Anomaly Detection in Time Series: A Comprehensive Evaluation

Sebastian Schmidl[*]
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany
sebastian.schmidl@hpi.de

Phillip Wenig[*]
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany
phillip.wenig@hpi.de

Thorsten Papenbrock
Philipps University of Marburg
Marburg, Germany
papenbrock@informatik.uni-marburg.de

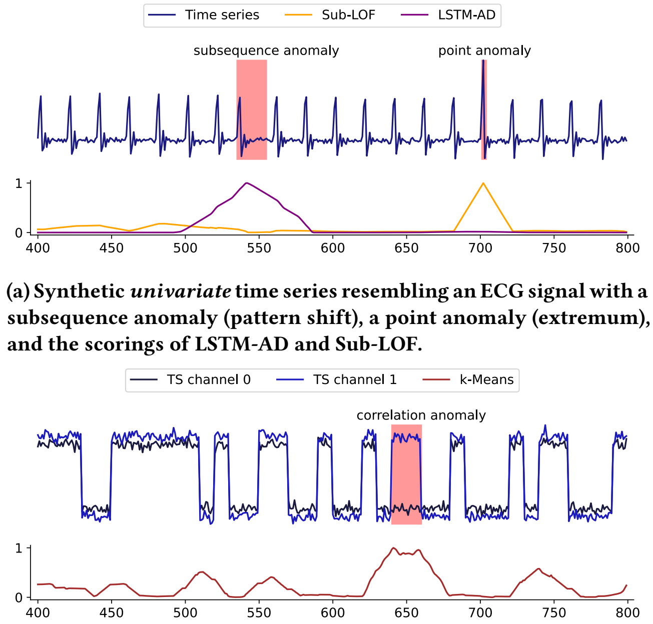
ABSTRACT

Detecting anomalous subsequences in time series data is an important task in areas ranging from manufacturing processes over finance applications to health care monitoring. An anomaly can indicate important events, such as production faults, delivery bottlenecks, system defects, or heart flicker, and is therefore of central interest. Because time series are often large and exhibit complex patterns, data scientists have developed various specialized algorithms for the automatic detection of such anomalous patterns. The number and variety of anomaly detection algorithms has grown significantly in the past and, because many of these solutions have been developed independently and by different research communities, there is no comprehensive study that systematically evaluates and compares the different approaches. For this reason, choosing the best detection technique for a given anomaly detection task is a difficult challenge.

This comprehensive, scientific study carefully evaluates most state-of-the-art anomaly detection algorithms. We collected and re-implemented 71 anomaly detection algorithms from different domains and evaluated them on 976 time series datasets. The algorithms have been selected from different algorithm families and detection approaches to represent the entire spectrum of anomaly detection techniques. In the paper, we provide a concise overview of the techniques and their commonalities; we evaluate their individual strengths and weaknesses and, thereby, consider factors, such as effectiveness, efficiency, and robustness. Our experimental results should ease the algorithm selection problem and open up new research directions.

PVLDB Reference Format:
Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly Detection in Time Series: A Comprehensive Evaluation. PVLDB, 15(9): 1779 - 1797, 2022.
doi:10.14778/3538598.3538602

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://hpi.de/naumann/s/time-series-anomaly-detection-evaluation.



(a) Synthetic *univariate* time series resembling an ECG signal with a subsequence anomaly (pattern shift), a point anomaly (extremum), and the scorings of LSTM-AD and Sub-LOF.

(b) Synthetic *multivariate* time series with a correlation anomaly and the scoring of k-Means.

Figure 1: Example time series with anomalies and scorings.

1 ANOMALY DETECTION WILDERNESS

A *data series* is an ordered sequence of data points. The data points describe some object or process property based on a continuous measure, such as temperature (e. g., physics), mass (e. g., chemistry), angle (e. g., astronomy), position (e. g., geology), or speed (e. g., mechanical engineering). If the order is based on time, the sequence is generally referred to as a *time series*. Regardless of the ordering measure, the recording of the data points usually follows discrete, equally-spaced intervals. For this reason and because most data series analytics algorithms are agnostic of the reference measure, we use the terms *data series*, *time series*, and *sequence* interchangeably.

The data points of a time series record are one or multiple real-valued variables. Each variable models one *channel* of the time series. If the data points consist of only one variable, the time series is called *univariate*; otherwise, it is *multivariate*. Figure 1 shows an example with two time series: a univariate time series on simulated electrocardiogram (ECG) signals and a synthetically generated multivariate time series with random mode-jumps. An *anomaly* in such time series is a point (e. g., outlier) or point sequence (e. g., irregularity) that deviates w. r. t. some measure, model, or embedding from

[*] Both authors contributed equally to this work.

Proceedings of the VLDB Endowment, Vol. 15, No. 9 ISSN 2150-8097.
doi:10.14778/3538598.3538602

the regular patterns of the sequence. In multivariate time series, these pattern deviations can occur in any single channel and also in the correlation of channels (e. g., skew) as shown in Figure 1b. Anomalous subsequences might have varying lengths and they might re-appear in the same time series. Depending on the domain of a time series, its anomalies can describe important events, such as heart failures in cardiology [4], structural defects in jet turbine engineering [145], or ecosystem disturbances in earth sciences [35].

The different anomaly types, pattern models, and time series properties led to the development of a multitude of different anomaly detection algorithms, about which various surveys exist [12, 20, 27, 28, 29, 37, 38, 53, 63]. In total, we collected 158 publications about anomaly detection algorithms for time series datasets. Many of them follow very similar detection approaches, but the general variety of approaches is remarkably high, ranging from simple outlier detection over statistical analysis, signal processing, and data mining to deep learning approaches. Because all of these approaches exhibit individual strengths and weaknesses, selecting a suitable algorithm for a given anomaly detection task is extremely difficult. Figure 1a, for example, shows that the distance method Sub-LOF detects the point anomaly very well, because it is sensitive to value magnitudes and change amplitudes; it, however, fails to detect the subsequence anomaly, because it is insensitive to point orders. LSTM-AD as a forecasting method, on the contrary, learns normal patterns and heavily relies on seasonality/periodicity; for this reason, it easily identifies the anomalous subsequence, but since it is robust to noise, it ignores the point anomaly.

Because there is no comprehensive scientific study that evaluates time series anomaly detection algorithms, it is completely unclear how well they perform w. r. t. varying datasets, anomaly types, and parameter settings. Considering the existing evaluations (surveys [20, 37] and individual algorithms), we find that they usually consider only a tiny fraction of related work algorithms and that they are often based on trivial, cherry-picked, biased, mislabeled, unrealistic, or only few datasets. For this reason, Wu and Keogh even claimed that all "*current time series anomaly detection benchmarks are flawed*" [147].

To create a meaningful evaluation, we collected and re-implemented a significant amount of 71 anomaly detection algorithms that represent a broad spectrum of anomaly detection families. For their evaluation, we put together 976 time series datasets from different domains and generated several datasets with artificial, but particularly interesting anomalies. The variety of algorithms and datasets used in this experimental study should provide a clear and reliable picture of the state-of-the-art in time series anomaly detection. In summary, this study makes the following contributions:

(1) **Short survey:** We provide a classification of 158 time series anomaly detection techniques together with short descriptions of the different algorithm families; we explain our algorithm choices for this experimental evaluation and the general ideas behind the algorithms. We offer the *most comprehensive survey* and highlight *novel dimensions.* (Section 3)
(2) **Exhaustive evaluation:** We evaluate 71 representative anomaly detection algorithms on 976 uni- and multivariate time series datasets and report their performance in terms of accuracy and runtime; we analyze the strengths and weaknesses of different algorithms/techniques and, thereby, consider a variety of time series characteristics. This offers the *broadest view on algorithms, datasets, and features.* (Section 4)

In the paper, we provide practical research insights (**RIs 1 to 14**) that should help experts select the optimal algorithm for their anomaly detection task. Our study also shows the current state-of-the-art in time series anomaly detection and, therefore, serves as an entry point for researchers into the topic. We indicate potential for future research and, in this way, support the development of new algorithms that are now able to address specific issues, exploit certain capabilities, and ultimately advance the field of anomaly detection.

In this evaluation, we focus on algorithms that have been scientifically published and/or integrated into popular time series libraries – in other words, anomaly detection approaches that are actually available to users. This study, therefore, judges concrete implementations and only indirectly the techniques and measures that they use. For a fair comparison, we tried not to modify, enhance or combine algorithms and state clearly, where modifications were necessary. Concerning parameterization, we invest the same careful and systematic effort into each algorithm. This might, in the end, not result in the optimal parameter configurations for each algorithm, but it leads to performance measurements that are representative of the algorithms in practice, where users are similarly restricted in their parameterization capabilities.

2 TIME SERIES AND ANOMALIES

In this study, we investigate algorithms for the detection of anomalous subsequences in time series data. A *time series* (or *data series* in general) is an ordered set $T = \{T_1, T_2, \ldots, T_m\}$ of m real-valued, potentially multidimensional data points $T_i \in \mathbb{R}^n$. A *subsequence* $T_{i,j} = \{T_i, \ldots, T_j\} \subseteq T$ is a contiguous segment of T with length $|T_{i,j}| = j - i + 1$ and $|T_{i,j}| \geq 1$. Our evaluation assumes that the data points are equidistant, which is true for most real-world time series and relieves the algorithms from interpreting the concrete continuous measures (time, mass, angle, etc.); data series not following this assumption need to be discretized.

The concrete definition of an *anomaly* differs in the literature. Individual, abnormal data points are usually referred to as *outliers* [2]. Given the ordering characteristic of time series data, anomalous data points have been classified into *point*, *sequence*, and *contextual* anomalies [20] to consider length and context information in the classification. When subdividing a time series into fixed-size subsequences (e. g., via windowing), the (ab)normality of individual patterns can be defined by their distance to (nearest) neighbor patterns w. r. t. certain similarity and distance measures. This leads i. a. to the anomaly concept of *discords* [70]. Because we aim to evaluate anomaly detection approaches with different interpretations of anomalous behavior, we use the following general definition:

Definition 2.1. A time series *anomaly* is a sequence of data points $T_{i,j}$ of length $j - i + 1 \geq 1$ that deviates w. r. t. some characteristic embedding, model, and/or similarity measure from frequent patterns in the time series T.

Time series anomaly detection is the process of marking anomalies in a given time series. Two related analytics tasks are *time* *series forecasting* [65] and *time series classification* [87]. Time series forecasting describes the process of predicting the future progression of a time series; although this activity is not in the focus of this study, many anomaly detection approaches, such as DeepLSTM [31], Torsk [60], ARIMA [65] and NumentaHTM [3], use time series prediction internally and determine anomalies using the deviation of predicted from observed values. Time series classification, in contrast, describes the process of assigning entire time series to certain classes; it is often used as a post-processing step to classify detected anomalies into domain-specific classes, but some anomaly detection algorithms, such as PS-SVM [85], SR-CNN [112], COPOD [80], and NoveltySVR [86], use classification techniques internally to assign time series decompositions to predefined categories.

The anomaly detection algorithms that we investigate in this study propose different embeddings, models, and similarity functions. Some of which are based on statistical analysis, others on machine learning, and again others on data mining. In this paper, we aim to evaluate these techniques by measuring the algorithm's accuracy in various (real-world) scenarios. For this comparison, the results need to be transformed into a uniform output format. The most uniform result format for all anomaly detection algorithms is a point-wise scoring of the data points. We define it as follows:

Definition 2.2. A time series *scoring* $S = \{s_1, s_2, \ldots, s_m\}$ with $s_i \in \mathbb{R}$ is the result of a time series anomaly detection algorithm that assigns each data point $T_i \in T$ an anomaly score $s_i \in S$. For any two scores s_i and s_j, it must be true that if $s_i > s_j$, then T_i is more anomalous than T_j (in their respective contexts).

Our re-implementations of the various algorithms transform all outputs (scores, confidences, distances, etc.) into time series *scorings* with properties from Definition 2.2. For algorithms that score entire subsequences $T_{i,j}$, we assign the subsequence score to all $T_k \in T_{i,j}$. If data points receive subsequence scores from multiple overlapping subsequences, which is when they are part of overlapping subsequences, we assign the average of these scores as individual point scores (maximum aggregation performed worse). The translation of anomaly scores into binary anomaly labels (e. g., 0 for normal and 1 for anomalous) via thresholding is an orthogonal, algorithm-independent problem that we do not consider in this evaluation. Hence, any thresholding that transforms scores into binary labels was (if possible) removed from the evaluated algorithms.

To evaluate the various scorings, we use three threshold-agnostic *Area Under the Curve* (AUC) measures: The Area under the *Receiver Operating Characteristics* Curve [57, 19] (AUC-ROC), the Area Under the *Precision-Recall* Curve [109, 39] (AUC-PR), and the Area Under the *ranged-based Precision, range-based Recall* Curve [131] (AUC-$\mathrm{P_T R_T}$). AUC-ROC and AUC-PR have been used in most evaluations of time series algorithms and, therefore, serve to relate our results to existing evaluation results. *Ranged-based Precision* ($\mathrm{P_T}$) and *range-based Recall* ($\mathrm{R_T}$) are more recent evaluation metrics specifically developed for time series anomalies [131]; AUC-$\mathrm{P_T R_T}$ simply calculates the area under the curve for these two new metrics. By also considering the ordering of the scorings, it offers a different perspective on the obtained results. All three metrics assign perfect scores of 1.0 if a threshold exists that clearly separates anomalous from normal points in the scoring, regardless of what that threshold actually is.

