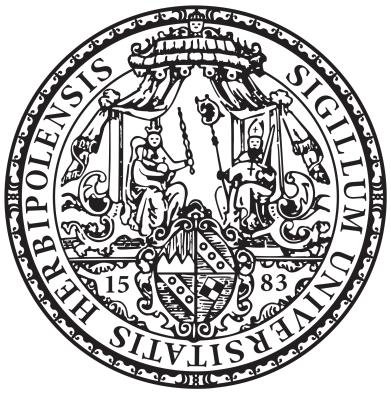
Time Series Anomaly Detection Benchmarking

Philip Spaier

3110375



Seminararbeit

Lehrstuhl für Wirtschaftsinformatik und Business Analytics
Universität Würzburg

Betreuer: Prof. Dr. Gunther Gust
Assistent: Viet Nguyen

Würzburg, den 05.04.2025

Contents

List of Figures

List of Tables

Abstract

Eine Kurzzusammenfassung der Vorgehensweise und der wesentlichen Ergebnisse.

Allgemeine Merkmale

- Objektivität: Es soll sich jeder persönlichen Wertung enthalten.
- Kürze: Es soll so kurz wie möglich sein.
- Verständlichkeit: Es weist eine klare, nachvollziehbare Sprache und Struktur auf.
- Vollständigkeit: Alle wesentlichen Sachverhalte sollen enthalten sein.
- Genauigkeit: Es soll genau die Inhalte und die Meinung der Originalarbeit wiedergeben.

1 Literature Review

Time Series Anomaly Detection (TSAD), as a subcategory of the broader field of Anomaly Detection, has seen increased attention since the start of the twenty first century. With the internet having established itself as a persistent and omnipresent force in every imaginable aspect of human life, time series data can be found in abundance. Modern developments in Internet-of-Things (IoT) applications, the digitization of financial data, and a massive rise in the consumption of streaming services have contributed to an exponential growth of time series data [source needed]. This in turn has made the manual search of potential anomalies in many fields completely infeasible, leading to an increased demand for automated anomaly detection methods. While there is a continuously growing repertoire of such automated detection methods, the lack of a generally accepted and reliable benchmark makes not just further developments but also the selection of appropriate models difficult. In the following sections of this literature review, I will provide the reader with a better understanding of context independent Time Series Anomaly Detection, the most commonly applied methods, and the current state of benchmarking.

1.1 Time Series Data and Anomaly Detection Definition

Time Series Data, as used in the rest of this thesis, shall be defined as follows: a sequence of data or observations, typically indexed by or associated with specific timestamps, collected in chronological order over a period of time. For the purpose of analysis, continuous signals must be converted into individual data points. Each datapoint can either represent a binary state (1 or 0), be a numerical value measured on a ratio scale (eg. number of occurrences), or a numerical value measured on an interval scale (eg. temperature on a Celsius scale). A time series with a dimensionality of one (only a single feature) will be referred to as "univariate", while a time series with higher dimensionality (multiple features) will be referred to as "multivariate".
An anomaly will be defined as follows: an abnormal, rarely occurring data point or sequence, that has to be be detectable with exclusively context independent methods. Individual anomalous data points will be referred to as "point based" anomalies. Multiple consecutive anomalous points, each of which might be unremarkable on their own, while displaying unusual behavior as a sequence, will be referred to as "sequence based" or "collective" anomalies (Liu and Paparrizos, 2024, p. 3; Chalapathy and Chawla, 2019, p. 8). A separate category of anomalies would be context dependent ones. Those are data points or sequences, possibly indistinguishable from normal ones if analyzed without context, but if combined with additional information about the field or time series, are considered anomalous (Chalapathy and Chawla, 2019, pp. 7-8). Context dependent anomalies will not be topic of the research presented here.
Given those definitions, Time Series Anomaly Detection is therefore the task of correctly and autonomously identifying anomalies within a given time series.

Insert Images of point vs sequence.

1.2 Relevant Fields

The following is an overview of fields relying on Time Series Anomaly Detection. It is a non-exhaustive list, simply highlighting some of the most prominent use cases to provide context.

