the dataset into training, validation and test set with a '4:1:5' ratio. It's worth noting that the training sets are anomaly-free, as required by [3, 17, 52, 58]. The parameter with the best performance in validation set will be settled for testing.

4.2 Intra-class Comparisons

As mentioned earlier, articles presenting new methods often include such comparisons, but the variety of cases with respect to factors and datasets is often limited. For simplicity, we use "**Point**" to denote the results of point methods and "**Sub**" of subsequence methods. "**Summary**" stands for the take-away conclusions.

4.2.1 *Varying Datasets.*

We first evaluate the performance of all methods on different datasets with their best parameters. To obtain the results of the univariate methods on multivariate data, we run them separately on each dimension (similar to [52]), and combine all the reported anomalies. Results are shown in Table 4.

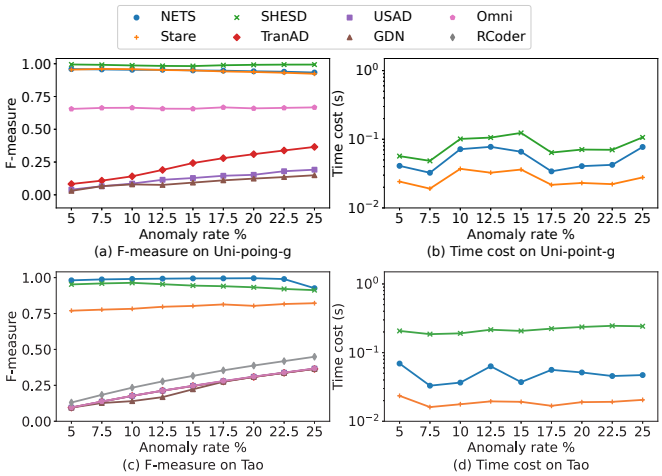

Figure 4: Varying rates on (a-b) Uni-point-g (c-d) Tao

Point. We can see that NETS always have high accuracy across all datasets. Stare relies heavily on the parameter indicating the number of anomaly points in each window. Complex distributions of real anomalies results in its relatively lower accuracy. As for deep learning methods, all except RCoder are close to NETS on *Yahoo* and perform worse on others. RCoder, which cannot be used for univariate data due to its mechanism, performs best on *ECG* with 32 dimensions but poorly on others whose dimensions are smaller than nine. Overall, it is surprising that deep learning methods do not behave well. The possible reason is that the number of data instances and dimensions are not sufficient to learn a good model. Figure 3(a) shows the efficiency of each method. We only show the first 4 datasets, since others are similar. NETS computes on a selected sub-dimension, making it the most efficient method. TranAD shows the best time efficiency due to its architecture.

Sub. Multivariate methods PBAD and BeatGAN achieve higher accuracy than univariate methods on multivariate data. In particular, univariate methods have high recall but very low precision. LRRDS performs better than PBAD and BeatGAN only on *Sed*. This is because the mean of abnormal patterns in *Sed* is obviously lower than that of normal patterns, so LRRDS is easier to identify with embedding techniques than simple mean and length features. As for univariate methods, NormA can perform quite well on all univariate data. MERLIN performs exceptionally well on *Power*, showing the capability over seasonality. SAND does not perform well on long anomalies (e.g., the length of the anomaly in *Power* is 750). It has to decompose a large matrix many times if the anomaly has a large length. We note that it takes much more time when the length of the anomaly is greater than 100. Both GrammarViz and PBAD use a pattern mining approach to identify sequence patterns, so they are comparable on univariate datasets.

Summary. (1) There is no super-algorithm that is suitable for all cases. (2) Given proper parameters, NETS can perform best in most cases. (3) PBAD and BeatGAN have better overall accuracy but cost more time. (4) Deep learning methods do not outperform others on datasets with complex anomaly patterns.

Table 4: Accuracy over various datasets

Dataset	NETS			Stare			SHESD			TranAD			USAD			GDN			Omni			RCoder		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Yahoo	0.727	1	**0.842**	0.429	0.375	0.400	1	0.625	0.769	0.368	0.875	0.519	0.368	0.875	0.518	0.368	0.875	0.519	0.471	1	0.64	-	-	-
Twitter	0.739	0.878	**0.802**	0.203	0.959	0.335	0.260	0.176	0.210	0.737	0.189	0.301	0.205	0.412	0.274	0.484	0.419	0.449	0.079	0.878	0.145	-	-	-
SMTP	0.400	0.375	**0.387**	0.294	0.313	0.303	0.001	1	0.003	0.001	0.188	0.003	0.001	0.222	0.002	0.004	0.022	0.007	0.250	0.375	0.300	0.001	0.312	0.001
DLR	0.468	0.424	**0.445**	0.109	0.517	0.179	0.061	1	0.115	0.115	0.224	0.152	0.115	0.224	0.154	0.118	0.852	0.060	0.180	0.180	0.180	0.030	0.252	0.053
ECG	0.484	0.441	0.462	0.239	0.267	0.252	0.373	0.430	0.399	0.160	0.557	0.248	0.280	0.293	0.286	0.232	0.328	0.272	0.347	0.165	0.224	0.640	0.426	**0.512**

Dataset	PBAD			LRRDS			BeatGAN			SAND			NP			MERLIN			GrammarViz			NormA			IDK		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Power	0.262	0.758	0.389	0.415	0.248	0.310	0.498	0.340	0.404	0.159	0.156	0.157	0.575	0.575	0.575	0.936	0.998	**0.966**	0.453	0.222	0.298	0.868	0.810	0.838	0.343	0.244	0.285
Sed	0.310	0.800	0.446	0.597	0.972	0.739	0.690	0.723	0.706	0.657	0.769	0.708	0.695	0.814	0.750	0.505	0.829	0.627	0.435	0.533	0.479	0.435	0.669	0.527	0.938	0.938	**0.938**
Taxi	0.143	0.664	0.235	0.164	0.463	0.242	0.354	0.437	0.391	0.250	0.242	0.246	0.468	0.498	0.482	0.460	0.460	0.460	0.333	0.135	0.192	0.441	0.533	**0.482**	0.498	0.373	0.427
Machine	0.227	0.500	0.312	0.073	0.500	0.127	0.521	0.471	**0.495**	0.100	0.106	0.103	0.009	0.018	0.012	0.143	0.429	0.215	0.197	1	0.33	0.171	0.332	0.226	0.429	0.454	0.441
Exercise	0.495	1	0.662	0.283	0.656	0.396	0.756	0.760	**0.758**	0.293	0.672	0.408	0.309	0.983	0.470	0.377	0.971	0.543	0.553	0.745	0.635	0.292	0.743	0.420	0.171	0.405	0.245
Exathlon	0.515	0.856	**0.643**	0.197	0.341	0.250	0.299	0.694	0.418	0.206	1	0.341	0.149	0.993	0.259	0.200	1	0.334	0.147	1	0.256	0.095	0.849	0.170	0.169	1	0.289
Swat	0.185	0.566	**0.279**	0.286	0.124	0.173	0.340	0.233	0.276	0.082	0.993	0.151	0.055	1	0.104	-	-	-	0.137	1	0.242	0.063	0.529	0.113	0.041	1	0.079
Smd	0.294	0.303	**0.298**	0.676	0.098	0.171	0.273	0.235	0.253	0.070	0.925	0.129	0.082	0.980	0.151	0.063	0.732	0.116	0.018	1	0.035	0.088	0.742	0.157	0.085	0.799	0.154