The AUC-PR contrasts the precision to the recall score, whereas the AUC-ROC uses the false-positives-rate (FPR) instead of precision. The number of negative samples (N) in a dataset has a higher influence on AUC-PR than on AUC-ROC because the precision is the ratio of true positives (TP) to all predicted positives (PP) (precision $= \frac{TP}{PP}$). A higher N introduces more potentially false positives (FP) and, thus, higher PP. The FPR, used by AUC-ROC, is the ratio $\frac{FP}{N}$. A higher N is potentially reflected in both FP and N. Therefore, the FPR is less influenced by N than precision is. The appropriate metric depends on the specific use-case: AUC-ROC rewards sensitive algorithms while AUC-PR rewards precise algorithms. AUC-$\mathrm{P_T R_T}$ softens the very strict preciseness requirements of AUC-PR to adapt the measure to subsequences.

3 ANOMALY DETECTION TECHNIQUES

This section provides a broad but also concise survey of existing anomaly detection algorithms that we consider in our evaluation. The algorithms originate from different research areas and belong to different method families. More specifically, we found anomaly detection algorithms in (i) Deep Learning, (ii) Stochastic Learning, (iii) Classic Machine Learning, (iv) Outlier Detection, (v) Statistics (Regression and Forecasting), (vi) Data Mining, and (vii) Signal Analysis. Overall, we collected 158 publications that each describes a unique approach for detecting anomalies in time series. Figure 2 shows all the collected publications grouped by their research area.

In our experimental evaluation, we consider a representative subset of 71 algorithms that cover all major types of approaches and method families. In Figure 2, the 71 considered algorithms are highlighted in *blue italics*. Table 1 gives a brief overview about the selected algorithms, their properties, and their implementation.

Table 1 is structured by two essential algorithm categories: the algorithms' supported *input dimensionality* (column "D") and their *learning type* (column "L"). The *input dimensionality* category describes if an algorithm can use inter-variable dependencies and correlations (*multivariate*) or not (*univariate*). Equal to the general outlier/anomaly detection problem, time series anomaly detection algorithms can also be grouped into three learning types (cf. [29]): *unsupervised*, *supervised*, and *semi-supervised*. Hodge and Austin call those categories TYPE I, TYPE II, and TYPE III respectively in their survey of outlier detection methods [63]. *Unsupervised* (TYPE I) algorithms separate anomalous points from the normal part of the time series without prior knowledge (no explicit training step is required). These algorithms assume that anomalous subsequences can be separated from normal subsequences because they, i. a., are less frequent, have different shapes and behaviors, or originate from a different distribution. *Supervised* (TYPE II) algorithms model normal and abnormal behavior in the time series and require a training step before they can be employed on a new time series. All points of the training time series must be marked as either normal (usually 0) or anomalous (usually 1). These algorithms learn to distinguish between the normal and the anomalous behavior of the training time series. Given an unseen test time series, the algorithms can, then, mark the anomalous subsequences that match their internal representation of anomalous behavior. Supervised algorithms are restricted in their ability to detect unseen anomalies and are, therefore, rarely used for anomaly detection in time series.

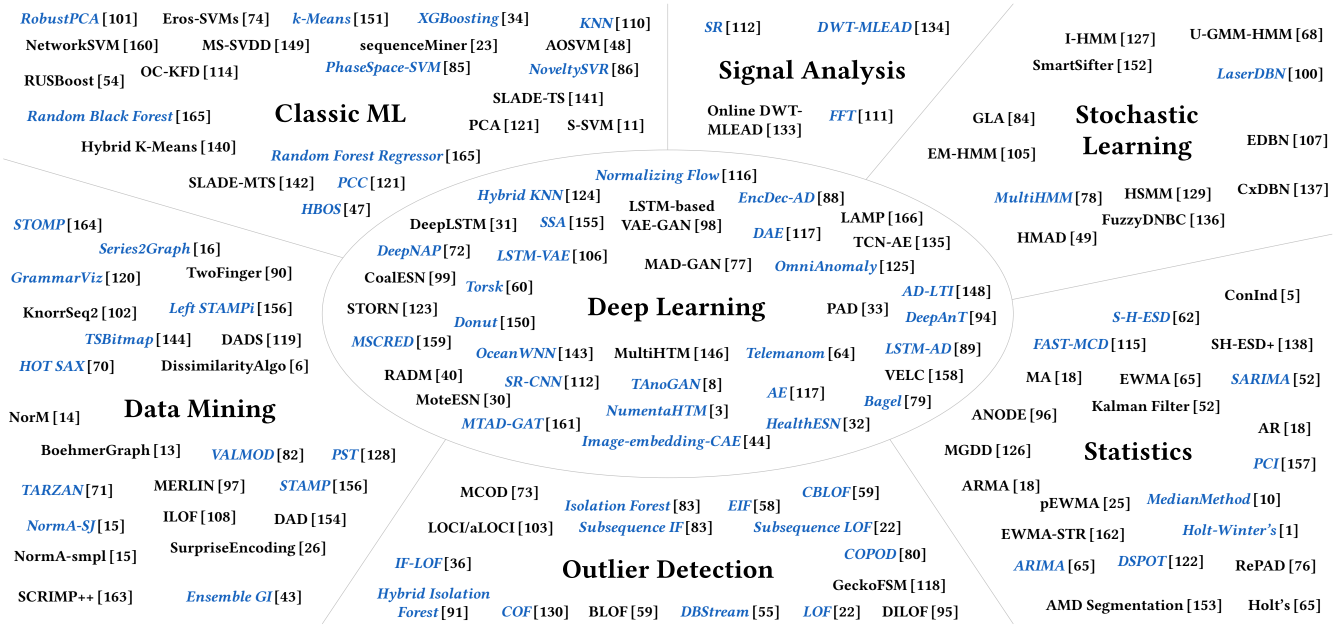


Figure 2: All 158 collected anomaly detection methods for time series data structured by their method family. Methods considered in this evaluation are highlighted in *blue italics*.

Semi-supervised (TYPE III) algorithms try to learn only the normal behavior of a training time series. This means that they should be trained on normal time series to build a model of the normal behavior. When applied to a test time series, all subsequences that do not conform to the normal behavior are marked as anomalous.

In addition to the input dimensionality and learning type categories, we also categorize time series anomaly detection algorithms into six method families: (i) forecasting (Section 3.1), (ii) reconstruction (Section 3.2), (iii) distance (Section 3.4), (iv) encoding (Section 3.3), (v) distribution (Section 3.5), and (vi) tree methods (Section 3.6). These method families characterize the algorithms by their general approach of determining the abnormality of specific points or subsequences within the time series. The following sections describe the method families in more detail and introduce the member algorithms that are evaluated in this study.

While we do not modify, enhance or combine algorithms that have been published in specific ways, we include Subsequence LOF (Sub-LOF) [22], Subsequence Isolation Forest (Sub-IF) [83], and Subsequence Fast-MCD (Sub-Fast-MCD) [115] as representatives of outlier detection techniques applied to time series subsequences, which is common practice in the subsequence anomaly detection research area [9, 15, 17, 119]. Practically, this means that instead of applying these algorithms to the individual multidimensional points of a multivariate time series, a univariate time series is first split into fixed-length subsequences using a sliding window and the algorithm is then applied to these subsequences. On request of our industry partner and to cover the widest range of approaches possible, we also included the Random Forest Regressor (RForest) [21], Random Black Forest (RBForest) [165], and XGBoosting [34] algorithms that are common semi-supervised machine learning techniques applied on subsequences for anomaly detection.

3.1 Forecasting Methods

Forecasting methods (symbol: ★) use a (continuously) learned model to forecast a number of time steps based on a current context window. The values for the forecasted data points depend solely on the time series' data points in the preceding context window and the previously learned model. The forecasted points are then compared to the observed values in the original time series to determine how anomalous the observed values are. Most forecasting methods use a sliding window with a stride of 1 to build the context window and forecast a single point at a time. The methods in this family differ most in the type of forecasting model that they use (i. e., the type of normal behavior model), in the way they build this model (i. e., the learning approach) and the calculation metric for the anomaly scores (i. e., the distance metric for forecasted and observed values). The representatives of this category that we consider in our evaluation are: AD-LTI [148], ARIMA [65], DeepAnT [94], DeepNAP [72], HealthESN [32], LSTM-AD [89], MedianMethod [10], MTAD-GAT [161], NumentaHTM [3], NoveltySVR [86], OceanWNN [143], RBForest [165], RForest [21], SARIMA [52], Telemanom [64], Torsk [60], Triple ES [1], and XGBoosting [34].

Usually, forecasting methods are trained in a semi-supervised way: The training time series without anomalies is used to learn the normal model of the data. A deviation from this normal, expected behavior in the test time series, i. e., a significant difference in the observed and forecasted data points, is regarded as anomalous. This applies to RBForest, RForest, OceanWNN, XGBoosting, AD-LTI, DeepAnT, DeepNAP, HealthESN, LSTM-AD (which can predict multiple points), MTAD-GAT, and Telemanom. However, NumentaHTM, NoveltySVR, Torsk (which can also predict multiple points),

Table 1: All 71 algorithms considered in our evaluation grouped by dimensionality ("D") and learning type ("L").

D	L	Method	Area	Family	Origin	Language
univariate	unsupervised	NoveltySVR [86]	Classic ML	● distance	own	Python
		PS-SVM [85]	Classic ML	● distance	own	Python
		Ensemble GI [43]	Data Mining	▼ encoding	own	Python
		GrammarViz [120]	Data Mining	▼ encoding	original	Java
		HOT SAX [70]	Data Mining	● distance	original	Python
		TSBitmap [144]	Data Mining	▼ encoding	community	Python
		NormA-SJ [15]	Data Mining	● distance	original	Python
		SAND [17]	Data Mining	● distance	original	Python
		Series2Graph [16]	Data Mining	▼ encoding	original	Python
		STAMP [156]	Data Mining	● distance	original	R
		STOMP [164]	Data Mining	● distance	original	R
		VALMOD [82]	Data Mining	● distance	original	R
		Left STAMPi [156]	Data Mining	● distance	original	Python
		SSA [155]	Data Mining	● distance	own	Python
		PST [128]	Data Mining	✚ trees	own	R
		NumentaHTM [3]	Deep L.	★ forecasting	original	Python
		Sub-LOF [22]	Outlier Det.	● distance	own	Python
		Sub-IF [83]	Outlier Det.	✚ trees	own	Python
		DWT-MLEAD [134]	Signal A.	▲ distribution	own	Python
		FFT [111]	Signal A.	🟩 reconstruction	own	Python
		SR [112]	Signal A.	🟩 reconstruction	original	Python
		S-H-ESD [62]	Statistics	▲ distribution	own	R
		DSPOT [122]	Statistics	▲ distribution	original	Python
		ARIMA [65]	Statistics	★ forecasting	own	Python
		MedianMethod [10]	Statistics	★ forecasting	own	Python
		SARIMA [52]	Statistics	★ forecasting	own	Python
		Triple ES [1]	Statistics	★ forecasting	own	Python
		PCI [157]	Statistics	🟩 reconstruction	own	Python
	semi-supervised	RForest [21]	Classic ML	★ forecasting	own	Python
		XGBoosting [34]	Classic ML	★ forecasting	own	Python
		TARZAN [71]	Data Mining	▼ encoding	original	Python
		HealthESN [32]	Deep L.	★ forecasting	own	Python
		OceanWNN [143]	Deep L.	★ forecasting	own	Pytorch
		Bagel [79]	Deep L.	🟩 reconstruction	original	Python
		Donut [150]	Deep L.	🟩 reconstruction	original	Pytorch
		IE-CAE [44]	Deep L.	🟩 reconstruction	own	Pytorch
		SR-CNN [112]	Deep L.	🟩 reconstruction	original	Pytorch
		Sub-Fast-MCD [115]	Statistics	▲ distribution	own	Python
multivariate	unsupervised	PCC [121]	Classic ML	🟩 reconstruction	community	Python
		HBOS [47]	Classic ML	● distance	community	Python
		k-Means [151]	Classic ML	● distance	own	Python
		KNN [110]	Classic ML	● distance	community	Python
		EIF [58]	Classic ML	✚ trees	original	Python
		Torsk [60]	Deep L.	★ forecasting	original	Pytorch
		CBLOF [59]	Outlier Det.	● distance	community	Python
		COF [130]	Outlier Det.	● distance	community	Python
		DBStream [55]	Outlier Det.	● distance	original	R
		LOF [22]	Outlier Det.	● distance	community	Python
		COPOD [80]	Outlier Det.	▲ distribution	community	Python
		IF-LOF [36]	Outlier Det.	✚ trees	own	Python
		iForest [83]	Outlier Det.	✚ trees	community	Python
	semi-supervised	RobustPCA [101]	Classic ML	🟩 reconstruction	community	Python
		RBForest [165]	Classic ML	★ forecasting	own	Python
		Hybrid KNN [124]	Deep L.	● distance	own	Pytorch
		DeepAnT [94]	Deep L.	★ forecasting	own	Pytorch
		DeepNAP [72]	Deep L.	★ forecasting	own	Pytorch
		LSTM-AD [89]	Deep L.	★ forecasting	own	Pytorch
		MTAD-GAT [161]	Deep L.	★ forecasting	own	Pytorch
		Telemanom [64]	Deep L.	★ forecasting	original	Tensorflow
		MSCRED [159]	Deep L.	🟩 reconstruction	own	Tensorflow
		AE [117]	Deep L.	🟩 reconstruction	own	Tensorflow
		DAE [117]	Deep L.	🟩 reconstruction	own	Tensorflow
		EncDec-AD [88]	Deep L.	🟩 reconstruction	own	Pytorch
		LSTM-VAE [106]	Deep L.	🟩 reconstruction	own	Tensorflow
		OmniAnomaly [125]	Deep L.	🟩 reconstruction	original	Tensorflow
		TAnoGan [8]	Deep L.	🟩 reconstruction	own	Pytorch
		Fast-MCD [115]	Statistics	▲ distribution	own	Python
		LaserDBN [100]	Stochastic L.	▼ encoding	own	Python
	superv.	NF [116]	Deep L.	▲ distribution	own	Pytorch
		HIF [91]	Outlier Det.	✚ trees	original	Python
		MultiHMM [78]	Stochastic L.	▼ encoding	own	Python

ARIMA, and SARIMA build their normal model directly on the test dataset using some number of initial points (unsupervised). For these initial points, the methods assume normality and do not calculate an anomaly score. The model is periodically rebuilt to adapt to changes in the data. The model updates are put to an extreme in MedianMethod and Triple ES. Both algorithms rebuild the entire normal model from every context window: The MedianMethod uses the median of the context window as forecast, while Triple ES fits a triple exponential smoothing model to each context window to forecast one subsequent point.