Illicit Activity and Fraud Detection: With the global financial system relying primarily on digital transactions, it has become crucial to detect fraudulent activities as quickly and accurately as possible. A particularly obvious example is credit card fraud, creating an estimated yearly loss in the billions of dollar (Zhou, Xun et al., 2018, p. 2). Companies like Visa and Mastercard put great emphasis on being able to detect anomalous transactions in real time to then analyses them and prevent potential harm to their customers (*Visa Acceptance Solutions* 2025). While credit card fraud is a prominent application, the scope of financial anomaly detection extends significantly further, playing a critical role in the operations of stock exchanges, brokerage firms, and banks. These institutions leverage anomaly detection techniques to identify various illicit activities, ensure market integrity, manage operational risks, and comply with stringent regulatory requirements (*Deutsche Börse* 2025).

Healthcare: Healthcare critically relies on analyzing physiological signals, such as those captured by the electrocardiogram (ECG), which provides vital time series data reflecting the heart's electrical activity. While historically, ECG analysis has focused on identifying established patterns of known heart diseases, this approach often fails to detect rare or atypical anomalies that do not fit predefined categories, potentially missing critical conditions. To address this issue, Time Series Anomaly Detection has been introduced for the purpose of detecting such rare anomalies that would go unnoticed by conventional pattern classification (Jiang et al., 2024, p. 1-2).

Website Traffic: A common threat faced by web-services are so called Denial or Service (DoS) and Distributed Denial of Service (DDoS) attacks. These include hitting a webserver with so many requests that the systems becomes inoperational and can no longer service legitimate users (*Bundesamt für Sicherheit in der Informationstechnik* 2025). A significant challenge in detecting these attacks is that the malicious traffic can often mimic normal network traffic, making it difficult for traditional packet-based intrusion detection systems or statistical methods reliant on fixed thresholds to accurately identify attacks, especially when they are hidden within legitimate flows. Time series analysis allows systems to observe and distinguish the instant changes in network traffic that indicate an attack, even when individual packets or simple statistics are insufficient. Time series anomaly detection provides a means to autonomously identify and localize potentially harmful deviations within the network traffic and thereby ensure the availability and reliability of services (Fouladi, Ermiş, and Anarim, 2020, pp. 1-2).

The list extends far beyond the fields named above. Time Series Anomaly Detection can be also found in astronomy (Huijse et al., 2014), earth sciences, manufacturing (Zamanzadeh Darban et al., 2024, p. 1), cybersecurity, and law enforcement (Boniol et al., 2024, p. 1).

1.3 Detection Methods

Detection methods, in common descriptions and within the scientific literature alike, are often grouped or distinguished by a variety of aspects. This categorization can sometimes lack a consistent taxonomy. To provide a clearer framework, I will now systematically explain and categorize these methods through three key perspectives:

- Degree of supervision
- Architecture
- Technique

1.3.1 Degree of Supervision

Unsupervised models operate on data without any explicit labels distinguishing normal from anomalous instances. While they don't require pre-labeled data, they typically do require a training or fitting phase. During this phase, the model learns the inherent structure, patterns, distributions, or densities from the unlabeled dataset.

Semi-supervised models are trained exclusively on data that is known or assumed to be 'normal.' They do not require labeled anomalies for training. The model learns a precise representation or boundary of this normal behavior. During deployment, any new data instance that significantly deviates from this learned model of normalcy is flagged as an anomaly.

Supervised models require a dataset where both normal and anomalous instances are explicitly labeled beforehand. The model is then trained to learn the distinguishing features or decision boundaries that separate these classes, effectively treating anomaly detection as a (often highly imbalanced) classification problem. (Boniol et al., 2024, pp. 5-6; Liu and Paparrizos, 2024, p. 3; Schmidl, Wenig, and Papenbrock, 2022, p. 3-4).

1.3.2 Architecture

Statistical models identify anomalies by relying on statistical assumptions to detect deviations from expected data distributions. They often involve fitting a distribution model to the data and measuring abnormality based on probabilities or distances from the calculated distribution. Statistical models often require a threshold to be set beforehand(Liu and Paparrizos, 2024, p. 6-7; Fouladi, Ermiş, and Anarim, 2020, p. 1).

Neural Network based models are a collection of distributed, adaptive, non-linear processing units with adjustable weights (Guresen and Kayakutlu, 2011, p. 427). They rely on a training dataset and are often semi-supervised. Deep neural networks, a subcategory of neural networks, model spacial and temporal dependencies (Liu and Paparrizos, 2024, p. 6-7; Zamanzadeh Darban et al., 2024, p. 6).

Foundational Models utilize transfer learning, using knowledge from a different class of tasks and then applying it on the target task. These models are pre-trained and are then being fine-tuned (Bommasani et al., 2022, p. 4). In the context of TSAD, those models are GPT

models fine tuned on time series data, general purpose time series models, or originally time series classification models now used for anomaly detection (Liu and Paparrizos, 2024, p. 7).