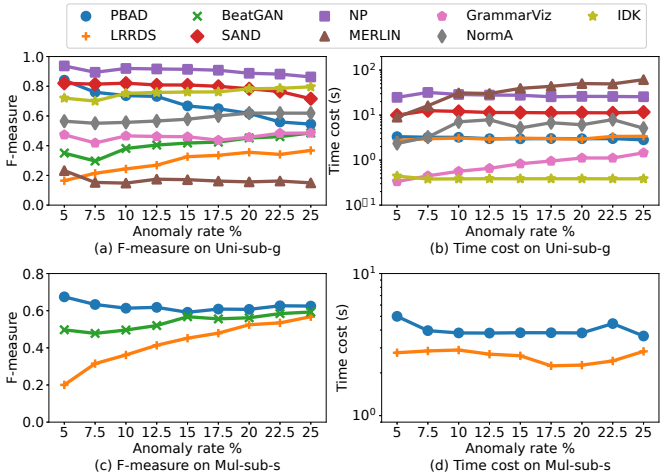

Figure 5: Varying rates on (a-b) Uni-sub-g (c-d) Mul-sub-s

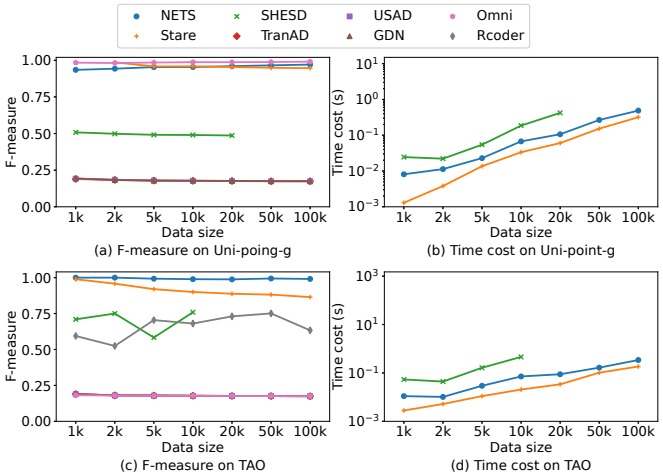

Figure 6: Varying sizes on (a-b) Uni-point-g (c-d) Tao

4.2.2 Varying Anomaly Rate a%. [61] claims that current datasets contain too few anomalies for even a simple method to find them. However, the majority of the data should be normal. Therefore, we vary anomaly rates from 5% to 25% to test the sensitivity of the methods on synthetic datasets. The data size is 10k for point and 5k for subsequence anomalies.

Point. As illustrated in Figures 4(a, c), all non-deep learning methods are relatively stable. NETS perform a bit worse on *Tao* when the anomaly rate achieves 25%. Since it is difficult to set an appropriate θ_k and θ_R to identify the anomalous points because they also have a similar number of neighbors as normal points. On the contrary, most deep learning methods achieve a better result with increasing anomaly rate. This may be because TranAD, USAD, and GDN fail to learn a good model (even with the best parameter via grid search) and tend to predict more points as anomalies. Therefore, more false positives become true positives when the anomaly rate increases. Omini and RCoder manage to learn a better model and perform well on *Uni-point-g* and *Tao*, respectively.

Sub. Figures 5(b, d) show that, the time costs of all methods remain constant for both univariate and multivariate data, except for MERLIN and GrammarViz, which is due to the fact that the time complexity of MERLIN and GrammarViz is linearly related to the number of anomalous subsequences. When we compare the results of the different methods, we find that most of them, such as PBAD, remain stable or perform better when the anomaly rate is lower, but gradually become worse as the anomaly rate increases. In contrast, LRRDS and BeatGAN performs even better at a higher anomaly rate, which is due to the reduction of false positives caused by rough recognition of anomaly point.

Summary. (1) Methods are stable when anomaly is rare (< 25%), which is different to the statements about random detection in [44]. (2) A method may perform better as anomaly rate increases since the large number of the reported false positives can become true positives. (3) Anomaly rate shows little effect on the efficiency.

4.2.3 Varying Data Size n. To check the sensitivity and scalability against the data size, we run experiments by varying data sizes from 1k to 100k on real and synthetic datasets with 10% anomalies.

Point. The results in Figures 6(b, d) show that all methods cost more time as the data size increases. In terms of accuracy, they show consistent results against the data size. It is noted that SHESD makes exceptions in some cases. Stare and NETS are the two fastest

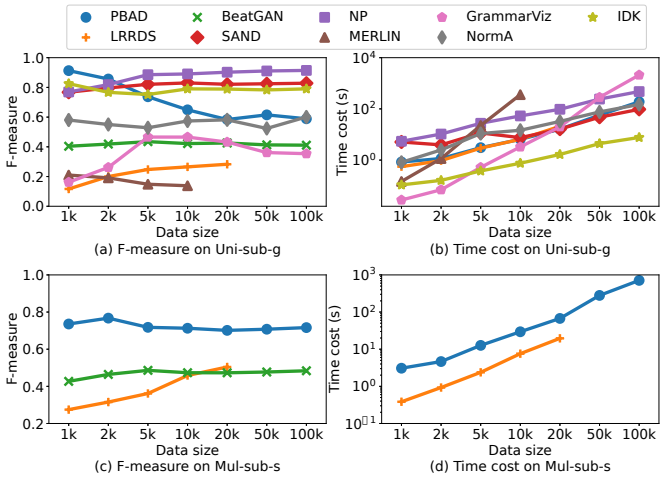

Figure 7: Varying sizes on (a-b) Uni-sub-g (c-d) Mul-sub-s

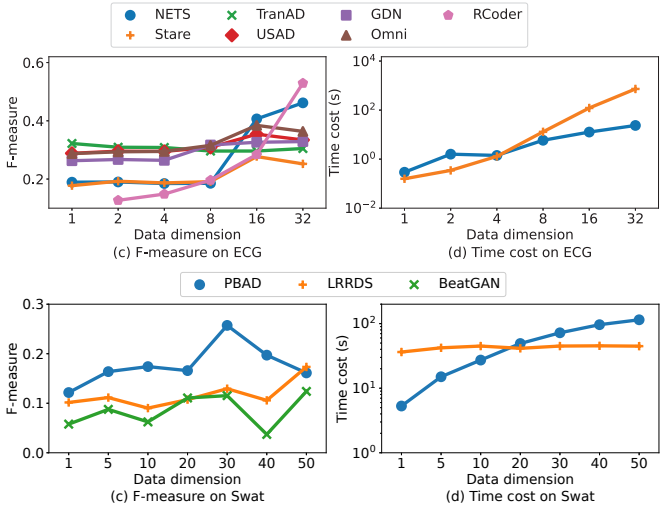

Figure 8: Varying dimensions on (a-b) ECG (c-d) Swat

methods. The latter two algorithms take advantage of the "set effect" that skips additional updates when new data arrives.