3.2 Reconstruction Methods

Reconstruction methods (symbol: 🟩) build a model of normal behavior by encoding subsequences of a normal training time series in a (low dimensional) latent space. To detect anomalies in a test time series, subsequences from the test series are reconstructed from the latent space and the reconstructed subsequences' values are then compared to the original, observed series values. The inputs to the reconstruction process are training windows (usually created using a sliding window with stride 1) that provide the time context to the model. Because the model is built only on normal data (semi-supervised), anomalous subsequences in the test series cannot be reconstructed by the model. Hence, the anomaly score can be calculated from the difference between the original and the reconstructed subsequences. Representatives of this category in our evaluation are: AutoEncoder (AE) [117], Bagel [79], DenoisingAutoEncoder (DAE) [117], Donut [150], EncDec-AD [88], FFT [111], Image-embedding CAE (IE-CAE) [44], LSTM-VAE [106], MSCRED [159], OmniAnomaly [125], PCI [157], PCC [121], RobustPCA [101], Spectral Residual (SR) [112], SR-CNN [112] and TAnoGan [8].

An exception to the semi-supervised training of these methods are the four unsupervised methods FFT, SR, PCC, and PCI: They encode the input subsequences of the test series into a predefined latent space and, thereby, deliberately loose information, i. e., precision, which is needed to capture anomalies. During the reconstruction process, not all details of the original subsequences can be recreated. Hence, the differences between the reconstructed subsequences and the original ones can be reported as anomaly scores.

3.3 Encoding Methods

Encoding methods (symbol: ▼) are similar to reconstruction methods in that they also encode subsequences of a time series in a low dimensional latent space. However, they do not attempt to reconstruct the subsequences from the latent space, but compute the anomaly score directly from the latent space representations. More specifically, the anomaly scores are attributed to the points which correspond to the encoded subsequences in the latent space. In this evaluation, we consider the following representatives of this category: Ensemble GI [43], GrammarViz [120], LaserDBN [100], MultiHMM [78], PST [128], Series2Graph [16], TARZAN [71], and TSBitmap [144].

GrammarViz and its successor Ensemble GI discretize the subsequences to, then, infer hierarchical grammar rules; both algorithms consider hard to compress subsequences (low grammar rule coverage) as anomalous. Similarly, TSBitmap encodes discretized subsequences as bitmaps that preserve the frequency of the subsequences; the distances between bitmaps of a leading and a lagging window are then used as anomaly score. TARZAN also encodes the frequencies of discretized subsequences, but it uses suffix trees for both the training and test time series (semi-supervised); the difference between the expected frequency (from training) of a subsequence to the observed frequency is used as anomaly score. LaserDBN, PST, and MultiHMM construct probabilistic models and use the log-likelihood of subsequences as their anomaly score; while MultiHMM builds the model from a normal training time series (semi-supervised), LaserDBN and PST consider only the test time series. Series2Graph transforms subsequences of the test time series into a lower-dimensional space, from which the approach builds a directed cyclic graph. The graph's edges represent the transitions between groups of subsequences. The more often an edge is traversed by the time series, the higher its score gets. Thus, edges with low scores are more anomalous.

3.4 Distance Methods

Distance methods (symbol: ●) use specialized distance metrics to compare points or subsequences of a time series with each other. Anomalous subsequences are expected to have larger distances to other subsequences than subsequences with normal behavior. For the distance calculations, algorithms in this family may take either all other subsequences, only some nearest neighbors, or certain cluster centroids as distance reference points. Some approaches perform a mapping of the subsequences into a multidimensional space before computing the distances. Cluster-based distance methods cluster similar subsequences together and, then, compute the distances to dense areas. Most of the methods in this category create subsequences via a sliding window with a stride of 1 on the test time series. Distance-based methods usually do not require training data and are, therefore, unsupervised. Representatives of this category in our evaluation are: CBLOF [59], COF [130], DBStream [55], HOT SAX [70], Hybrid KNN [124], k-Means [151], KNN [110], LOF [22], NormA-SJ [15], PhaseSpace-SVM (PS-SVM) [85], SAND [17], SSA [155], STAMP [156], STOMP [164], Sub-LOF [22], VALMOD [82], and Left STAMPi [156].

Nearest neighbor methods determine anomaly scores by computing the distance of points (KNN, COF, LOF) or subsequences (STAMP, STOMP, VALMOD, Left STAMPi, Sub-LOF, HOT SAX, Hybrid KNN) to their nearest neighbors. Infrequent, uncommon subsequences have large distances to their neighbors and are, therefore, scored as anomalous. A particular subclass of this group (STAMP, STOMP, VALMOD, Left STAMPi) efficiently computes the *matrix profile*, which records the distance of each subsequence to its nearest non-self neighbor [156, 164]. DBStream and k-Means cluster the subsequences and then use the distances between subsequences and their corresponding cluster centroids as anomaly scores. Similarly, CBLOF uses multidimensional points instead of subsequences for the clustering. NormA-SJ, SAND, and SSA build a reference model of normal behavior to which subsequences are compared to. The distance between a subsequence and the reference model is used as an anomaly score. PS-SVM fits a one-class SVM to a transformed representation of the subsequences and uses the inverse distance to the decision boundary as the anomaly score. The only semi-supervised approach in this family is Hybrid KNN; it requires normal training data without anomalies to build a model of normality to which new subsequences are compared to.

3.5 Distribution Methods

Distribution methods (symbol: ▲) estimate the distribution of the data or fit a distribution model to the data. The distributions are calculated either over data points or subsequences obtained via windowing. Although similarity of points and subsequences can be a factor for the distribution fit (very similar patterns are counted as equal), abnormality is judged by frequency rather than distance in this algorithm family. The anomaly scores are usually measured using probabilities, likelihoods, or distances of the points or subsequences w. r. t. the prior calculated distributions. In general, this is an unsupervised approach, as anomalies can be found in the extremes/tails of the distributions. In the semi-supervised case, the distribution is estimated over a training time series that contains only normal behavior, while the points or subsequences of the test time series are then checked against the previously learned distribution. Representatives of distribution methods in our evaluation are: COPOD [80], DWT-MLEAD [134], Fast-MCD [115], HBOS [47], Normalizing Flows (NF) [116], S-H-ESD [62], DSPOT [122], and Sub-Fast-MCD [115].

DWT-MLEAD, Fast-MCD, and Sub-Fast-MCD estimate a Gaussian distribution over the time series. Afterwards, the anomaly of points or subsequences is measured by their distance to the mean of the distribution. The difference in these approaches is that DWT-MLEAD uses a discrete wavelet transform (DWT) as preprocessing step and the log-likelihood of subsequences as the anomaly score; Fast-MCD and Sub-Fast-MCD, on the other hand, calculate the score as the Mahalanobis distance between the points (Fast-MCD) or subsequences (Sub-Fast-MCD) and the estimated Gaussian distribution of a normal training time series (semi-supervised). Another algorithm, HBOS, estimates a generic probability distribution of subsequences with the help of histograms; then, it uses the inverse density of the subsequences' histogram bins as anomaly score. The COPOD algorithm builds an empirical multidimensional cumulative distribution function (copula) to estimate the tail probability of points; these tail probabilities then translate into the anomaly scores. S-H-ESD is a point anomaly detector that, first, performs STL time series decomposition and then applies a Grubbs test, which assumes a Gaussian distribution, on the residuals to mark outlier points. Another point anomaly detector, DSPOT, estimates a generalized Pareto distribution of the time series' extreme values and applies a threshold on the distribution tails to flag anomalous points. Finally, NF is a supervised sequence anomaly detection method that transforms an arbitrary prior distribution to a Gaussian distribution using a neural network; it then marks subsequences falling into the distribution's tails as anomalous.

3.6 Isolation Tree Methods

Isolation tree methods (symbol: ✚) build an ensemble of random trees that partition the samples (points or subsequences) of the test time series. For the tree construction, the methods recursively select random features and random split values as tree nodes to eventually isolate the samples in the tree leaves. The number of splits required to isolate a sample is a measure described by the average path length over all random trees in the ensemble. Because anomalous samples are easier to separate than normal samples, they are on average closer to the tree root and have noticeably shorter paths. For this reason, path lengths are characteristic for the normality of samples and, hence, their reciprocal value translate into anomaly scores. Representatives of this category in our evaluation are: Extended Isolation Forest (EIF) [58], Hybrid Isolation Fores (HIF) [91], Isolation Forest - Local Outlier Factor (IF-LOF) [36], Isolation Forest (iForest) [83], and Sub-IF [83].

The general isolation tree method described above, which is also the method that all algorithms in this family are based upon, is the iForest algorithm. Supervised variants of this approach are EIF and HIF. The Sub-IF algorithm is an extension for subsequence anomaly detection, i. e., an algorithm that can process sequences instead of points, and IF-LOF is a combination of iForest [83] and LOF [22].

4 EXPERIMENTAL EVALUATION

For our experimental evaluation, we executed 71 algorithms (Section 4.1.1) on 976 real-world and synthetic datasets (Section 4.1.2). ~~We optimized the parameters of all algorithms globally over a set of well-labeled synthetic datasets so that the experiments use the possibly best configuration for each algorithm (Section 4.1.3).~~ In this section, we finally present an exhaustive evaluation on the algorithms' quality (Section 4.2) and runtime behavior (Section 4.3).

We emphasize that this evaluation compares time series anomaly detection algorithms by their properly published implementations and draws conclusions about the different concepts and families from the results. An exhaustive evaluation of all concepts (matrix profiles vs. auto-encoders vs. trees vs. …) in all possible implementations is technically impossible. Hence, if an algorithm exhibits a bad performance in our evaluation, this does not necessarily mean that its concept is bad. However, a bad performance shows that, in practice, achieving a good result with a particular approach is hard.

4.1 Environment and Setup

For this study, we developed an evaluation tool called TimeEval[1] that manages the executions of the numerous evaluation tasks. An evaluation task is one algorithm analyzing one dataset. TimeEval is a Python 3.8 program that automatically schedules, executes, and analyzes a pre-defined set of evaluation tasks on a compute-cluster. It measures and calculates all the metrics, status codes, and execution times that we report in this section. Because TimeEval runs the algorithms in separate Docker containers, the algorithms are not bound to a specific programming language or version and TimeEval can distribute the evaluation tasks over a computer cluster. We ran our experiments on a cluster of 14 servers equipped with Intel Xeon E5-2630 v4 CPUs (10 cores at 2.2 GHz). Every Docker container, and thus every algorithm, is restricted to one CPU core and 3 *GB* of memory. Hence, every node can simultaneously run 10 tasks. The entire cluster can perform 140 evaluation tasks in parallel. We do not use GPUs – neither for training nor testing.

Table 2: Data collections used for experimental evaluation with their dimensionality (*uni*variate and/or *multi*variate).

Collection Name	Origin	Dim.	Avg. Length	Avg. Cont.	# Datasets total	used
Callt2 [41, 66]	real	multi	5,040	4.1%	1	1
Daphnet [7, 41]	real	multi	32,594	13.2%	35	3
Dodgers [41, 66]	real	uni	50,400	11.1%	1	0
Exathlon [67]	real	multi	47,530	18.3%	39	2
GHL [42]	synthetic	multi	200,001	0.4%	48	0
Genesis [139]	real	multi	16,220	0.3%	1	1
GutenTAG	synthetic	multi/uni	10,000	2.4%	193	187
IOPS [112]	real	uni	100,649	1.8%	29	4
KDD-TSAD [69, 147]	synthetic	uni	77,415	0.6%	250	249
Kitsune [92]	real	multi	2,335,288	17.0%	9	0
LTDB [45, 46]	real	multi	9,706,422	14.1%	7	0
MGAB [132]	synthetic	uni	100,000	0.2%	10	10
MITDB [45, 93]	real	multi	650,000	12.5%	48	4
Metro [41, 61]	real	multi	48,204	0.1%	1	0
NAB [3]	real/synthetic	uni	6,302	8.9%	58	56
NASA-MSL [64]	real	uni	2,730	12.0%	27	16
NASA-SMAP [64]	real	uni	8,070	12.4%	54	35
OPPORTUNITY [41, 113]	real	multi	36,224	3.4%	24	0
Occupancy [24, 41]	real	multi	6,208	28.7%	2	0
SMD [125]	real	multi	25,300	4.2%	28	23
SSA [104, 155]	real	uni	27,038	22.5%	23	0
SVDB [45, 50, 51]	real	multi	230,400	13.6%	78	16
WebscopeS5 [75]	real/synthetic	uni	1,561	0.7%	367	360

All experiments define a time limit for training and execution times of 2 *h*, respectively, and a memory limit of 3 *GB*. We employ early-stopping and model-checkpointing for (semi-)supervised algorithms that need to iterate the training time series multiple times, i. e. once per epoch. Every algorithm gets up to 2 hours to fit to the training data, after which the best-so-far-model is used for the anomaly detection on the test data. If an algorithm cannot finish a single epoch within the time limit or takes more than 2 hours to detect the anomalies on the test time series, we consider it timed-out. Fast-MCD, RForest, XGBoosting, HealthESN, HIF, LaserDBN, MultiHMM, RBForest, RobustPCA, Sub-Fast-MCD, and TARZAN are (semi-)supervised and have no iterative training approach. Thus, the model-checkpointing strategy does not apply to those algorithms.