1.3.3 Technique

Distance based models work on the idea that anomalous points or sequences will further away when using a distance measurement. They can be either be compared to their nearest neighbor, all other points/subsequences, or cluster centers (Schmidl, Wenig, and Papenbrock, 2022, p. 6). Such distances are calculated in various ways depending on the model and implementation, with the most common definitions being the Euclidean distance or the Z-normalized Euclidean distance. Distance based models use only the x- and y-axis data, with no labels being required (Boniol et al., 2024, p. 8).

Forecasting models learn the normal patters of a time series and, often using a sliding context window, forecast the next datapoint in the series. The forecasted and actual data points are then compared, with the difference being used for an anomaly score. Given a high enough anomaly score, a point is considered an anomaly. Such models are usually semi-supervised (Schmidl, Wenig, and Papenbrock, 2022, p. 4-5).

Isolation Tree Models use ensembles of random trees, selecting random features and splits, to separate points or sequences from each other. It operates on the idea that anomalies require fewer steps to be separated from the rest of the data than normal points/sequences. For each point/sequence, the distance from the root is calculated. The shorter a distance is, the more likely is a point/sequence to be an anomaly. These models can be both unsupervised and supervised (Schmidl, Wenig, and Papenbrock, 2022, p. 6-7)

Distribution based models estimate a distribution of the time series and then score individual points or sequences as anomalous or normal based on it. Anomalous points are expected to have a low probability. Alternatively to probabilities, the anomaly score can also be calculated using likelihoods or distances. These models are generally unsupervised or occasionally semi-supervised (Schmidl, Wenig, and Papenbrock, 2022, p. 6).

Graph based models methods turn time series data, or parts of it, into a graph structure. This graph represents the different types of patterns (subsequences) found in the data as nodes, and how these patterns follow each other over time as connections (edges) between the nodes. Anomalies are then determined based on usual structures or behaviors found in the graph (Boniol et al., 2024, p. 23-24). Graph based time series models can be further divided in multiple subcategories, including AutoEncoder- and GAN-based methods, as well as predictive graph models (Ho, Karami, and Armanfard, 2025).

Reconstruction models learn a time series' features and patterns by encoding normal data into a low dimensional space. Given a test dataset, they compress test data and reconstruct it using their model based on that low-dimensional space. Should a point or sequence of this reconstructed version deviate substantially from the actual data, then it is labeled as anomalous. These models are often considered semi-supervised because they typically use

normal labeled data for training. However, models that do not rely on a training dataset and instead directly encode and reconstruct the test data also exist, operating in an unsupervised manner. (Schmidl, Wenig, and Papenbrock, 2022, p. 5).

Encoder based models operate similarly to reconstruction models. They compress a given time series into a low-dimensional representation, but instead of reconstructing it, they directly compare this compressed version to their model of normal time series. Anomalous points or sequences might have unusual encoded representations, and their deviations from the normal model are then used to calculate an anomaly score (Schmidl, Wenig, and Papenbrock, 2022, p. 5-6).

1.4 Performance Metrics

For the effective evaluation of a models performance, as defined by Paparrizos, Boniol, et al., 2022, metrics have to fullfil the following criteria:

- *Robustness to Lag:* The evaluation measure should be insensitive to slight temporal shifts or lags in anomaly scores.
- *Robustness to Noise:* The evaluation measure should be stable and unaffected by noise in the anomaly scores.
- *Robustness to Anomaly Cardinality Ratio:* The evaluation measure's score should not be influenced by the proportion of anomalies in the data.
- *High Separability between Accurate and Inaccurate Methods:* The measure must effectively distinguish between accurate and inaccurate detection methods.
- *Consistency:* The measure should produce repeatable scores for similar data and consistently rank different methods.

Commonly applied performance measures for TSAD can generally be classified based on two characteristics: Point-wise or Range-wise, and Threshold-dependent or Threshold-independent.