Sub. As can be seen in Figure 7, most pattern-based methods show unstable results for small (e.g., 5k) because it is difficult to extract patterns. When the data is large enough, the results of the subsequence methods are usually quite consistent. LRRDS reflects the same trend in Section 4.2.2. We do not report the results of LRRDS after 20k because it is out of memory and do not report the results of MERLIN after 10k because it takes more than a day to run. Both of them show poor scalability.

Summary. (1) Stare, NETS, and NormA have good efficiency (also referred to the results in Figure 3). (2) Small data size (< 10k) may lead to unstable results for subsequence methods. (3) We recommend IDK when the data size is more than 50k because it provides good scalability and stable performance.

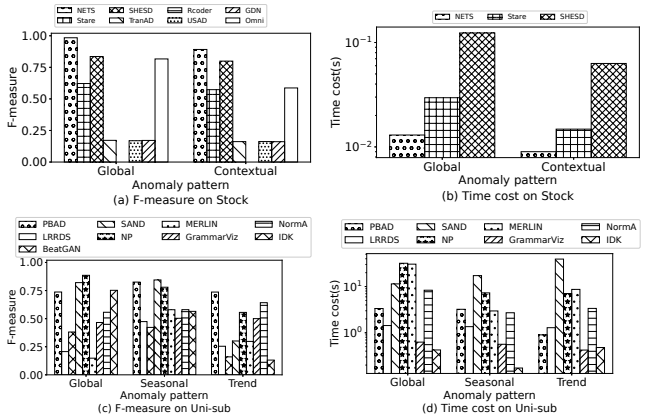

Figure 9: Varying patterns on (a-b) point (c-d) subsequence

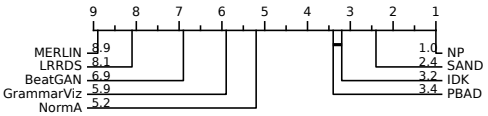

Figure 10: Critical difference diagram on subg

4.2.4 Varying Data Dimension |D|. Here we perform experiments on real datasets by varying the dimensions of the data.

Point. Figures 8(a-b) show that all methods have better results with larger dimensions. Besides, we need to adjust the parameter θ_R with the change in dimension to get a reasonable result. We also retrain the models of the deep learning methods, which results in all but RCoder remaining stable as the data dimension increases. RCoder has more reference data and estimators with more data dimensions, which significantly increases its accuracy. In terms of efficiency, all distance-based methods cost more time as the dimension increases, since it is more expensive to compute the distance.

Sub. The result of the subsequence anomaly can be checked in Figure 8(c-d). LRRDS applies dimensional compression and exhibits great scalability. In contrast, PBAD must extract the features in each dimension, which results in a significantly higher time overhead as the dimension increases. Since the anomaly is localized differently in each dimension of Swat, it is not surprising that performance varies widely across all methods, including BeatGAN.

Summary. (1)LRRDS scales well with data dimension, while NETS, Stare, and PBAD are significantly affected by it. (2) The sparsity problem can be handled by a good parameter setting. (3) Since the number of estimators increases with the data dimension, RCoder achieves better results with higher dimensions. (4) NETS is recommended when data dimension is limited (< 30). (5) RCoder is a good choice when data dimension is large (> 30).

4.2.5 Varying Anomaly Patterns. Finally, we test the selected algorithms for different patterns of anomalies.

Point. As shown in Figures 9(a-b), all methods achieve better accuracy for *global* anomaly (0.618) than *contextual* anomaly (0.551),

which is consistent with the common sense. They have similar efficiency for these two anomaly patterns.

Sub. The result for the subsequence case is shown in Figures 9(c-d). The average f-measures are 0.550, 0.619, and 0.398 for *global*, *seasonal*, and *trend* anomaly, respectively. Consistent with previous observations, PBAD and NormA has stable performance across all patterns. SAND and NP perform particularly well on *global* and *seasonal* outliers. Detecting *trend* outliers is challenging for BeatGAN and IDK, reflecting the need for improvement of non-stationary time series. To find out whether there are methods that fit all patterns, we apply the Friedman test [20] and a post-hoc Wilcoxon test [60] (with $\alpha = 0.05$) to the f-measures for different anomaly patterns. First, the Friedman tests yield a p-value greater than 0.05 and thus do not indicate that these methods are significantly different. Figure 10 shows the critical difference diagram[16] for the global anomaly as an example. Methods that are not connected by a bold line differ significantly in their average ranks. This proves that for a single anomaly pattern there is a particular method that clearly outperforms the others. For example, NP not only achieves first place for *global* outliers, but also significantly outperforms others.

Summary. (1) In point methods, *global* anomalies are easier to detect than *contextual* ones. (2) In subsequence methods, *seasonal* anomalies are easiest to find, while *trend* anomalies are hardest to find. (3) No one method fits all patterns. But certain methods are clearly better than others for a particular anomaly pattern, e.g., NP performs best for the *global* anomaly. (4) SAND and IDK are not suitable for non-stationary time series (with *trend* anomaly). (5) Distance-based methods are well suited for *global* and *seasonal* anomalies. (6) IDK can also work well for *global* anomaly, which is different from RI(8) in [49].

4.3 Inter-class Comparisons

In this section, we will test the performance of methods between different classes under the same facet. In the data dimension facet, we run univariate methods separately for each dimension of the multivariate data and then combine the results. In the processing technique facet, we run online methods across different window and slide sizes on datasets larger than 100k and with dimensions larger than 30. In the anomaly type facet, as mentioned in Section 1, we run both point and subsequence methods on datasets with subsequence anomalies under different evaluation metrics to analyze the impact of the metrics and the adjustment in predictions. In addition, we also test the impact of thresholds and performance under different application aspects.