4.1.1 Algorithms. Of the 71 algorithms that we consider in this evaluation, we collected 22 author implementations (*original*), found 10 reliable community implementations (*community*), and re-implemented 39 algorithms if no implementation was obtainable (*own*) (cf. Table 1). A special TimeEval wrapper adds required interfaces and transforms the algorithm outputs into time series scorings. From the 71 implemented algorithms, five algorithms could not process a single dataset due to hitting memory or time limits [43, 52, 86, 159, 161]. Another five algorithms did not produce a single reasonable result on any dataset [106, 115, 115, 117, 117]. Because we could not solve the issues (if there are any) from the provided material, we excluded these 10 algorithms from our analysis.

4.1.2 Datasets. For this study, we collected 1,354 datasets from 24 collections. Table 2 lists the collections, their cardinality, and the dimensionality of their time series. Due to the unreliability of the anomaly labels in the collected datasets [147], we developed the Good Time Series Anomaly Generator (GutenTAG) and generated another 194 time series with a large variety of well-labeled anomalies. The generated time series are of five base-types (Sine, ECG,

[1] The source code, data, and other artifacts have been made available at https://hpi.de/naumann/s/time-series-anomaly-detection-evaluation.

Random Walk, Cylinder Bell Funnel, and Polynomial) with different lengths, variances, amplitudes, frequencies, and dimensions. The selection of injected anomalies covers nine different types (amplitude, extremum, frequency, mean, pattern, pattern shift, platform, trend, and variance), which are particularly interesting for the analysis of certain detection capabilities. Furthermore, the well-defined and reliable labels enable a stable parameterization of the algorithms.

From the 1,354 collected datasets, we removed all time series that (a) have no anomalies because there is nothing to be found; (b) have a *contamination* > 0.1 because having more than 10% anomalous points objects the assumption that anomalies are rare (primarily SSA, Occupancy, and Dodgers); (c) could not be processed by at least 50% of the algorithms (usually due to their size and our strict memory and time limits) because this distorts our analytical results (primarily Exathlon, IOPS, Kitsune, and LTDB); and (d) could not be processed with an AUC-ROC score of at least 0.8 by a single algorithm because this indicates a bad label quality (primarily MITDB, SVDB, and Daphnet). The NAB collection contains a few datasets that do not conform to some of our filters, but we included them anyway to see how the algorithms react to these datasets. We also included four randomly selected datasets from Exathlon and IOPS despite their large size, to capture at least a few large time series in the evaluation as food for very efficient approaches. After the filtering, 976 time series datasets remained (cf. Table 2).

4.1.3 Parameterization. To compare the algorithms in their best possible configurations, we ran a systematic (hyper-)parameter tuning process. Because a full parameter grid search on 192 parameters, 976 datasets, and three evaluation metrics is clearly infeasible (and unrealistic to be run in practice), we assume parameter independence and optimize each parameter separately for best average AUC-ROC score on all synthetic GutenTAG datasets. The best parameter values (see availability page) are then used for the entire evaluation. More specifically, the tuning was executed as follows:

(1) Parameter initialization: We first derive initial settings from default values provided with the algorithms' publications or reference implementations; if no settings are provided, we set reasonable values manually by performing brief tests on a few datasets.

(2) Parameter classification: We classify all individual 192 parameters into four groups: *fixed* (99), *dependent* (12), *shared* (11), and *optimize* (70). *Fixed* parameters are default parameters without obvious alternatives, parameters that impact only the runtime performance, or parameters with very explicit default suggestions. *Dependent* parameters are ones whose best values depend on time series properties so that they cannot be defined globally. For example, the optimal `window-size` depends on the period size or dominant frequency of the processed dataset. Hence, for dependent parameters, we optimize the heuristic that is used to dynamically calculate them from time series metadata. *Shared* parameters are parameters that occur in multiple algorithms with the exact same function. These parameters can be optimized jointly for all algorithms, in which they appear, which saves time and increases fairness. *Optimize* parameters are all remaining parameters that need to be optimized per algorithm.

(3) Parameter limitation: We define a search space (based on default values, human expertise, and ad-hoc tests) for each non-fixed parameter, which is a specific range of values this parameter (or heuristic) can take. For numeric values, we then draw 5 equidistant samples (including the default value) from this range; for categorical and boolean values, we test all possible settings.

(4) Automatic optimization: We set up TimeEval to test every parameter in every previously determined setting. For algorithms that take exceptionally long on certain datasets, we reduced the size of these datasets for the parameter search. Settings with the best average AUC-ROC scores are then promoted to default parameters.

4.2 Quality of Results

In this section, we first analyze the effectiveness of all implemented algorithms on all datasets (Section 4.2.1). We then investigate the reliability of our results and implementations (Section 4.2.2). Finally, we evaluate the overall performance for specific anomaly types and dataset characteristics (Section 4.2.3).

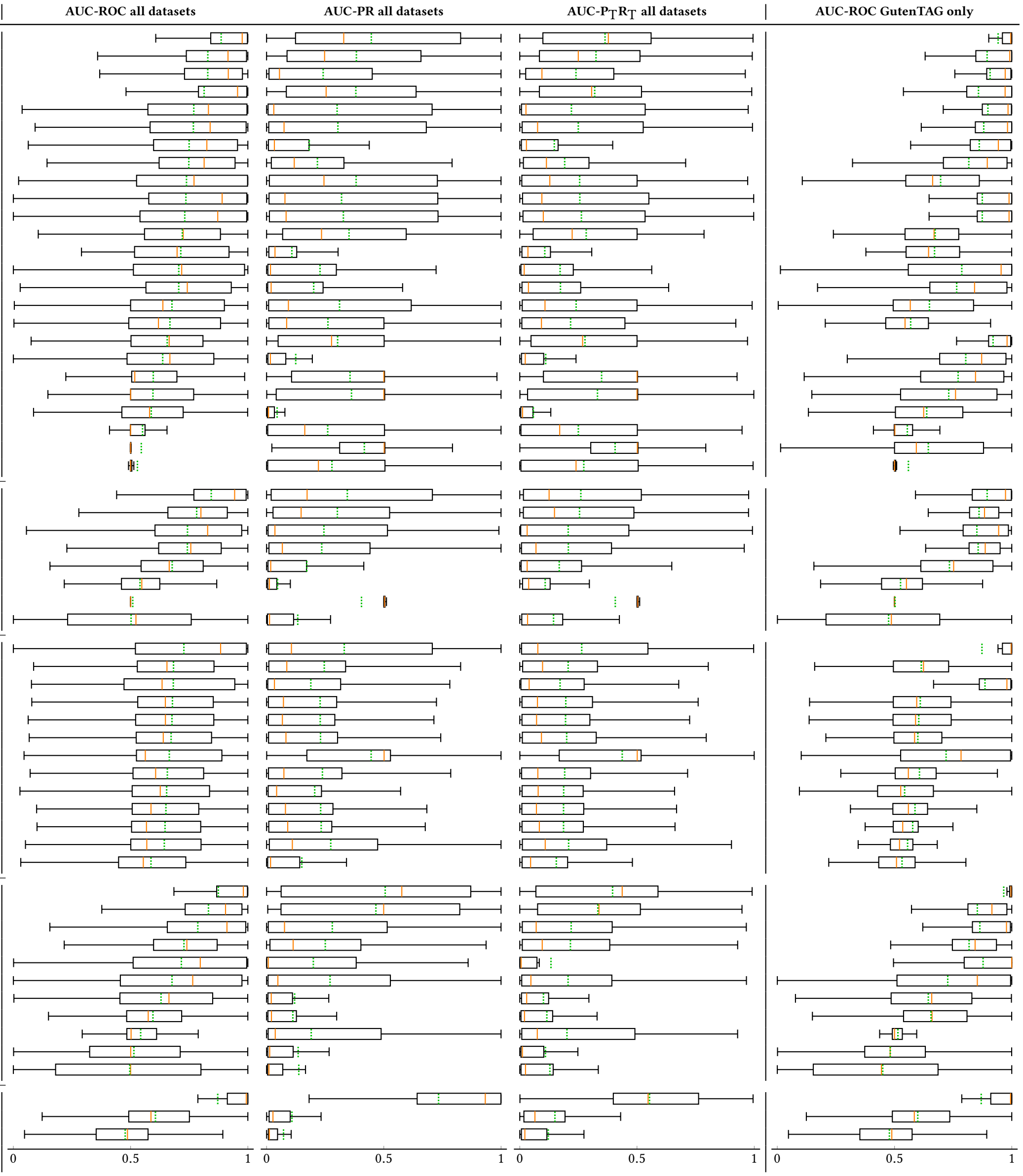
4.2.1 AUC-ROC, AUC-PR and AUC-$P_T R_T$. Table 3 provides a comprehensive qualitative performance overview of all our experimental results. The table lists all 61 algorithms grouped by their input dimensionality and learning type. The table columns *TL*, *OOM*, and *ERR* show the percentage of timeout, out-of-memory, and other errors during the experiments respectively. Within their groups, the algorithms are sorted descending by their mean AUC-ROC score over all datasets. The remaining columns to the right contain a box plot for each algorithm (rows) and specific quality metric (columns). We report the AUC-ROC, AUC-PR, and AUC-$P_T R_T$ scores for all datasets and, additionally, the AUC-ROC score aggregated over only our synthetically generated GutenTAG datasets.

39 of the 61 algorithms detect anomalies in an unsupervised way. 19 algorithms are semi-supervised and, thus, have to train on data without anomalies beforehand. (**RI 1**) Despite supervised algorithms using additional information during training (labels for normal *and* anomalous points), they do not achieve superior results compared to semi-supervised or even unsupervised approaches. One exception is the NF algorithm, which achieved excellent detection rates, albeit only on 34% of the datasets; for the majority of datasets, it either ran into timeouts or errors, which makes the reported results not very reliable (cf. Section 4.2.2). In addition to their weak performance (in this experiment), supervised algorithms are rather unpopular in practice (only 7 out of 158 methods are supervised) most likely because anomalies are usually not known beforehand in real-world scenarios. However, since supervised detection methods translate easily to multivariate data, we have not found any supervised algorithm restricted to only univariate data.

The comparison of the scores of multivariate and univariate algorithms shows that on average, univariate algorithms perform slightly better. For this comparison, we need to consider that multivariate algorithms were run on *all* time series while univariate algorithms processed only univariate time series. One might conclude that multivariate time series are harder to analyze, but the multivariate algorithms performed 6% worse on the univariate data w. r. t. AUC-ROC than on the multivariate data in our evaluation. We therefore conclude that (**RI 2**) there is no one-size-fits-all solution in the set of currently available algorithms: A multivariate algorithm is necessary to detect multivariate anomalies (e. g., anomalies in the correlation of series), but a univariate algorithm is preferable for univariate data.

Table 3: Qualitative performance overview for all 61 algorithms with box plots (mean in green and median in orange) showing the score distribution for the metrics AUC-ROC, AUC-PR, and AUC-$P_T R_T$.

Dim.	Learn.	Algorithm	TL	OOM	ERR
UNIVARIATE	UNSUPERVISED	● Sub-LOF [22]	2 %	0 %	0 %
		▼ GrammarViz [120]	3 %	0 %	0 %
		▲ DWT-MLEAD [134]	0 %	0 %	0 %
		● VALMOD [82]	1 %	9 %	11 %
		● SAND [17]	5 %	1 %	22 %
		● Left STAMPi [156]	2 %	0 %	1 %
		▼ Series2Graph [16]	0 %	0 %	5 %
		★ ARIMA [65]	7 %	0 %	0 %
		■ PCI [157]	0 %	0 %	0 %
		● STOMP [164]	2 %	0 %	0 %
		● STAMP [156]	4 %	0 %	0 %
		★ Triple ES [1]	15 %	0 %	9 %
		★ NumentaHTM [3]	0 %	0 %	0 %
		● NormA-SJ [15]	10 %	1 %	3 %
		✚ Sub-IF [83]	0 %	0 %	0 %
		★ MedianMethod [10]	0 %	0 %	0 %
		■ SR [112]	0 %	0 %	0 %
		● PS-SVM [85]	12 %	0 %	0 %
		✚ PST [128]	0 %	4 %	0 %
		● SSA [155]	1 %	0 %	0 %
		● HOT SAX [70]	24 %	1 %	1 %
		▼ TSBitmap [144]	0 %	0 %	0 %
		▲ DSPOT [122]	6 %	0 %	0 %
		■ FFT [111]	0 %	0 %	0 %
		▲ S-H-ESD [62]	0 %	0 %	49 %
	SEMI-SUPERVISED	■ Donut [150]	1 %	1 %	2 %
		★ RForest [21]	12 %	0 %	0 %
		■ IE-CAE [44]	0 %	0 %	1 %
		★ XGBoosting [34]	0 %	0 %	0 %
		★ OceanWNN [143]	0 %	0 %	10 %
		■ Bagel [79]	19 %	0 %	2 %
		■ SR-CNN [112]	22 %	0 %	1 %
		▼ TARZAN [71]	0 %	0 %	18 %
MULTIVARIATE	UNSUPERVISED	● k-Means [151]	0 %	4 %	1 %
		● KNN [110]	0 %	0 %	0 %
		★ Torsk [60]	7 %	0 %	0 %
		✚ EIF [58]	0 %	0 %	0 %
		✚ iForest [83]	0 %	0 %	0 %
		● HBOS [47]	0 %	0 %	0 %
		● DBStream [55]	0 %	2 %	78 %
		● CBLOF [59]	0 %	0 %	0 %
		▲ COPOD [80]	0 %	0 %	0 %
		✚ IF-LOF [36]	0 %	0 %	3 %
		● LOF [22]	0 %	0 %	0 %
		● COF [130]	0 %	24 %	0 %
		■ PCC [121]	0 %	0 %	0 %
	SEMI-SUPERVISED	★ LSTM-AD [89]	14 %	50 %	3 %
		★ HealthESN [32]	66 %	0 %	0 %
		★ Telemanom [64]	0 %	0 %	0 %
		★ RBForest [165]	6 %	5 %	0 %
		■ EncDec-AD [88]	55 %	26 %	3 %
		★ DeepAnT [94]	0 %	0 %	5 %
		■ OmniAnomaly [125]	0 %	4 %	2 %
		▼ LaserDBN [100]	4 %	0 %	7 %
		■ RobustPCA [101]	0 %	0 %	0 %
		■ TAnoGan [8]	65 %	0 %	1 %
		● Hybrid KNN [124]	0 %	1 %	1 %
	SUPER	▲ NF [116]	60 %	6 %	0 %
		✚ HIF [91]	1 %	0 %	0 %
		▼ MultiHMM [78]	0 %	0 %	52 %



Some algorithms take exceptionally long for training and/or execution or consume a lot of memory, which caused them to exceed our resource limits: NF, HealthESN, EncDec-AD, and TAnoGan, for example, finished less than 50% of the datasets within the time limit and COF, LSTM-AD, and EncDec-AD ran out of memory in more than 20% of the experiments. These timeout and memory errors often occur with multivariate algorithms due to the higher complexity of multivariate data and algorithms. Algorithm implementations with high error rates, such as S-H-ESD, DBStream, and MultiHMM, often failed because they could not deal with certain input characteristics (e. g., value plateaus) or made assumptions that do not generalize; errors also indicate technical flaws that we could not resolve. A high error rate does, in general, not reflect on the performance of an algorithm, as we see unstable implementations with both good (e. g., NF) and bad (e. g., S-H-ESD) scores. **(RI 3)** The relatively high overall error susceptibility despite our strong investment into each implementation shows that every practical algorithm deployment needs careful testing; only few implementations, such as DWT-MLEAD, KNN, and ~~Sub-LOF, actually appear to be both robust and effective~~.