1.4.1 Point-wise or Range-wise

Point-wise evaluation measures look at each anomalous point independently, determining in a binary fashion whether a model classified them correctly as normal or anomalous (Liu and Paparrizos, 2024, p. 7). These measures suffer from a variety of issues. Most crucially, they can unfairly penalize methods that detect only part of an anomalous range or whose detection peak doesn't perfectly align with the labeled range. Further more, they are sensitive to temporal lag. Should an anomalous data point be detected slightly before or after the actual anomaly occurs, a fully point-wise metric will score it with an unreasonably low score (Paparrizos, Boniol, et al., 2022, p. 2778).
Range-wise measures look at anomalies not just from the perspective of individual points but take sequences into consideration. For anomalous sequences, their evaluation can involve determining how much the detected and the actual sequence overlap. Additionally, such measures may incorporate strategies like adequately handling lag (e.g., by considering

an anomaly detected if it's within a specified range of an actual one, even if not at the exact spot) or including a cardinality factor to penalize models that incorrectly segment anomalies (such as detecting multiple short ones for a single large event, or vice-versa) (Liu and Paparrizos, 2024, p. 7).

1.4.2 Threshold-dependent or Threshold-independent

Threshold-dependent measures require a threshold to be set that determines whether an anomaly score classifies a value as anomalous or normal. This can be done based on statistical assumptions, or using dynamic algorithms that adjust to the data and results (Boniol et al., 2024, p. 38-39). Setting these thresholds automatically, however, is often difficult when working with large and diverse datasets, and the chosen thresholds can drastically change a metric's accuracy. Noise and the normal-to-anomalous ratio in a time series can be particularly problematic (Paparrizos, Boniol, et al., 2022, p. 2777-2778).
Threshold-independent measures evaluate the performance of a time series anomaly detection method without needing a specific score cutoff to decide what constitutes an anomaly. Instead of relying on a fixed threshold, they assess how effectively the method's anomaly scores rank true anomalies higher than normal data points across the entire range of scores (Boniol et al., 2024, p. 39-41).

1.4.3 Definition and Classifgication of Metrics

The following defines the most commonly used metrics and classifies them into the above described categories (Paparrizos, Boniol, et al., 2022, p.2776-2780):

- *Precision / Range Precision:* number of correctly identified anomalies over all anomalies.
- *Recall (TPR) / Range Recall:* number of correctly identified anomalies over all anomalies.
- *F-Score / Range F-Score:* Harmonic Mean of Precision and Recall.
- *False Positive Rate (FPR):* number of points wrongly identified as anomalies over the total number of normal points.
- *AUC-ROC:* area under the curve corresponding to TPR on the y-axis and FPR on the x-axis at all threshold levels.
- *AUC-Precision:* area under the curve corresponding to the Recall on the x-axis and Precision on the y-axis at all threshold levels.
- *VUS-ROC:* generating multiple ROC curves for a range of different buffer lengths. These stacked ROC curves form a 3D surface, and VUS-ROC is the volume beneath this surface.
- *VUS-Precision:* generating multiple Precision-Recall curves for a range of different buffer lengths. These stacked PR curves form a 3D surface, and VUS-Precision (VUS-PR) is the volume beneath this surface.

	Threshold-dependent	Threshold-independent
Point-wise	Precision Recall False Positive Rate F-Score	AUC-ROC AUC-Precision
Range-wise	Range Precision Range Recall Range F-Score	VUS-ROC VUS-Precision

Table 1: Evaluation Measures

1.5 State of Benchmarking

While time series anomaly detection is a well-established field, most advancements in systematic bench marking have been made within the last decade. The following will provide an overview over the most important papers and datasets contributing to this endeavor. As always, given the large corpus of work, this is not an exhaustive list.

Yahoo (2015): Yahoo provides one of the earliest available labeled large scale TSAD datasets. It consists of real data with time series from various Yahoo services and synthetic data, containing trends, noise, and seasonality (*Yahoo* 2015).

Numenta Anomaly Benchmark (NAB), 2015: The Numeta Anomaly Benchmark is often considered to be the first large scale open source benchmarking environment for TSAD. At the time of release, it contained 58 datasets, made up of a mixture of artificial and real time series. The labels are first created by multiple humans, then combined into a ground truth by an algorithm. NAB uses a custom threshold dependent scoring function for the evaluation of a model's performance, designating high value to an algorithms ability to detect an anomaly as early as possible. It is designed specifically for real time anomaly detection, not static analysis. Therefore, only unsupervised models can be tested, with no training/test split of the data (Lavin and Ahmad, 2015).