Table 5: Performance on single dimension and combination

Dataset	PBAD			BeatGAN			SAND			NP		
	P	R	F	P	R	F	P	R	F	P	R	F
Exercise	0.495	1	**0.662**	0.756	0.760	**0.758**						
Combination	0.220	1	0.360	0.332	0.997	0.498	0.293	0.672	0.408	0.309	0.983	0.47
E_A1	0.357	1	0.526	0.708	0.72	0.714	0.483	0.513	0.498	0.894	0.943	0.918
E_A2	0.389	1	0.560	0.505	0.498	0.501	0.27	0.275	0.272	0.222	0.224	0.223
E_A3	0.352	0.719	0.473	0.249	0.248	0.248	0.192	0.204	0.197	0.196	0.202	0.199
Mul_ncor_g	0.684	1	**0.813**	0.514	0.753	**0.611**						
Combination	0.097	1	0.178	0.215	0.861	0.344	0.250	0.897	0.391	0.279	0.941	0.431
M_A1	0.092	0.300	0.140	0.219	0.246	0.232	0.315	0.329	0.322	0.377	0.463	0.416
M_A2	0.083	0.390	0.137	0.222	0.355	0.273	0.261	0.353	0.300	0.245	0.368	0.294
M_A3	0.107	0.402	0.168	0.223	0.358	0.275	0.311	0.42	0.357	0.217	0.325	0.260

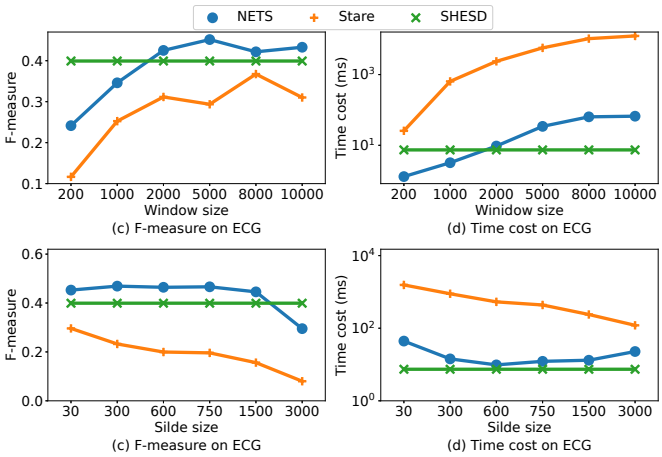

Figure 11: Varying (a-b) window sizes (c-d) slide sizes on ECG

4.3.1 Univariate Methods in Multivariate Datasets. As reported in Section 4.2.1, the point method SHESD lags far behind multivariate ones, therefore, we focus on the subsequence case.

Settings. If the detected anomalies in different dimensions overlap after combination, we merge them into a large range(subsequence) as the final result. Following the same logic, we also run multivariate methods (PBAD and BeatGAN) for each dimension separately and combine the detected anomalies for direct comparison, the results of which are displayed in the 'Combination' column in Table 5. We note that anomalies in *Exercise* (similar results are observed in other real datasets) always occur in all dimensions at the same position. Therefore, we test the effects of such 'overlap' on another synthetic data. The 'ncor' indicates that positions of the generated anomalies are different in each dimension. 'A1-A3' denote the three dimensions.

Results. Compared with the results on each dimension and after combination, most of the methods have a promotion on recall but a sharp reduction on precision, since the aggregation in each dimension will involve in more predicted anomalies (cover more real anomalies but also much more false positives). Compared with combined results on the synthetic data, we find out that anomaly positions (co-occur or not) do not clearly impact the effectiveness and efficiency. Univariate methods can achieve good results on some specific dimensions but acts poorly overall, while multivariate ones perform the opposite, showing the advantage of considering relationships over dimensions.

Summary. Promotion on efficiency cannot compensate for the sharp drop in accuracy (we omit efficiency results and show one in motivating scenario 1). Multivariate methods are better, unless we have sufficient prior knowledge about specific dimensions.

4.3.2 Online Methods in Batched Time Series. As explained in Section 4.2.1, the online method SAND suffers from the iterations of eigenvalue decompositions and its efficiency is significantly affected. Thus, we focus on the point case.

Settings. We run two experiments in *ECG* varying window sizes θ_W and slide sizes θ_S. Similar results are observed in other datasets. $\theta_S = 150$ in the first case and $\theta_W = 3k$ in the second. It should be

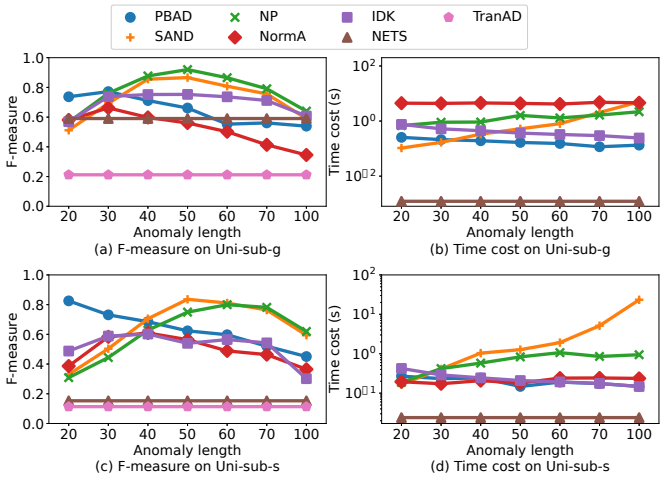

Figure 12: Varying lengths on (a-b) Uni-sub-g (c-d) Uni-sub-s

noted that $\theta_S \leq \theta_W$ always holds. The batch method SHESD is used as a baseline.

Results. Figure 11(a) shows that the accuracy of NETS first increases and then decreases with increasing window size. The reason is that if the window size is too small, it is difficult to find enough neighbors even for a normal data point, leading to false positives. If the window size is too large, the number of neighbors for a local anomaly point may exceed the predefined threshold θ_k, leading to false negatives. As can be seen in Figure 11(b), as expected, a larger window size causes the methods to consume more data points, which costs more time. One possible explanation for Figure 11(c) is that a larger slide size increases the variation in the number of anomaly points, which decreases the accuracy of Stare. On the other hand, if the slide size is too large, the intermediate information stored for NETS changes too much, which also negatively affects the accuracy. Regarding the efficiency shown in Figure 11(d), Stare take less time as the slide size increases. NETS cannot exploit the 'net effect' when the slide size is equal to the window size. Therefore, it can take the least time for a medium θ_S. It is noted that even the fastest online method NETS will be ten times slower than the simple batch method SHESD when $\theta_W = 10k$.

Summary. (1) Larger sizes do not necessarily produce better results, but cost more time. (2) Online methods can outperform batch methods under a proper setting, due to the evolvement characteristic of time series.

4.3.3 Point Methods in Subsequence Anomalies. Subsequence methods always consider the anomaly length ℓ as an input parameter [7]. However, this affects accuracy to some extent, since subsequence anomalies in real time series data rarely have a fixed length. In contrast, point anomaly methods require no such input (the length is 1). Therefore, we would like to know the effect of anomaly length and whether we can perform point methods for subsequence anomalies.

Settings. We use univariate data with injected *global* and *seasonal* anomalies to avoid the effect of dimensions. We omit *trend* because they have similar results to *seasonal*. For brevity, we report only NETS and TranAD, as other point methods perform similarly.