When comparing the results of different quality metrics, we observe that the rankings for the AUC-ROC and AUC-PR scores

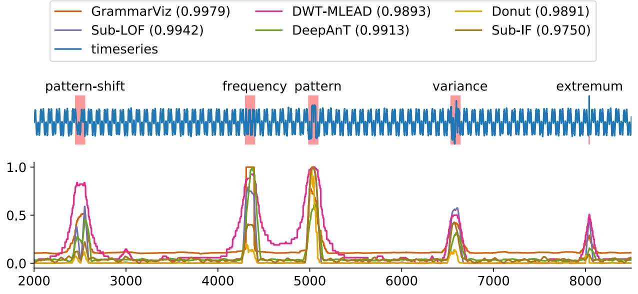


Figure 3: Scorings of GrammarViz (encoding), Sub-LOF (distance), DWT-MLEAD (distribution), DeepAnT (forecasting), Donut (reconstruction), and Sub-IF (trees) on a GutenTAG *sinus* series with five anomalies; AUC-ROC in parentheses.

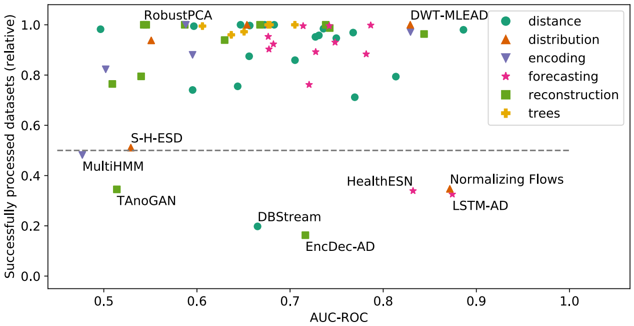


Figure 4: Reliability of reported results.

clearly deviate due to their different scoring focus (cf. Section 2). The reason is that some algorithms mark the start and the end of an anomalous section, while others mark the entire section; most algorithms also mark the transition into an anomaly as anomalous, while, i. a., forecasting methods tend to mark anomalies not before they actually started. For example, DeepAnT is much stronger in AUC-ROC than in AUC-PR, while DBStream is the strongest approach in AUC-PR but only mediocre in AUC-ROC. (**RI 4**) The most relevant scoring metric, therefore, depends on the use case and what the results are expected to indicate. We provide the scores for all three metrics in Table 3 for reference and report only AUC-ROC scores in all further charts.

The results in Table 3 further show higher average AUC-ROC scores for experiments on only GutenTAG datasets than on all datasets; in particular, the AUC-ROC score variances are lower on average. This is partially because the algorithms have been optimized for these datasets, which demonstrates the need for parameter tuning. However, the clarity of the improvements (despite varying parameter counts) also shows that real-world datasets are either harder or their labeling quality is worse. We therefore follow the recommendation of [147] and also provide results on controlled data for performance comparison. Contrary to our general observation, a few outlier-detection algorithms, such as MedianMethod, actually perform better on all datasets, which is because the fraction of extreme-value anomalies is higher in the real-world data than in the generated data and they excel in this discipline. The results for supervised algorithms are the same for both aggregation types because the collected data does not include time series with proper labels for supervised training.

The scorings in Table 3 let us draw the conclusion that k-Means, which leverages simple subsequence time series clustering, is a very effective multivariate approach performing similarly well as other representatives of the distance family. This is surprising because previous research suggests that subsequence time series clustering produces meaningless cluster centers (usually sinusoids) [81]. The authors of [81] claim that motifs, similar to those used in NormA-SJ, capture true cluster centroids more accurately, but our experiments cannot confirm that this is actually beneficial for anomaly detection.

Figure 3 shows six AUC-ROC scorings for a time series with different types of anomalies. From each anomaly detection family, we selected one interesting representative to demonstrate certain algorithmic strengths: GrammarViz (encoding) is particularly strong in frequency and pattern anomaly detection; Sub-LOF (distance) marks pattern shifts particularly precisely and works well in general; DWT-MLEAD (distribution) also finds all anomalies very well, but it produces high scores for noticeably wide areas around the anomalous sections; DeepAnT (forecasting) detects pattern shifts and frequencies particularly well and, as a forecasting approach, its scores always peak a bit later than all other scores; Donut (reconstruction) is clearly very sensitive to patterns but insensitive to magnitudes, which is why the pattern anomaly dominates its scores; Sub-IF (trees) is particularly sensitive to pattern anomalies and struggles a bit with point anomalies and pattern shifts.

(**RI 5**) Our experimental results on the different datasets show that, overall, every anomaly detection family can be effective and there is no clear winner. Furthermore, no single algorithm achieved perfect scores leaving much room for future work.

4.2.2 Reliability of Results.

Figure 4 relates the algorithms' average AUC-ROC scores to their percentage of successfully processed datasets. Hence, the y-coordinate of a point indicates the reliability of the point's AUC-ROC score. The algorithms are color-coded by their family and the most relevant points are labeled.

Most algorithms (87%) have successfully processed more than 70% of the datasets, and many algorithms (35%) even processed more than 99% of the datasets. The large majority of reported quality measurements is, therefore, reliable. For example, DWT-MLEAD has an average AUC-ROC score of 0.83 with a reliability of 100% and will most likely perform well also on other datasets. On the contrary, RobustPCA has an average AUC-ROC score of only 0.54 with a reliability of 100% and will, therefore, most likely perform badly also on other datasets.

Only 8 algorithms have a reliability lower than 52%, among which MultiHMM, S-H-ESD, and TAnoGan performed poorly, DBStream and EncDec-AD performed mediocre, and HealthESN, NF and LSTM-AD performed great. The qualitative characterizations are, therefore, not particularly representative, but demonstrate significant setup and runtime difficulties: MultiHMM, S-H-ESD, and DBStream encountered many internal errors that we could not fix; NF, HealthESN, EncDec-AD, and TAnoGan struggled with the 4 *h* time limit; and LSTM-AD often exceeded the 3 *GB* memory limit.

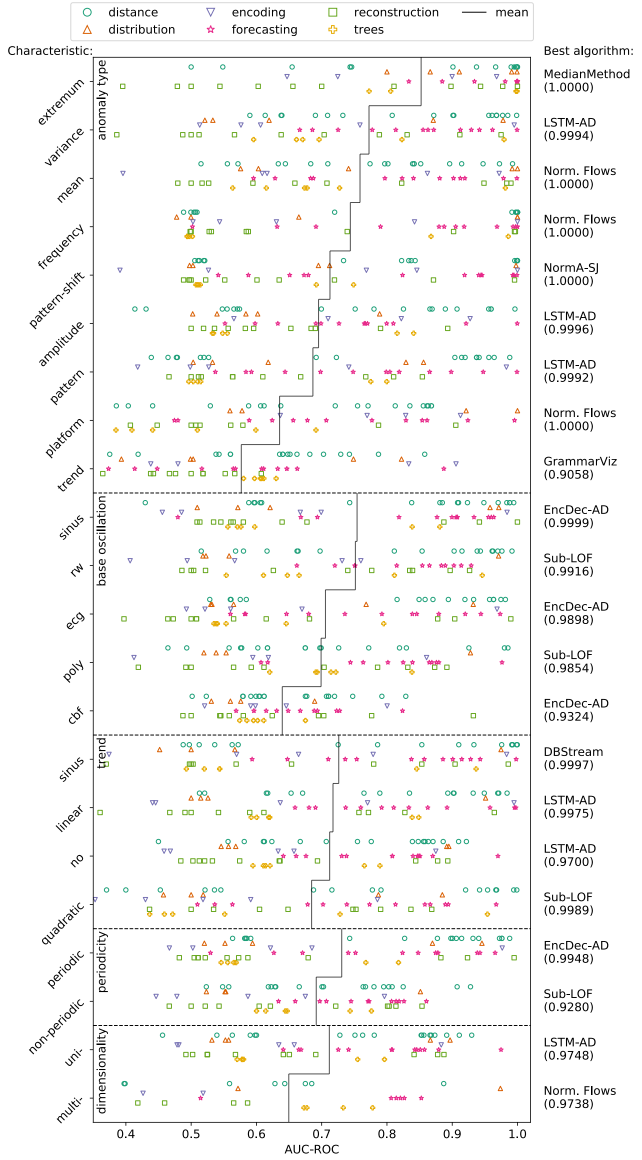


Figure 5: Quality results grouped by dataset characteristics (anomaly type, base oscillation, trend, periodicity) with emphasis on algorithm families.

4.2.3 *Anomaly Types and Dataset Characteristic.*

Our data generation tool GutenTAG[2] generates time series with very specific characteristics (anomaly type, base oscillation, trend, periodicity, and dimensionality), which we can use to conduct insightful analyses. Figure 5 shows the average AUC-ROC scores of all 61 algorithms grouped by these characteristics: Every point in the plot represents an algorithm and every row a specific characteristic. For better readability, every row is subdivided into six color-coded sub-rows – one for each method family. The black vertical marker line in each row marks the mean AUC-ROC score of all algorithms for the particular characteristic and, hence, the overall performance for a specific characteristic (in the generated time series). We annotate the best performing algorithm per data characteristic (highest average AUC-ROC score; smallest standard deviation as tiebreaker).

Considering the anomaly types, (**RI 6**) an *extremum* seems to be the easiest anomaly (avg. AUC-ROC score above 0.8) and (**RI 7**) a *trend* the hardest anomaly (avg. AUC-ROC score below 0.6). For *extremum* anomalies, we find that tree-based algorithms and point anomaly detectors work exceptionally well; for *trend* anomalies, only few encoding, distance and forecasting approaches delivered acceptable results. (**RI 8**) *Frequency* and *pattern-shift* anomalies clearly separate the algorithms in ones that can (mostly distance and forecasting) and ones that cannot (mostly reconstruction and tree) detect them.

The base oscillation groups show that anomalies in periodic sine waves are the easiest to detect, while the same types of anomalies are the hardest to detect in chaotic Cylinder Bell Funnel (CBF) base oscillations. (**RI 9**) Due to the lack of structure in chaotic time series, most algorithms struggle to find a suitable representation for normal behavior or cannot distinguish between normal and abnormal subsequences at all. The algorithms reached a suspiciously high average AUC-ROC score of 0.75 on the non-periodic Random Walk (RW) time series because these series do not contain the relatively hard-to-find *pattern* and *pattern-shift* anomalies – on non-periodic data, such anomalies cannot be represented. However, the comparison of *sinus* and *ecg* indicates that increasingly complex patterns also increase the difficulty for anomaly detection.

Figure 5 shows that trends do have an impact on individual algorithms (sometimes positive and sometimes negative), but the average performance of all algorithms is surprisingly robust against simple trends. Only the quadratic trend seems to complicate the detection a bit. For distance-based algorithms, trends are the slightest problem (if not an advantage).

(**RI 10**) Anomalies on periodic time series are easier to detect than on non-periodic time series. Even though the average values differ by only 0.03, more algorithms reached scores higher than 0.90 on periodic time series (16 algorithms) than on non-periodic time series (two algorithms). As mentioned earlier, periodic time series also contain more difficult anomalies so that their scores are disadvantaged in this comparison.

Our measurements show that anomaly detection on univariate time series is on average easier than on multivariate time series. A direct comparison yields a higher average AUC-ROC score of 0.06. This effect is particularly evident on our generated datasets because most of their anomalies are present only in a single channel, which makes them harder to distinguish from noise.

(**RI 11**) In summary, most reconstruction methods yielded rather bad AUC-ROC scores (∼ 0.5) and only some algorithms in this group, i. e., EncDec-AD and Donut, can detect anomalies well across all characteristics; forecasting and distance algorithms, on the contrary, yielded particularly good results and many of their representatives, such as DeepAnT and Sub-LOF, are amongst the best performing algorithms in almost every characteristic; finally, distance algorithms performed remarkably well on variance anomalies.

[2] For a more detailed description, please consider the GutenTAG documentation at https://github.com/HPI-Information-Systems/gutentag.