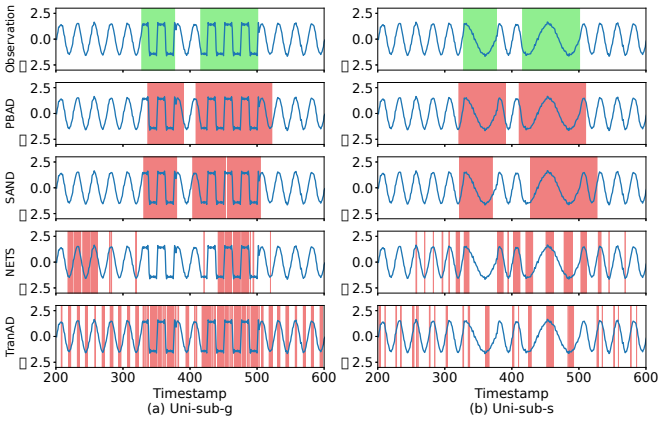

Figure 13: Case study on (a) Uni-sub-g (b) Uni-sub-s

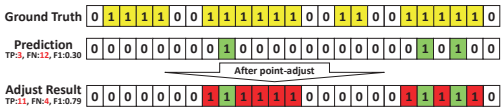

Figure 14: Point-adjust method

Results. Figure 12 shows the sensitivity of the methods to the anomaly length ℓ. PBAD does not identify the subsequence whose length is exactly ℓ, making it different from others. Other methods claim to achieve the best accuracy when ℓ is similar to the actual length ($\ell = 50$) of the anomalies, which can be seen in Figures 12(a, c). IDK demonstrates robustness for the input length, while the best input length at NormA is difficult to predict. It is interesting that NETS under *global* patterns (in Figure 12(a)) can also give good results with much less time (in Figure 12(b)). In Figure 13(a), we zoom in on 400 data points in the *Uni-sub-g* dataset and highlight the actual anomaly in green, while the anomaly identified by the different methods is highlighted in red. We find out that PBAD and SAND can cover the whole anomaly subsequence. NETS is also able to match some true anomaly (on the right), while the subsequence methods usually have a lag in the result. We perform another experiment with *Uni-sub-s* and find similar results for the subsequence methods in terms of length ℓ. However, point methods can not give accurate results in such complex cases (in Figure 12(c)). Combining these two observations, we argue that point methods facilitate the detection of some global outliers, while anomalous patterns requiring long-term context are hard to detect.

Summary. (1) Point methods can also work well for *global* subsequence anomalies with extreme values, which may help relax the length input requirement. (2) The length parameter ℓ, which is close to the actual anomaly length, gives good results. Adaptive input length are needed to deal with real-world situations.

4.3.4 Effect of Metrics and Adjustment. Using point metrics for datasets with subsequence anomaly will bias the accuracy of many time series anomaly detection systems by not capturing specific properties [53]. For a real-time application, it might be more important to detect the earlier part of an anomaly to reduce response

Table 6: F-measure w/o point-adjust under different metrics

Algorithm	Point Metric						Range Metric		
	Twitter	ECG	inc/alg	Exathlon	Uni-sub-g	inc/alg	Exathlon	Uni-sub-g	inc/alg
TranAD	0.455	0.224	39.7%	0.245	0.299	6.1%	0.484	0.345	-68.9%
TranAD*	0.466	0.397		0.246	0.414		**0.246**	0.039	
USAD	0.274	0.286	31.0%	0.246	0.330	6.1%	0.245	0.365	-44.8%
USAD*	0.289	0.448		0.246	0.452		0.245	0.038	
Omni	0.138	0.318	36.4%	0.245	0.267	14.3%	0.409	0.240	-53.5%
Omni*	0.166	0.483		0.246	0.553		0.246	0.079	
RCoder	-	0.441	43.0%	-	-	-	-	-	-
RCoder*	-	0.631		-	-		-	-	
GDN	0.449	0.274	26.9%	0.238	0.267	14.7%	**0.851**	0.239	-69.1%
GDN*	0.460	0.410		0.246	0.554		0.246	0.079	
NETS	**0.802**	**0.461**	25.3%	**0.286**	0.942	42.8%	0.217	0.942	-28.5%
NETS*	**0.852**	**0.666**		0.754	1		0.080	1	
Stare	0.334	0.252	66.6%	0.253	0.653	14.5%	0.189	0.438	**-85.0%**
Stare*	0.334	0.589		0.312	0.725		0.018	0.090	
SHESD	0.209	0.339	**82.8%**	0.257	**0.978**	**43.8%**	0.227	**0.979**	-29.0%
SHESD*	0.510	0.488		**0.776**	**1**		0.090	**1**	
inc/data	23.1%	58.3%		18.5%	12.5%		-53.4%	-54.8%	

time [32]. Therefore, the overlap of predicted anomalies and actual anomalies should be addressed. To deal with interpretations of such overlaps and the problem of label imbalance, some methods [3, 52, 58] use a point-adjust method [62] that converts false negatives to true positives. As shown in Figure 14, for each point in the anomaly segment of the ground truth, if it is detected as an anomaly by the proposed algorithm, all observations in the subsequence will be considered to have been correctly detected as anomalies. Therefore, such a method ignores latency and reports much higher accuracy than it actually is. RCoder proposes a different way to adjust predictions [1]. However, we focus on analyzing the more widely used point-adjust method in this work.

Settings. We conduct experiments with both point (*Twitter*, *ECG*) and subsequence datasets (*Exathlon*, *Uni-sub-g*). The results of point datasets are evaluated by the point metric and the results of subsequence datasets are evaluated by both point and range metrics. Results with the point-adjust method are denoted by $*$.

Results. Table 6 shows the f-measure in different cases. 'inc/alg' gives the average promotion after applying point-adjust method per method and 'inc/data' shows this promotion per data. Point-adjust method can have a great impact on the evaluation as follows: (1) Overall, the algorithm will have an average promotion of 27.0% for point datasets and a higher promotion of 31.2% for subsequence datasets under point metrics. (2) For subsequence datasets with range metrics, the algorithms have an average negative promotion of −67.6%. (3) TranAD and Omni have a higher f-measure (0.245) than GDN (0.238) under the point metric, but GDN performs far better (0.851) than TranAD (0.484) and Omni (0.409) on *Exathlon* under range metric. (4) Stare has a higher f-measure (0.334) than SHESD (0.209), but after the adjustment, SHESD will perform better (0.510) than Stare (0.334) on *Twitter*. Such inversions can confuse anomaly detection systems and influence user preferences in selecting appropriate methods..

Summary. (1) point-adjust method tends to report 'false' higher results for algorithms and can lead to misleading analyzes. (2) The use of range metrics on datasets with subsequence anomalies is preferable as it leads to more reasonable and robust results.

4.3.5 Effect of Threshold. Threshold is a key hyper-parameter for anomaly detection problems whose effect has seldom been discussed in existing works, as stated in Section 1.2. We will test the

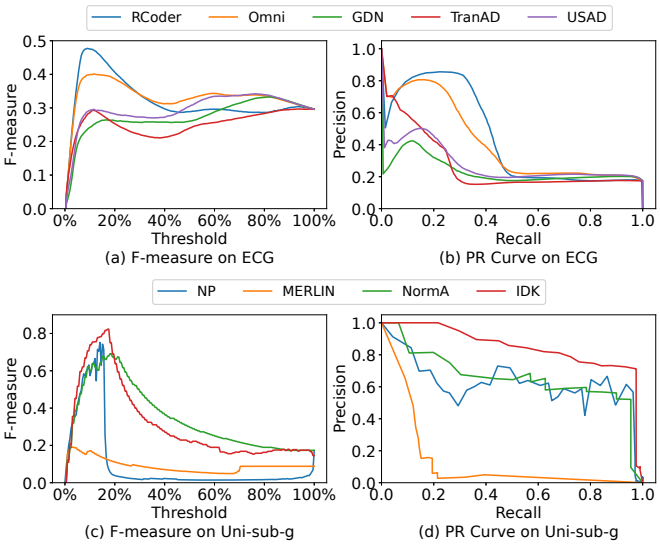

Figure 15: Varying thresholds on (a-b) ECG (c-d) Uni-sub-g

Table 7: F-measure w/o POT method

Data Set	TranAD	TranAD-P	USAD	USAD-P	Omni	Omni-P	GDN	GDN-P
ECG	0.248	0.350	0.286	0.338	0.224	0.345	0.271	**0.353**
DLR	0.152	0.007	**0.154**	0.070	0.065	0.070	0.060	0.069
TAO	**0.181**	0.181	0.180	0.181	0.181	0.181	0.012	0.181
UNI	0.179	0.182	0.158	0.182	**0.660**	0.181	0.181	0.183

robustness of each algorithm w.r.t. threshold and the impact of the POT [51] method for automatic threshold selection.

Settings. We unify the threshold setting for all methods in the following way: let the threshold be the ratio of points/sequences identified as anomalies. 0% means all are normal, while 100% means all are anomalies. However, NETS , Stare, and SHESD have different logic in identifying anomalies (more than one factor is considered), so we report only other methods in this part. The overall performance is the area included by the recall and precision curves. Besides, the results of using POT method are presented in Table 7.

Results. Figure 15(c, d) show the results on *Uni-sub-g*. We note that NP, despite having a better best F-measure than NormA, drops sharply as the threshold approaches 20%. In contrast, NormA shows better robustness as the curve is smoother. It is interesting to note that IDK performs best in this experiment, but does not outperform in Figure 9. The reason for this is that in all other experiments we perform a grid search on the validation set to find the proper (hyper)-parameters, but there is a small difference between the validation set and the test set. As for the point methods shown in Figures 15(a, b), RCoder shows the overall best performance among all the deep learning methods, while Omni shows better robustness.

In terms of the impact of POT method, GDN always have a better result after using it. Other methods show different preferences on different datasets. In around 55% cases, POT can achieve similar or slightly better results than grid search on validation set.

Summary. (1) Automatic threshold selection methods can still be improved to take effect in practical uses. (2) IDK has the best overall

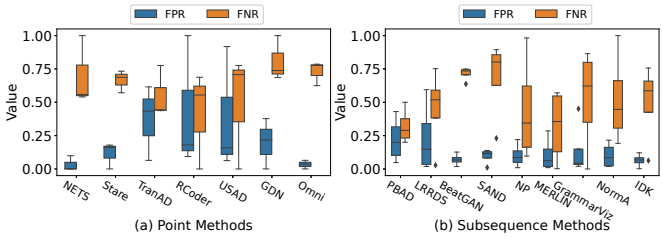

Figure 16: FPR and FNR on real datasets

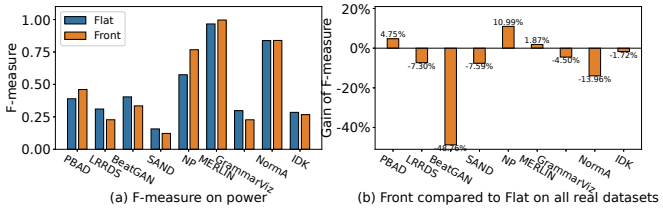

Figure 17: Early detection under Front metric

performance, while NormA is more robust, specifically when the threshold is above 20%. (3) RCoder has the best overall performance, while Omni is more robust.

4.3.6 Application. In this section, we focus on anomaly detection in real-world applications. Scenario 1: Some applications are more interested in positive outcomes, such as cancer detection, where we do not want cancer patients to go undetected. Others are more interested in negative outcomes, such as when we do not want a good email to become spam. Scenario 2: In the previous study, we assumed that all positions of an outlier range are equally important. Therefore, larger overlaps lead to a higher score of the metric. However, in practice, there are many situations where early response is critical, e.g., cancer detection or real-time systems.

Settings. We employ experiments on all real datasets with point and subsequence methods for Scenario 1, where $FPR = \frac{FP}{TN+FP}$, $FNR = \frac{FN}{TP+FN}$. Scenario 2 is particularly suitable for subsequence methods. *Flat* denotes the metric with the same score for all positions, while *Front* assigns more weight to the early positions of the subsequence anomaly [53].

Results. All algorithms generally have a higher false negative rate and a lower false positive rate, as can be seen in Figure 16. This suggests that they are more suitable for negative applications (not reporting false anomalies) and are more capable of detecting normal samples than abnormal samples. In particular, NETS and Omni have low FPR and are recommended for negative cases such as spam detection. On the other hand, TranAD and RCoder are recommended for positive applications since they manage to report all anomalies. Similarly, PBAD is more suitable than BeatGAN for such cases due to its lower FNR.

Figure 17(a) shows the performance of the algorithms under *Flat* and *Front* metrics on *Power*. (Similar results are observed on other real-world datasets.) BeatGAN has a higher f-measure than PBAD under the *Flat* metric, but achieves a lower score when the *Front* metric is used, indicating that it is not suitable for early detection.

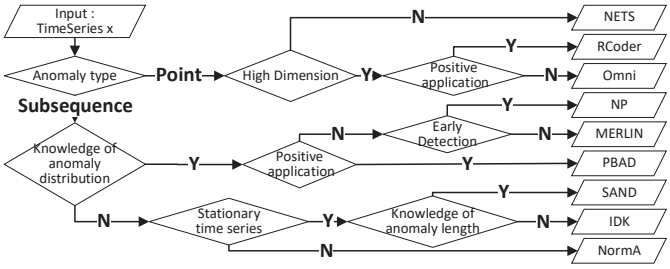

Figure 18: A practical guide for timeseries anomaly detection

We see similar results in Figure 17(b), where the average comparison is tested over all real datasets. BeatGAN scores a 49% drop compared to its performance under the *Flat* metric, showing that it captures outliers with latency. NP appears to be the best performing algorithm for early detection, with a performance improvement of 10.99%.