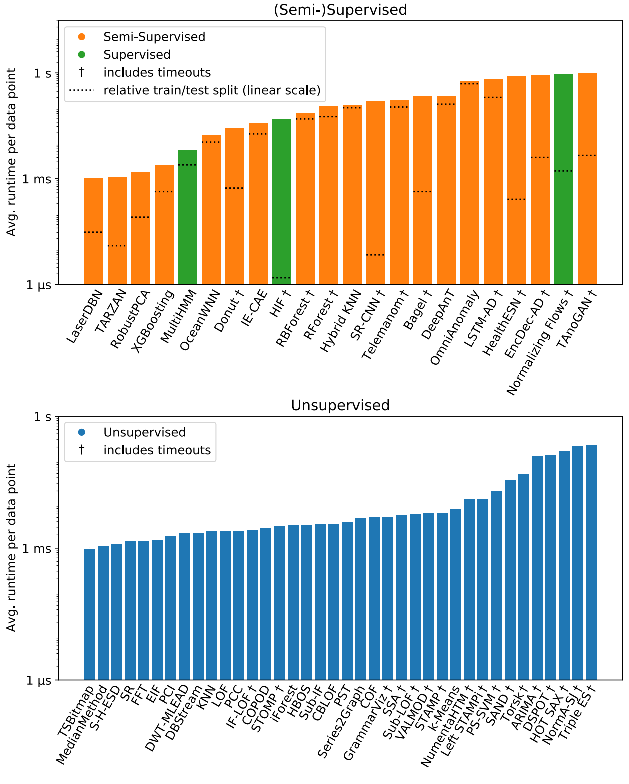


Figure 6: Average runtime per algorithm for one data point (logarithmically scaled).

4.3 Runtime and Memory

In a final experiment, we measured the runtimes for all algorithms on our univariate time series and normalized the runtimes by the lengths of the respective time series. Figure 6 shows the average measured runtime of every algorithm for one time series data point. The plotted times cover training and testing times, both with a time limit of 2 *h*. Consequently, (semi-)supervised algorithms can take up to 4 *h* per time series if their training does not converge before the limit and they time out during testing. Algorithm executions that exceed the time limit are considered with a runtime equal to the time limit. If this happens to an algorithm at least once, it is marked with a † to indicate that its true runtime could be longer than listed in the plot. Algorithm executions with errors are excluded from consideration because we could not measure any runtimes.

(**RI 12**) Most supervised and semi-supervised algorithms are among the slowest algorithms in our evaluation and need on average 255 *ms* for one data point, which is largely due to their long training times. This observation confirms similar conclusions made in related work [67]. There are, however, (semi-)supervised methods, such as XGBoosting, RobustPCA, TARZAN, and LaserDBN, that compete well (< 2.5 *ms* per data point) with the runtime performance of the unsupervised methods. We also find unsupervised algorithms, such as Triple ES, NormA-SJ, HOT SAX, DSPOT, and ARIMA, that took exceptionally long (> 125 *ms* per data point). Because every approach calculates/learns some model to solve the task, the performance is not necessarily tied to the learning type. (**RI 13**) Overall, neither the fastest algorithms nor the slowest algorithms delivered qualitatively good results (cf. Table 3). The algorithm with the clearly best cost/benefit ratio in our experiments is DWT-MLEAD with an average AUC-ROC score of 83% and an outstanding runtime of 2.2 *ms* per data point.

(**RI 14**) Only few implementations actually struggle with our 3 *GB* memory limit. LSTM-AD and EncDec-AD, for example, are deep learning methods that innately require more memory than other methods and, therefore, exceeded the limit in 50% and 26% of the experiments, respectively. The unsupervised COF algorithm also exceeded the limit in 24% of all experiments. It is comparatively memory inefficient but able to analyze multivariate data, while other unsupervised methods with lower memory footprints cannot.

5 DISCUSSION

In line with related work [67], we found that deep learning approaches are not (yet) competitive despite their higher processing effort on training data. We could also confirm that *"simple methods yield performance almost as good as more sophisticated methods"* [56]. Still, no single algorithm clearly performs best. We highlighted several algorithms with specific strengths, but the overall performance results call for further research in the following three areas:

Flexibility: No algorithm (or algorithm family) clearly dominates all other approaches and solves all anomaly detection setups. To advance the field of anomaly detection, we suggest further research on *holistic* and *hybrid* anomaly detection systems that combine existing strengths for the detection of more diverse anomalies in time series with arbitrary characteristics.

Reliability: Despite our best efforts, only very few algorithms could process all time series without errors and within common time and memory limits. We therefore emphasize the importance of further research on the *robustness* and *scalability* of time series anomaly detection algorithms.

Simplicity: Most anomaly detection algorithms of this study were remarkably sensitive to their parameter settings and required on average seven settings. What makes this problem worse is that most practical use cases do not have training data for algorithm configuration. For this reason, further research on *auto-configuring* and *self-tuning* algorithms is very much needed.

ACKNOWLEDGMENTS

We thank all members of our community that published or provided their data and source code, without whom this work would not have been possible. We thank very much Grit Fessel, Ulrike Herwig, Siddeshkanth Logonathan, David Matuschek, Rohan Sawahn, and Richard Schulz for their support in the implementation of individual algorithms and John Paparrizos, Paul Boniol, Themis Palpanas, and Felix Naumann for their valuable input to our research project.

The work was funded by the German government as part of the LuFo VI call I program (Luftfahrtforschungsprogramm) under the grant number 20D1915. The management of Rolls-Royce Deutschland Ltd. & Co. KG is gratefully acknowledged for supporting the work and permitting the presentation of results.

REFERENCES

[1] Adam Aboode. 2018. Anomaly Detection in Time Series Data Based on Holt-Winters Method. Master Thesis. KTH Royal Institute of Technology.

[2] Charu C. Aggarwal. 2017. *Outlier Analysis*. Springer International Publishing. isbn: 978-3-319-47578-3. doi: 10.1007/978-3-319-47578-3.

[3] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. 2017. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262, 134–147. doi: 10.1016/j.neucom.2017.04.070.

[4] Sardar Ansari, Negar Farzaneh, Marlena Duda, Kelsey Horan, Hedvig B. Andersson, Zachary D. Goldberger, Brahmajee K. Nallamothu, and Kayvan Najarian. 2017. A review of automated methods for detection of myocardial ischemia and infarction using electrocardiogram and electronic health records. *IEEE Reviews in Biomedical Engineering*, 10, 264–298. doi: 10.1109/RBME.2017.2757953.

[5] Jérôme Antoni and Pietro Borghesani. 2019. A statistical methodology for the design of condition indicators. *Mechanical Systems and Signal Processing*, 114, 290–327. doi: 10.1016/j.ymssp.2018.05.012.

[6] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. 1996. A linear method for deviation detection in large databases. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 164–169.

[7] M. Bachlin, M. Plotnik, D. Roggen, I. Maidan, J.M. Hausdorff, N. Giladi, and G. Troster. 2010. Wearable Assistant for Parkinson's Disease Patients With the Freezing of Gait Symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14, 2, 436–446. doi: 10.1109/TITB.2009.2036165.

[8] Md Abul Bashar and Richi Nayak. 2020. TAnoGAN: Time Series Anomaly Detection with Generative Adversarial Networks. arXiv: 2008.09567 [cs, stat]. Retrieved 09/14/2020 from http://arxiv.org/abs/2008.09567.

[9] Luis Basora, Xavier Olive, and Thomas Dubot. 2019. Recent Advances in Anomaly Detection Methods Applied to Aviation. *Aerospace*, 6, 11, 117. doi: 10.3390/aerospace6110117.

[10] Sabyasachi Basu and Martin Meckesheimer. 2007. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11, 2, 137–154. doi: 10.1007/s10115-006-0026-6.

[11] Arpita Bhargava and A.S. Raghuvanshi. 2013. Anomaly Detection in Wireless Sensor Networks Using S-Transform in Combination with SVM. In *Proceedings of the International Conference on Computational Intelligence and Communication Networks (CICN)*, 111–116. doi: 10.1109/CICN.2013.34.

[12] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. 2020. A review on outlier/anomaly detection in time series data. arXiv: 2002.04236 [cs, stat]. Retrieved 07/20/2020 from http://arxiv.org/abs/2002.04236.

[13] Kristof Böhmer and Stefanie Rinderle-Ma. 2016. Multi-perspective Anomaly Detection in Business Process Execution Events. In *Proceedings of the Confederated International Conference "On the Move to Meaningful Internet Systems" (OTM)*, 80–98. doi: 10.1007/978-3-319-48472-3_5.

[14] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. 2020. Automated Anomaly Detection in Large Sequences. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 1834–1837. doi: 10.1109/ICDE48307.2020.00182.

[15] Paul Boniol, Michele Linardi, Federico Roncallo, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. 2021. Unsupervised and Scalable Subsequence Anomaly Detection in Large Data Series. *The VLDB Journal*. doi: 10.1007/s00778-021-00655-8.

[16] Paul Boniol and Themis Palpanas. 2020. Series2Graph: graph-based subsequence anomaly detection for time series. *Proceedings of the VLDB Endowment (PVLDB)*, 13, 11, 14. doi: 10.14778/3407790.3407792.

[17] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. 2021. SAND: Streaming Subsequence Anomaly Detection. *Proceedings of the VLDB Endowment (PVLDB)*, 14, 10, 1717–1729. doi: 10.14778/3467861.3467863.

[18] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. 2015. *Time Series Analysis: Forecasting and Control*. (5th edition). John Wiley & Sons. isbn: 978-1-118-67502-1.

[19] Andrew P Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30, 7, 1145–1159.

[20] Mohammad Braei and Sebastian Wagner. 2020. Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. Version 1. arXiv: 2004.00433 [cs, stat]. Retrieved 10/27/2020 from http://arxiv.org/abs/2004.00433.

[21] Leo Breiman. 2001. Random forests. *Machine Learning*, 45, 1, 5–32. doi: 10.1023/A:1010933404324.

[22] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 93–104. doi: 10.1145/342009.335388.

[23] S. Budalakoti, S. Budalakoti, A.N. Srivastava, M.E. Otey, and M.E. Otey. 2009. Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39, 1, 101–113. doi: 10.1109/TSMCC.2008.2007248.

[24] Luis M. Candanedo and Véronique Feldheim. 2016. Accurate occupancy detection of an office room from light, temperature, humidity and CO 2 measurements using statistical learning models. *Energy and Buildings*, 112, 28–39. doi: 10.1016/j.enbuild.2015.11.071.

[25] Kevin M. Carter and William W. Streilein. 2012. Probabilistic reasoning for streaming anomaly detection. In *Proceedings of the Workshop on Statistical Signal Processing (SSP)*, 377–380. doi: 10.1109/SSP.2012.6319708.

[26] Soumen Chakrabarti, Sunita Sarawagi, and Byron Dom. 1998. Mining Surprising Patterns Using Temporal Description Length. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 606–617.

[27] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep Learning for Anomaly Detection: A Survey. arXiv: 1901.03407 [cs, stat]. Retrieved 09/14/2020 from http://arxiv.org/abs/1901.03407.

[28] V. Chandola, A. Banerjee, and V. Kumar. 2012. Anomaly Detection for Discrete Sequences: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 24, 5, 823–839. DOI: 10.1109/TKDE.2010.235.

[29] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys*, 41, 3, 1–58. DOI: 10.1145/1541880.1541882.

[30] Marcus Chang, Andreas Terzis, and Philippe Bonnet. 2009. Mote-Based Online Anomaly Detection Using Echo State Networks. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOOS)*, 72–86. DOI: 10.1007/978-3-642-02085-8_6.

[31] S. Chauhan and L. Vig. 2015. Anomaly detection in ECG time signals via deep long short-term memory networks. In *Proceedings of the International Conference on Data Science and Advanced Analytics (DSAA)*, 1–7. DOI: 10.1109/DSAA.2015.7344872.

[32] Qing Chen, Anguo Zhang, Tingwen Huang, Qianping He, and Yongduan Song. 2020. Imbalanced dataset-based echo state networks for anomaly detection. *Neural Computing and Applications*, 32, 8, 3685–3694. DOI: 10.1007/s00521-018-3747-z.

[33] Run-Qing Chen, Guang-Hui Shi, Wan-Lei Zhao, and Chang-Hui Liang. 2021. A joint model for IT operation series prediction and anomaly detection. *Neurocomputing*, 448, 130–139. DOI: 10.1016/j.neucom.2021.03.062.

[34] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: a scalable tree boosting system. In *Proceedings of the International conference on Knowledge Discovery and Data Mining (SIGKDD)*, 785–794. DOI: 10.1145/2939672.2939785.

[35] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven Klooster. 2009. Detection and characterization of anomalies in multivariate time series. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 413–424. DOI: 10.1137/1.9781611972795.36.

[36] Zhangyu Cheng, Chengming Zou, and Jianwei Dong. 2019. Outlier detection using isolation forest and local outlier factor. In *Proceedings of the Conference on Research in Adaptive and Convergent Systems (RACS)*, 161–168. DOI: 10.1145/3338840.3355641.

[37] Dhruv Choudhary, Arun Kejariwal, and Francois Orsini. 2017. On the Runtime-Efficacy Trade-off of Anomaly Detection Techniques for Real-Time Streaming Data. arXiv: 1710.04735 [cs, eess, stat]. Retrieved 08/18/2020 from http://arxiv.org/abs/1710.04735.

[38] Andrew A. Cook, Goksel Misirli, and Zhong Fan. 2020. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet of Things Journal*, 7, 7, 6481–6494. DOI: 10.1109/JIOT.2019.2958185.

[39] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the International Conference on Machine Learning (ICML)*, 233–240. DOI: 10.1145/1143844.1143874.

[40] Nan Ding, Huanbo Gao, Hongyu Bu, Haoxuan Ma, and Huaiwei Si. 2018. Multivariate-Time-Series-Driven Real-time Anomaly Detection Based on Bayesian Network. *Sensors*, 18, 10, 3367. DOI: 10.3390/s18103367.

[41] Dheeru Dua and Casey Graff. 2017. UCI machine learning repository. (2017).