Summary. (1) Specific metrics are required for different scenarios. (2) All subsequence methods are better suited for negative applications and PBAD can be selected for positive cases. (3) NP is recommended for early detection. (4) Omni is recommended for negative applications, while RCoder is better for positive ones.

5 DISCUSSIONS

In this paper, a taxonomy of anomaly detection methods is presented and systematic experimental intra- and inter-class comparisons are proposed. Detailed findings are shown in the Summary part of Section 4. We first summarize these findings into a practical guide and finally highlight some research opportunities below.

A Practical Guide. A practical guide for timeseries anomaly detection is presented in Figure 18 based on the experimental findings on various aspects like type of anomaly, dimensionality, type of application etc. Such a guide is formed according to current work and future work is still encouraged.

Explainability. The explainability of anomaly detection methods has raised many concerns in recent years [37]. A decision maker may be more interested in the cause of the occurrence of outliers so that they can take appropriate action, especially in the area of IoT data [27]. Developing a method that provides both high accuracy and reasonable explainability may be of interest for future work.

ACKNOWLEDGMENTS

Aoqian Zhang is supported by the NSFC (Grant Nos. 6210070801, U21B2007). Guoren Wang is supported by the NSFC (Grant Nos. 61732003, U2001211). Ye Yuan is supported by the National Key R&D Program of China(Grant No. 2022YFB2702100), the NSFC (Grant Nos. 61932004, 62225203, U21A20516) and the DITDP (Grant No. JCKY2021211B017). We also thank all the members of our community who open sourced their data and codes, which help us a lot on this work.

REFERENCES

[1] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. 2021. Practical Approach to Asynchronous Multivariate Time Series Anomaly Detection and Localization. In *KDD*. ACM, 2485–2494.

[2] Charu C. Aggarwal. 2013. *Outlier Analysis*. Springer.

[3] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. 2020. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *KDD*. ACM, 3395–3404.

[4] Vic Barnett, Toby Lewis, et al. 1994. *Outliers in statistical data*. Vol. 3. Wiley New York.

[5] Ane Blázquez-García, Angel Conde, Usue Mori, and José Antonio Lozano. 2021. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Comput. Surv.* 54, 3 (2021), 56:1–56:33.

[6] Paul Boniol, Michele Linardi, Federico Roncallo, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. 2021. Unsupervised and scalable subsequence anomaly detection in large data series. *VLDB J.* 30, 6 (2021), 909–931.

[7] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J. Franklin. 2021. SAND: Streaming Subsequence Anomaly Detection. *Proc. VLDB Endow.* 14, 10 (2021), 1717–1729.

[8] Mohammad Braei and Sebastian Wagner. 2020. Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. *CoRR* abs/2004.00433 (2020).

[9] Bernardo Branco, Pedro Abreu, Ana Sofia Gomes, Mariana S. C. Almeida, João Tiago Ascensão, and Pedro Bizarro. 2020. Interleaved Sequence RNNs for Fraud Detection. In *KDD*. ACM, 3101–3109.

[10] Mikel Canizo, Isaac Triguero, Angel Conde, and Enrique Onieva. 2019. Multi-head CNN-RNN for multi-time series anomaly detection: An industrial case study. *Neurocomputing* 363 (2019), 246–260.

[11] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep Learning for Anomaly Detection: A Survey. *CoRR* abs/1901.03407 (2019).

[12] Dhruv Choudhary, Arun Kejariwal, and Francois Orsini. 2017. On the Runtime-Efficacy Trade-off of Anomaly Detection Techniques for Real-Time Streaming Data. *CoRR* abs/1710.04735 (2017).

[13] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition. *J. Off. Stat* 6, 1 (1990), 3–73.

[14] Andrew A. Cook, Goksel Misirli, and Zhong Fan. 2020. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet Things J.* 7, 7 (2020), 6481–6494.

[15] Zahra Zamanzadeh Darban, Geoffrey I. Webb, Shirui Pan, Charu C. Aggarwal, and Mahsa Salehi. 2022. Deep Learning for Time Series Anomaly Detection: A Survey. *CoRR* abs/2211.05244 (2022).

[16] Janez Demsar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* 7 (2006), 1–30.

[17] Ailin Deng and Bryan Hooi. 2021. Graph Neural Network-Based Anomaly Detection in Multivariate Time Series. In *AAAI*. AAAI Press, 4027–4035.

[18] Ethan W. Dereszynski and Thomas G. Dietterich. 2011. Spatiotemporal Models for Data-Anomaly Detection in Dynamic Environmental Monitoring Campaigns. *ACM Trans. Sens. Networks* 8, 1 (2011), 3:1–3:36.

[19] Len Feremans, Vincent Vercruyssen, Boris Cule, Wannes Meert, and Bart Goethals. 2019. Pattern-Based Anomaly Detection in Mixed-Type Time Series. In *ECML/PKDD (1) (Lecture Notes in Computer Science)*, Vol. 11906. Springer, 240–256.

[20] Milton Friedman. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association* 32, 200 (1937), 675–701.

[21] Markus Goldstein and Seiichi Uchida. 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one* 11, 4 (2016), e0152173.

[22] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. 2014. Outlier Detection for Temporal Data: A Survey. *IEEE Trans. Knowl. Data Eng.* 26, 9 (2014), 2250–2267.

[23] D. M. Hawkins. 1980. *Identification of Outliers*. Springer.

[24] Yuanduo He, Xu Chu, and Yasha Wang. 2020. Neighbor Profile: Bagging Nearest Neighbors for Unsupervised Time Series Mining. In *ICDE*. IEEE, 373–384.

[25] Jordan Hochenbaum, Owen S. Vallis, and Arun Kejariwal. 2017. Automatic Anomaly Detection in the Cloud Via Statistical Learning. *CoRR* abs/1704.07706 (2017).

[26] Min Hu, Xiaowei Feng, Zhiwei Ji, Ke Yan, and Shengchen Zhou. 2019. A novel computational approach for discord search with local recurrence rates in multivariate time series. *Inf. Sci.* 477 (2019), 220–233.

[27] Ruihong Huang, Zhiwei Chen, Zhicheng Liu, Shaoxu Song, and Jianmin Wang. 2019. TsOutlier: Explaining Outliers with Uniform Profiles over IoT Data. In *IEEE BigData*. IEEE, 2024–2027.

[28] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Söderström. 2018. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. In *KDD*. ACM, 387–395.

[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. *Proc. VLDB Endow.* 14, 11 (2021), 2613–2626.

[30] Eamonn J. Keogh, Jessica Lin, Sang-Hee Lee, and Helga Van Herle. 2007. Finding the most unusual time series subsequence: algorithms and applications. *Knowl. Inf. Syst.* 11, 1 (2007), 1–27. https://doi.org/10.1007/s10115-006-0034-6

[31] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. 2021. Revisiting Time Series Outlier Detection: Definitions and Benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. https://openreview.net/forum?id=r8IvOsnHchr

[32] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. 2015. Generic and Scalable Framework for Automated Time-series Anomaly Detection. In *KDD*. ACM, 1939–1947.