[42] Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. 2016. Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. arXiv: 1612.06676 [cs, stat]. Retrieved 11/26/2021 from http://arxiv.org/abs/1612.06676.

[43] Yifeng Gao, Jessica Lin, and Constantin Brif. 2020. Ensemble Grammar Induction For Detecting Anomalies in Time Series. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. DOI: 10.5441/002/edbt.2020.09.

[44] Gabriel Rodriguez Garcia, Gabriel Michau, Mélanie Ducoffe, Jayant Sen Gupta, and Olga Fink. 2020. Time Series to Images: Monitoring the Condition of Industrial Assets with Deep Learning Image Processing Algorithms. arXiv: 2005.07031 [cs, eess, stat]. Retrieved 09/16/2020 from http://arxiv.org/abs/2005.07031.

[45] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, 101, 23. DOI: 10.1161/01.CIR.101.23.e215.

[46] Ary L Goldberger, Luis A N Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. The MIT-BIH Long Term Database. physionet.org. DOI: 10.13026/C2KS3F.

[47] Markus Goldstein and Andreas Dengel. 2012. Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm. In *Proceedings of the German Conference on Artificial Intelligence Poster and Demo Track (KI)*, 59–63.

[48] V Gomez-Verdejo, J Arenas-Garcia, M Lazaro-Gredilla, and Ángel Navia-Vazquez. 2011. Adaptive One-Class Support Vector Machine. *IEEE Transactions on Signal Processing*, 59, 6, 2975–2981. DOI: 10.1109/TSP.2011.2125961.

[49] Nico Görnitz, Mikio Braun, and Marius Kloft. 2015. Hidden Markov anomaly detection. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1833–1842.

[50] Scott D Greenwald. The MIT-BIH Supraventricular Arrhythmia Database. physionet.org. DOI: 10.13026/C2V30W.

[51] Scott David Greenwald. 1990. *Improved Detection and Classification of Arrhythmias in Noise-Corrupted Electrocardiograms Using Contextual Information*. PhD thesis. Havard University, Massachusetts Institute of Technology (MIT).

[52] Ralf Greis, Thorsten Ries, and D. Cu Nguyen. 2018. Comparing Prediction Methods in Anomaly Detection: An Industrial Evaluation. In *Proceedings of the Workshop on Mining and Learning from Time Series*.

[53] 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 (TKDE)*, 26, 9, 2250–2267. DOI: 10.1109/TKDE.2013.184.

[54] Medina Hadjem, Farid Nait-Abdesselam, and Ashfaq Khokhar. 2016. ST-segment and T-wave anomalies prediction in an ECG data using RUSBoost. In *Proceedings of the International Conference on E-Health Networking, Applications and Services (Healthcom)*, 1–6. DOI: 10.1109/HealthCom.2016.7749493.

[55] Michael Hahsler and Matthew Bolaos. 2016. Clustering Data Streams Based on Shared Density between Micro-Clusters. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 28, 6, 1449–1461. DOI: 10.1109/TKDE.2016.2522412.

[56] David J. Hand. 2006. Classifier Technology and the Illusion of Progress. *Statistical Science*, 21, 1. DOI: 10.1214/088342306000000060.

[57] James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143, 1, 29–36.

[58] Sahand Hariri, Matias Carrasco Kind, and Robert J. Brunner. 2019. Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. DOI: 10.1109/TKDE.2019.2947676. arXiv: 1811.02141.

[59] Zengyou He, Xiaofei Xu, and Shengchun Deng. 2003. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24, 9-10, 1641–1650. DOI: 10.1016/S0167-8655(03)00003-5.

[60] Niklas Heim and James E. Avery. 2019. Adaptive Anomaly Detection in Chaotic Time Series with a Spatially Aware Echo State Network. arXiv: 1909.01709 [cs, stat]. Retrieved 09/15/2020 from http://arxiv.org/abs/1909.01709.

[61] Nikolai Helwig, Eliseo Pignanelli, and Andreas Schutze. 2015. Condition monitoring of a complex hydraulic system using multivariate statistics. In *Proceedings of the International Instrumentation and Measurement Technology Conference (I2MTC)*, 210–215. DOI: 10.1109/I2MTC.2015.7151267.

[62] Jordan Hochenbaum, Owen S. Vallis, and Arun Kejariwal. 2017. Automatic Anomaly Detection in the Cloud Via Statistical Learning. arXiv: 1704.07706 [cs]. Retrieved 07/22/2020 from http://arxiv.org/abs/1704.07706.

[63] Victoria J. Hodge and Jim Austin. 2004. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22, 2, 85–126. DOI: 10.1007/s10462-004-4304-y.

[64] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 387–395. DOI: 10.1145/3219819.3219845.

[65] R.J. Hyndman and G. Athanasopoulos. 2018. *Forecasting: Principles and Practice.* (2nd edition). OTexts.

[66] Alexander Ihler, Jon Hutchins, and Padhraic Smyth. 2006. Adaptive event detection with time-varying poisson processes. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 207–216. DOI: 10.1145/1150402.1150428.

[67] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. 2021. Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series. *Proceedings of the VLDB Endowment (PVLDB)*, 14, 2613–2626. DOI: 10.14778/3476249.3476307.

[68] W. R. Jacobs, H. Edwards, P. Li, V. Kadirkamanathan, and A. R. Mills. 2018. Gas turbine engine condition monitoring using Gaussian mixture and hidden Markov models. *International Journal of Prognostics and Health Management*, 9.

[69] E. Keogh, T. Dutta Roy, U. Naik, and A Agrawal. 2021. Multi-dataset time-series anomaly detection competition. (2021).

[70] E. Keogh, J. Lin, and A. Fu. 2005. HOT SAX: efficiently finding the most unusual time series subsequence. In *Proceedings of the International Conference on Data Mining (ICDM)*. DOI: 10.1109/ICDM.2005.79.

[71] Eamonn Keogh, Stefano Lonardi, and Bill 'Yuan-chi' Chiu. 2002. Finding surprising patterns in a time series database in linear time and space. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 550. DOI: 10.1145/775047.775128.

[72] Chunggyeom Kim, Jinhyuk Lee, Raehyun Kim, Youngbin Park, and Jaewoo Kang. 2018. DeepNAP: deep neural anomaly pre-detection in a semiconductor fab. *Information Sciences*, 457–458, 1–11. DOI: 10.1016/j.ins.2018.05.020.

[73] Maria Kontaki, Anastasios Gounaris, Apostolos N. Papadopoulos, Kostas Tsichlas, and Yannis Manolopoulos. 2011. Continuous monitoring of distance-based outliers over data streams. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 135–146. DOI: 10.1109/ICDE.2011.5767923.

[74] Bouchra Lamrini, Augustin Gjini, Simon Daudin, François Armando, Pascal Pratmarty, and Louise Travé-Massuyès. 2018. Anomaly Detection Using Similarity-based One-Class SVM for Network Traffic Characterization. In *Proceedings of the International Workshop on Principles of Diagnosis*, 8.

[75] N Laptev, S Amizadeh, and Y Billawala. 2015. S5 - A labeled anomaly detection dataset, version 1.0 (16M). (2015).

[76] Ming-Chang Lee, Jia-Chun Lin, and Ernst Gunnar Gran. 2020. RePAD: Real-Time Proactive Anomaly Detection for Time Series. In *Proceedings of the International Conference on Advanced Information Networking and Applications (AINA)*, 1291–1302. DOI: 10.1007/978-3-030-44041-1_110.

[77] 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 *Proceedings of the International Conference on Artificial Neural Networks*, 703–716. DOI: 10.1007/978-3-030-30490-4_56.

[78] Jinbo Li, Witold Pedrycz, and Iqbal Jamal. 2017. Multivariate time series anomaly detection: A framework of Hidden Markov Models. *Applied Soft Computing*, 60, 229–240. DOI: 10.1016/j.asoc.2017.06.035.

[79] Zeyan Li, Wenxiao Chen, and Dan Pei. 2018. Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder. In *Proceedings of the International Performance Computing and Communications Conference (IPCCC)*, 1–9. DOI: 10.1109/PCCC.2018.8710885.

[80] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. 2020. COPOD: Copula-Based Outlier Detection. In *Proceedings of the International Conference on Data Mining (ICDM)*.

[81] Jessica Lin, Eamonn Keogh, and Wagner Truppel. 2003. Clustering of streaming time series is meaningless. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, 56. DOI: 10.1145/882082.882096.

[82] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn Keogh. 2020. Matrix profile goes MAD: variable-length motif and discord discovery in data series. *Data Mining and Knowledge Discovery*, 34, 4, 1022–1071. DOI: 10.1007/s10618-020-00685-w.

[83] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *Proceedings of the International Conference on Data Mining (ICDM)*, 413–422. DOI: 10.1109/ICDM.2008.17.

[84] Boris Lorbeer, Tanja Deutsch, Peter Ruppel, and Axel Kupper. 2019. Anomaly Detection with HMM Gauge Likelihood Analysis. In *Proceedings of the International Conference on Big Data Computing Service and Applications (BigDataService)*, 1–8. DOI: 10.1109/BigDataService.2019.00008.

[85] J. Ma and S. Perkins. 2003. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 1741–1745. DOI: 10.1109/IJCNN.2003.1223670.

[86] Junshui Ma and Simon Perkins. 2003. Online novelty detection on temporal sequences. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 613. DOI: 10.1145/956750.956828.

[87] Elizabeth Ann Maharaj, Pierpaolo D'Urso, and Jorge Caiado. 2019. *Time Series Clustering and Classification*. (1st edition). Chapman and Hall/CRC. ISBN: 978-1032093499.

[88] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. arXiv: 1607.00148 [cs, stat]. Retrieved 07/23/2020 from http://arxiv.org/abs/1607.00148.

[89] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*.

[90] Carla Marceau. 2000. Characterizing the behavior of a program using multiple-length N-grams. In *Proceedings of the Workshop on New Security Paradigms (NSPW)*, 101–110. DOI: 10.1145/366173.366197.

[91] Pierre-François Marteau, Saeid Soheily-Khah, and Nicolas Béchet. 2017. Hybrid Isolation Forest - Application to Intrusion Detection. arXiv: 1705.03800 [cs]. Retrieved 11/05/2020 from http://arxiv.org/abs/1705.03800.

[92] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. DOI: 10.14722/ndss.2018.23204.

[93] George B Moody and Roger G Mark. MIT-BIH Arrhythmia Database. physionet.org. DOI: 10.13026/C2F305.

[94] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. 2019. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access*, 7, 1991–2005. DOI: 10.1109/ACCESS.2018.2886457.

[95] Gyoung S. Na, Donghyun Kim, and Hwanjo Yu. 2018. DILOF: Effective and Memory Efficient Local Outlier Detection in Data Streams. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 1993–2002. DOI: 10.1145/3219819.3220022.

[96] Benjamin Nachman and David Shih. 2020. Anomaly detection with density estimation. *Physical Review D*, 101, 7, 075042. DOI: 10.1103/PhysRevD.101.075042.

[97] Takaaki Nakamura, Makoto Imamura, Ryan Mercer, and Eamonn Keogh. 2020. MERLIN: Parameter-Free Discovery of Arbitrary Length Anomalies in Massive Time Series Archives. In *Proceedings of the International Conference on Data Mining (ICDM)*, 1190–1195. DOI: 10.1109/ICDM50108.2020.00147.

[98] Zijian Niu, Ke Yu, and Xiaofei Wu. 2020. LSTM-Based VAE-GAN for Time-Series Anomaly Detection. *Sensors*, 20, 13. DOI: 10.3390/s20133738. pmid: 32635374.

[99] Oliver Obst, X. Rosalind Wang, and Mikhail Prokopenko. 2008. Using Echo State Networks for Anomaly Detection in Underground Coal Mines. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 219–229. DOI: 10.1109/IPSN.2008.35.

[100] Alberto Ogbechie, Javier Díaz-Rozo, Pedro Larrañaga, and Concha Bielza. 2017. Dynamic Bayesian Network-Based Anomaly Detection for In-Process Visual Inspection of Laser Surface Heat Treatment. In *Proceedings of the International Conference on Machine Learning for Cyber Physical Systems (ML4CPS)*, 17–24. DOI: 10.1007/978-3-662-53806-7_3.

[101] Randy Paffenroth, Kathleen Kay, and Les Servi. 2018. Robust PCA for Anomaly Detection in Cyber Networks. arXiv: 1801.01571 [cs]. Retrieved 12/16/2020 from http://arxiv.org/abs/1801.01571.

[102] Girish Keshav Palshikar. 2005. Distance-Based Outliers in Sequences. In *Proceedings of the International Conference on Distributed Computing and Internet Technology (ICDCIT)*, 547–552. DOI: 10.1007/11604655_61.

[103] S. Papadimitriou, H. Kitagawa, P.B. Gibbons, and C. Faloutsos. 2003. LOCI: fast outlier detection using the local correlation integral. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 315–326. DOI: 10.1109/ICDE.2003.1260802.

[104] John Paparrizos, Yuhao Kang, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin. 2021. TSB-AUD: an end-to-end anomaly detection benchmark suite for univariate time-series data. (2021).

[105] Daehyung Park, Zackory Erickson, Tapomayukh Bhattacharjee, and Charles C. Kemp. 2016. Multimodal execution monitoring for anomaly detection during robot manipulation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 407–414. doi: 10.1109/ICRA.2016.7487160.

[106] Daehyung Park, Yuuna Hoshi, and Charles C. Kemp. 2018. A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder. *IEEE Robotics and Automation Letters*, 3, 3, 1544–1551. doi: 10.1109/LRA.2018.2801475.

[107] Stephen Pauwels and Toon Calders. 2019. An anomaly detection technique for business processes based on extended dynamic bayesian networks. In *Proceedings of the ACM/SIGAPP Symposium on Applied Computing (SAC)*, 494–501. doi: 10.1145/3297280.3297326.