[33] N. Jesper Larsson and Alistair Moffat. 1999. Offline Dictionary-Based Compression. In *Data Compression Conference, DCC 1999, Snowbird, Utah, USA, March 29-31, 1999*. IEEE Computer Society, 296–305.

[34] Alexander Lavin and Subutai Ahmad. 2015. Evaluating Real-Time Anomaly Detection Algorithms - The Numenta Anomaly Benchmark. In *ICMLA*. IEEE, 38–44.

[35] Kim-Hung Le and Paolo Papotti. 2020. User-driven Error Detection for Time Series with Events. In *ICDE*. IEEE, 745–757.

[36] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. 2019. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. In *ICANN (4) (Lecture Notes in Computer Science)*, Vol. 11730. Springer, 703–716.

[37] Zhong Li, Yuxuan Zhu, and Matthijs van Leeuwen. 2022. A Survey on Explainable Anomaly Detection. *CoRR* abs/2210.06959 (2022).

[38] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *ICDM*. IEEE Computer Society, 413–422.

[39] Shenghua Liu, Bin Zhou, Quan Ding, Bryan Hooi, Zheng bo Zhang, Huawei Shen, and Xueqi Cheng. 2022. Time series anomaly detection with adversarial reconstruction networks. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[40] Yue Lu, Renjie Wu, Abdullah Mueen, Maria A. Zuluaga, and Eamonn J. Keogh. 2022. Matrix Profile XXIV: Scaling Time Series Anomaly Detection to Trillions of Datapoints and Ultra-fast Arriving Data Streams. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 1173–1182.

[41] Aditya P. Mathur and Nils Ole Tippenhauer. 2016. SWaT: a water treatment testbed for research and training on ICS security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks, CySWater@CPSWeek 2016, Vienna, Austria, April 11, 2016*. IEEE Computer Society, 31–36. https://doi.org/10.1109/CySWater.2016.7469060

[42] Takaaki Nakamura, Makoto Imamura, Ryan Mercer, and Eamonn J. Keogh. 2020. MERLIN: Parameter-Free Discovery of Arbitrary Length Anomalies in Massive Time Series Archives. In *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*. IEEE, 1190–1195.

[43] Craig G. Nevill-Manning and Ian H. Witten. 1997. Identifying Hierarchical Structure in Sequences: A linear-time algorithm. *J. Artif. Intell. Res.* 7 (1997), 67–82.

[44] Antonios Ntroumpogiannis, Michail Giannoulis, Nikolaos Myrtakis, Vassilis Christophides, Eric Simon, and Ioannis Tsamardinos. 2023. A meta-level analysis of online anomaly detectors. *VLDB J.* 32, 4 (2023), 845–886.

[45] John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S Tsay, Aaron Elmore, and Michael J Franklin. 2022. Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2774–2787.

[46] John Paparrizos and Luis Gravano. 2015. k-Shape: Efficient and Accurate Clustering of Time Series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives (Eds.). ACM, 1855–1870.

[47] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin. 2022. TSB-UAD: An End-to-End Benchmark Suite for Univariate Time-Series Anomaly Detection. *Proc. VLDB Endow.* 15, 8 (2022), 1697–1711.

[48] Bernard Rosner. 1975. On the detection of many outliers. *Technometrics* 17, 2 (1975), 221–227.

[49] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly Detection in Time Series: A Comprehensive Evaluation. *Proc. VLDB Endow.* 15, 9 (2022), 1779–1797.

[50] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, and Susan Frankenstein. 2018. GrammarViz 3.0: Interactive Discovery of Variable-Length Time Series Patterns. *ACM Trans. Knowl. Discov. Data* 12, 1 (2018), 10:1–10:28.

[51] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouët. 2017. Anomaly Detection in Streams with Extreme Value Theory. In *KDD*. ACM, 1067–1075.

[52] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *KDD*. ACM, 2828–2837.

[53] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. 2018. Precision and Recall for Time Series. In *NeurIPS*. 1924–1934.

[54] Markus Thill, Wolfgang Konen, and Thomas Bäck. 2017. Online anomaly detection on the webscope S5 dataset: A comparative study. In *EAIS*. IEEE, 1–8.

[55] Kai Ming Ting, Zongyou Liu, Hang Zhang, and Ye Zhu. 2022. A New Distributional Treatment for Time Series and An Anomaly Detection Investigation. *Proc. VLDB Endow.* 15, 11 (2022), 2321–2333.

[56] Kai Ming Ting, Bi-Cun Xu, Takashi Washio, and Zhi-Hua Zhou. 2020. Isolation Distributional Kernel: A New Tool for Kernel based Anomaly Detection. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 198–206.

[57] Luan Tran, Liyue Fan, and Cyrus Shahabi. 2016. Distance-based Outlier Detection in Data Streams. *Proc. VLDB Endow.* 9, 12 (2016), 1089–1100.

[58] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proc. VLDB Endow.* 15, 6 (2022), 1201–1214.

[59] Chen Wang, Xiangdong Huang, Jialin Qiao, Tian Jiang, Lei Rui, Jinrui Zhang, Rong Kang, Julian Feinauer, Kevin Mcgrail, Peng Wang, Diaohan Luo, Jun Yuan, Jianmin Wang, and Jiaguang Sun. 2020. Apache IoTDB: Time-series database for Internet of Things. *Proc. VLDB Endow.* 13, 12 (2020), 2901–2904.

[60] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. Springer, 196–202.

[61] Renjie Wu and Eamonn Keogh. 2021. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[62] 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 *WWW*. ACM, 187–196.

[63] Hui Yang, Satish T. S. Bukkapatnam, and Leandro G. Barajas. 2011. Local recurrence based performance prediction and prognostics in the nonlinear and nonstationary systems. *Pattern Recognit.* 44, 8 (2011), 1834–1840.

[64] Dragomir Yankov, Eamonn J. Keogh, and Umaa Rebbapragada. 2008. Disk aware discord discovery: finding unusual time series in terabyte sized datasets. *Knowl. Inf. Syst.* 17, 2 (2008), 241–262.

[65] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn J. Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*. IEEE Computer Society, 1317–1322.

[66] Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. 2019. NETS: Extremely Fast Outlier Detection from a Data Stream via Set-Based Processing. *Proc. VLDB Endow.* 12, 11 (2019), 1303–1315.

[67] Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. 2020. Ultrafast Local Outlier Detection from a Data Stream with Stationary Region Skipping. In *KDD*. ACM, 1181–1191.

[68] Mohammed J. Zaki and Wagner Meira Jr. 2014. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.

[69] Yong Zou, Marco Thiel, M. Carmen Romano, and Jürgen Kurths. 2007. Analytical Description of Recurrence Plots of Dynamical Systems with Nontrivial Recurrences. *Int. J. Bifurc. Chaos* 17, 12 (2007), 4273–4283.