[108] Dragoljub Pokrajac, Aleksandar Lazarevic, and Longin Jan Latecki. 2007. Incremental Local Outlier Detection for Data Streams. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 504–515. doi: 10.1109/CIDM.2007.368917.

[109] Vijay Raghavan, Peter Bollmann, and Gwang S. Jung. 1989. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7, 3, 205–229.

[110] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 427–438. doi: 10.1145/342009.335437.

[111] Faraz Rasheed, Peter Peng, Reda Alhajj, and Jon Rokne. 2009. Fourier transform based spatial outlier mining. In *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, 317–324.

[112] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-Series Anomaly Detection Service at Microsoft. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 3009–3017. doi: 10.1145/3292500.3330680.

[113] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Forster, Gerhard Troster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and Jose del R. Millan. 2010. Collecting complex activity datasets in highly rich networked sensor environments. In *Proceedings of the International Conference on Networked Sensing Systems (INSS)*, 233–240. doi: 10.1109/INSS.2010.5573462.

[114] Volker Roth. 2006. Kernel Fisher Discriminants for Outlier Detection. *Neural Computation*, 18, 4, 942–960. doi: 10.1162/neco.2006.18.4.942.

[115] Peter J. Rousseeuw and Katrien Van Driessen. 1999. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41, 3, 212–223. doi: 10.1080/00401706.1999.10485670. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00401706.1999.10485670.

[116] Artem Ryzhikov, Maxim Borisyak, Andrey Ustyuzhanin, and Denis Derkach. 2019. Normalizing flows for deep anomaly detection. arXiv: 1912.09323 [cs, stat]. Retrieved 08/01/2021 from https://arxiv.org/abs/1912.09323.

[117] Mayu Sakurada and Takehisa Yairi. 2014. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. In *Proceedings of the Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, 4–11. doi: 10.1145/2689746.2689747.

[118] Stan Salvador and Philip Chan. 2005. Learning States and Rules for Detecting Anomalies in Time Series. *Applied Intelligence*, 23, 3, 241–255. doi: 10.1007/s10489-005-4610-3.

[119] Johannes Schneider, Phillip Wenig, and Thorsten Papenbrock. 2021. Distributed Detection of Sequential Anomalies in Univariate Time Series. *The VLDB Journal*, 30, 579–602. doi: 10.1007/s00778-021-00657-6.

[120] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, and Susan Frankenstein. Time series anomaly discovery with grammar-based compression. OpenProceedings.org. doi: 10.5441/002/edbt.2015.42.

[121] Mei-ling Shyu, Shu-ching Chen, Kanoksri Sarinnapakorn, and Liwu Chang. 2003. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*, 172–179.

[122] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. 2017. Anomaly Detection in Streams with Extreme Value Theory. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 1067–1075. doi: 10.1145/3097983.3098144.

[123] Maximilian Soelch, Justin Bayer, Marvin Ludersdorfer, and Patrick van der Smagt. 2016. Variational Inference for Online Anomaly Detection in High-Dimensional Time Series. arXiv: 1602.07109 [cs, stat]. Retrieved 07/22/2020 from http://arxiv.org/abs/1602.07109.

[124] Hongchao Song, Zhuqing Jiang, Aidong Men, and Bo Yang. 2017. A Hybrid Semi-Supervised Anomaly Detection Model for High-Dimensional Data. *Computational Intelligence and Neuroscience*, 2017, 1–9. doi: 10.1155/2017/8501683.

[125] 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 *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2828–2837. doi: 10.1145/3292500.3330672.

[126] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. 2006. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 187–198.

[127] Afroza Sultana, Abdelwahab Hamou-Lhadj, and Mario Couture. 2012. An improved Hidden Markov Model for anomaly detection using frequent common patterns. In *Proceedings of the International Conference on Communications (ICC)*, 1113–1117. DOI: 10.1109/ICC.2012.6364527.

[128] Pei Sun, Sanjay Chawla, and Bavani Arunasalam. 2006. Mining for Outliers in Sequential Databases. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 94–105. DOI: 10.1137/1.9781611972764.9.

[129] Xiaobin Tan and Hongsheng Xi. 2008. Hidden semi-Markov model for anomaly detection. *Applied Mathematics and Computation*, 205, 2, 562–567. DOI: 10.1016/j.amc.2008.05.028.

[130] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David Wai-Lok Cheung. 2002. Enhancing Effectiveness of Outlier Detections for Low Density Patterns. In *Proceedings of the Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, 535–548. DOI: 10.1007/3-540-47887-6_53.

[131] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. 2018. Precision and Recall for Time Series. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 1920–1930.

[132] Markus Thill, Wolfgang Konen, and Thomas Bäck. Markus-Thill / MGAB: the mackey-glass anomaly benchmark. Version v1.0.1. Zenodo. DOI: 10.5281/ZENODO.3762385.

[133] Markus Thill, Wolfgang Konen, and Thomas Bäck. 2018. Online adaptable time series anomaly detection with discrete wavelet transforms and multivariate gaussian distributions. *Archives of Data Science, Series A (Online First)*, 5, 1, 17. DOI: 10.5445/KSP/1000087327/04.

[134] Markus Thill, Wolfgang Konen, and Thomas Bäck. 2017. Time Series Anomaly Detection with Discrete Wavelet Transforms and Maximum Likelihood Estimation. In *Proceedings of the International Conference on Time Series (ITISE)*.

[135] Markus Thill, Wolfgang Konen, and Thomas Bäck. 2020. Time Series Encodings with Temporal Convolutional Networks. In *Proceedings of the International Conference on Bioinspired Methods and Their Applications (BIOMA)*, 161–173. DOI: 10.1007/978-3-030-63710-1_13.

[136] Achyut Mani Tripathi and Rashmi Dutta Baruah. 2019. Anomaly Detection in Multivariate Time Series Using Fuzzy AdaBoost and Dynamic Naive Bayesian Classifier. In *Proceedings of the International Conference on Systems, Man and Cybernetics (SMC)*, 1938–1944. DOI: 10.1109/SMC.2019.8914477.

[137] Achyut Mani Tripathi and Rashmi Dutta Baruah. 2020. Contextual Anomaly Detection in Time Series Using Dynamic Bayesian Network. In *Proceedings of the Asian Conference on Intelligent Information and Database Systems (ACIIDS)*, 333–342. DOI: 10.1007/978-3-030-42058-1_28.

[138] Rafael G. Vieira, Marcos A. Leone Filho, and Robinson Semolini. 2018. An Enhanced Seasonal-Hybrid ESD Technique for Robust Anomaly Detection on Time Series. In *Simpósio Brasileiro de Redes de Computadores (SBRC)*.

[139] Alexander von Birgelen and Oliver Niggemann. 2018. Anomaly Detection and Localization for Cyber-Physical Production Systems with Self-Organizing Maps. In *IMPROVE - Innovative Modelling Approaches for Production Systems to Raise Validatable Efficiency*. Volume 8. Springer Berlin Heidelberg, 55–71. ISBN: 978-3-662-57804-9. DOI: 10.1007/978-3-662-57805-6_4.

[140] Ke-Wei Wang and Su-Juan Qin. 2016. A hybrid approach for anomaly detection using K-means and PSO. In *Proceedings of the International Conference on Electronics, Network and Computer Engineering (ICENCE)*. DOI: 10.2991/icence-16.2016.151.

[141] Xing Wang, Jessica Lin, Nital Patel, and Martin Braun. 2016. A Self-Learning and Online Algorithm for Time Series Anomaly Detection, with Application in CPU Manufacturing. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 1823–1832. DOI: 10.1145/2983323.2983344.

[142] Xing Wang, Jessica Lin, Nital Patel, and Martin Braun. 2018. Exact variable-length anomaly detection algorithm for univariate and multivariate time series. *Data Mining and Knowledge Discovery*, 32, 6, 1806–1844. DOI: 10.1007/s10618-018-0569-7.

[143] Yi Wang, Linsheng Han, Wei Liu, Shujia Yang, and Yanbo Gao. 2019. Study on wavelet neural network based anomaly detection in ocean observing data series. *Ocean Engineering*, 186, 106129. DOI: 10.1016/j.oceaneng.2019.106129.

[144] Li Wei, Nitin Kumar, Venkata Lolla, Eamonn J. Keogh, Stefano Lonardi, and Chotirat Ratanamahatana. 2005. Assumption-free anomaly detection in time series. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, 237–240.

[145] Mark Woike, Ali Abdul-Aziz, and Michelle Clem. 2014. Structural health monitoring on turbine engines using microwave blade tip clearance sensors. In *Smart Sensor Phenomena, Technology, Networks, and Systems Integration 2014*, 167–180. DOI: 10.1117/12.2044967.

[146] Jia Wu, Weiru Zeng, and Fei Yan. 2018. Hierarchical Temporal Memory method for time-series-based anomaly detection. *Neurocomputing*, 273, 535–546. DOI: 10.1016/j.neucom.2017.08.026.

[147] Renjie Wu and Eamonn J. Keogh. 2020. Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress. arXiv: 2009.13807 [cs, stat]. Retrieved 03/15/2021 from http://arxiv.org/abs/2009.13807.

[148] Wentai Wu, Ligang He, Weiwei Lin, Yi Su, Yuhua Cui, Carsten Maple, and Stephen Jarvis. 2020. Developing an Unsupervised Real-time Anomaly Detection Scheme for Time Series with Multi-seasonality. arXiv: 1908.01146 [cs, eess, stat]. Retrieved 10/30/2020 from http://arxiv.org/abs/1908.01146.

[149] Yanshan Xiao, Bo Liu, Longbing Cao, Xindong Wu, Chengqi Zhang, Zhifeng Hao, Fengzhao Yang, and Jie Cao. 2009. Multi-sphere Support Vector Data Description for Outliers Detection on Multi-distribution Data. In *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*, 82–87. DOI: 10.1109/ICDMW.2009.87.

[150] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. In *Proceedings of the International Conference on World Wide Web (WWW)*. International World Wide Web Conferences Steering Committee, 187–196. DOI: 10.1145/3178876.3185996.

[151] Takehisa Yairi, Yoshikiyo Kato, and Koichi Hori. 2001. Fault detection by mining association rules from house-keeping data. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (-SAIRAS)*.

[152] Kenji Yamanishi, Jun-ichi Takeuchi, Graham Williams, and Peter Milne. 2004. On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. *Data Mining and Knowledge Discovery*, 8, 3, 275–300. DOI: 10.1023/B:DAMI.0000023676.72185.7c.

[153] Chao-Lung Yang and Wei-Ju Liao. 2017. Adjacent Mean Difference (AMD) method for dynamic segmentation in time series anomaly detection. In *Proceedings of the International Symposium on System Integration (SII)*, 241–246. DOI: 10.1109/SII.2017.8279219.

[154] Dragomir Yankov, Eamonn Keogh, and Umaa Rebbapragada. 2007. Disk Aware Discord Discovery: Finding Unusual Time Series in Terabyte Sized Datasets. In *Proceedings of the International Conference on Data Mining (ICDM)*, 381–390. DOI: 10.1109/ICDM.2007.61.

[155] Yuan Yao, Abhishek Sharma, Leana Golubchik, and Ramesh Govindan. 2010. Online anomaly detection for sensor systems: A simple and efficient approach. *Performance Evaluation*, 67, 11, 1059–1075. DOI: 10.1016/j.peva.2010.08.018.

[156] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *Proceedings of the International Conference on Data Mining (ICDM)*, 1317–1322. DOI: 10.1109/ICDM.2016.0179.

[157] Yufeng Yu, Yuelong Zhu, Shijin Li, and Dingsheng Wan. 2014. Time Series Outlier Detection Based on Sliding Window Prediction. *Mathematical Problems in Engineering*, 2014, 1–14. DOI: 10.1155/2014/879736.

[158] Chunkai Zhang, Shaocong Li, Hongye Zhang, and Yingyang Chen. 2020. VELC: A New Variational AutoEncoder Based Model for Time Series Anomaly Detection. arXiv: 1907.01702 [cs, stat]. Retrieved 10/01/2020 from http://arxiv.org/abs/1907.01702.

[159] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V. Chawla. 2019. A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1409–1416. DOI: 10.1609/aaai.v33i01.33011409.

[160] Rui Zhang, Shaoyan Zhang, Sethuraman Muthuraman, and Jianmin Jiang. 2007. One class support vector machine for anomaly detection in the communication network performance data. In *Proceedings of the Conference on Applied Electromagnetics, Wireless and Optical Communications (ELECTROSCIENCE)*, 31–37.

[161] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2020. Multivariate Time-series Anomaly Detection via Graph Attention Network. arXiv: 2009.02040 [cs, stat]. Retrieved 09/14/2020 from http://arxiv.org/abs/2009.02040.

[162] Zeng-Guang Zhou and Ping Tang. 2016. Improving time series anomaly detection based on exponentially weighted moving average (EWMA) of season-trend model residuals. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS)*, 3414–3417. DOI: 10.1109/IGARSS.2016.7729882.

[163] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh. 2018. Matrix Profile XI: SCRIMP++: Time Series Motif Discovery at Interactive Speeds. In *Proceedings of the International Conference on Data Mining (ICDM)*, 837–846. DOI: 10.1109/ICDM.2018.00099.

[164] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth Funning, Abdullah Mueen, Philip Brisk, and Eamonn Keogh. 2016. Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. In *Proceedings of the International Conference on Data Mining (ICDM)*, 739–748. DOI: 10.1109/ICDM.2016.0085.

[165] Johannes Ziegelmeir. 2019. Development and Comparison of Self-Learning Modules for Automated Bench Test Data Analysis of Transient Flight Engine Development Tests. Master Thesis. Technische Universität, Berlin.

[166] Zachary Zimmerman, Kaveh Kamgar, Nader Shakibay Senobari, Brian Crites, Gareth Funning, Philip Brisk, and Eamonn Keogh. 2019. Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond. In *Proceedings of the ACM Symposium on Cloud Computing (SOCC)*, 74–86. DOI: 10.1145/3357223.3362721.