Illusion of Progress (2021): Wu and Keogh, 2021 provide substantial criticism regarding the state of TSAD benchmarking. The authors find most previously created datasets to contain one or multiple of the following flaws:

- *Triviality:* Many anomalies are "are so simple that solving them seems pointless or even absurd" (Wu and Keogh, 2021, p. 2).
- *Unrealistic Anomaly Density:* Many time series have anomaly rates so high that they can realistically no longer be defined as anomalous. The task turns into a classification problem.
- *Mislabeled Ground Truth:* Many time series have data that, without context, appears to be mislabeled. Normal data points are falsely labeled as anomalies and vice versa.
- *Run-to-failure Bias:* In the case of real data, many systems are operated until failure.

This results in an unusually high anomaly count towards the end of the dataset.

The authors introduce their UCR Time Series Anomaly Archive, a collection of 250 curated univariate time series from human medicine, biology, meteorology and industry to provide a dataset that combats these issues (Wu and Keogh, 2021).

Exathlon (2021): Jacob et al., 2021 introduce the first public benchmarking suite for multivariate time series anomaly detection. Their dataset consists of time series collected 100 executions of 10 distributed streaming jobs on a Spark cluster. The datasets contain primarily sequence based anomalies; the tested models are primarily semi-supervised. Exathlon evaluates detection performance (Precision, Recall, F-Score, and AUPRC) and computational efficiency.

TODS (2021): Lai et al., 2021 contribute crucially to the taxonomy for outliers and synthetic anomaly injection. 'Point-wise' outliers. including their subcategories 'global outliers' and 'contextual outliers', as well as 'pattern-wise' outliers, including their subcategories 'shapelet outliers', 'seasonal outliers', and 'trend outliers' are introduced. The authors provide 35 new synthetic datasets and 4 new multivariate real datasets, in addition to 9 existing datasets for their benchmark.

GutenTAG (2022): Schmidl, Wenig, and Papenbrock, 2022 implement and evaluate 71 different algorithms on 976 time series across a variety of fields, both univariate and multivariate. Further more, they introduced the GutenTAG synthetic dataset generator, allowing the creation of time series with with different lengths, variances, amplitudes, frequencies, and dimensions.

TSD-UAD (2022): Paparrizos, Kang, et al., 2022 introduce TSB-UAD, a new comprehensive end-to-end benchmark suite designed for evaluating univariate TSAD methods. The benchmark aims to address limitations in current practices, such as the reliance on biased proprietary/synthetic data or limited public datasets. TSB-UAD provides a reproducible platform for researchers by collecting, processing, and formatting a large and diverse set of time series with labeled anomalies. The TSB-UAD suite encompasses 13,766 univariate time series across various domains, featuring high variability in anomaly types (point, contextual, collective), ratios, and sizes. It includes 18 previously proposed public datasets and contributes two new collections: 126 "artificial" datasets derived from transforming time-series classification data (leveraging the UCR Archive) and 92 "synthetic" datasets generated by applying various global, local, and subsequence transformations to public data to introduce new anomalies and increase detection difficulty. The benchmark suite also provides a Python library to handle pre-processing, post-processing, data generation, transformation, and includes statistical analysis methods (Friedman, Nemenyi tests) for comparing algorithms. It evaluates 12 representative AD methods and introduces measures (Relative Contrast (RC), Normalized Clusteredness of Abnormal Points (NC), Normalized Adjacency of Normal/Abnormal Cluster (NA)) to quantify dataset difficulty. Data and code are made publicly available in a Github repository.

TSD-AD (2024): TSB-AD is presented as a new comprehensive benchmark suite for univari-

ate and multivariate time-series anomaly detection (TSAD), designed to address limitations in existing evaluations stemming from flawed datasets, unreliable measures, and inconsistent practices. The benchmark offers 1070 high-quality, curated time series derived from 40 diverse public datasets, substantially increasing the scale and integrity of available data for benchmarking. TSB-AD includes 40 representative AD algorithms spanning statistical, neural network, and foundation model categories. The paper identifies VUS-PR as a robust and reliable evaluation metric, contrasting it with traditional measures prone to biases like Point Adjustment and sensitivity to lag. TSB-AD is released open-source to provide a stable platform for research and establish a leaderboard (Liu and Paparrizos, 2024).

When looking at the evaluation results of all major benchmarks, it is difficult to point to any specific model as the conclusively best. Given the constantly evolving benchmarking criteria and evolution of dataset quality, this is not surprising. Overall, traditional models have been found across multiple papers and benchmarks to rival, or in many case outperform, newer more complex architectures. Foundational models are promising for point-wise anomalies but get beaten decisively for sequence-wise anomalies. [insert sources]

2 Dataset Analysis

3 Replication of TSB-AD Benchmark Results

3.1 Setup and Implementation

For the recreation of the TSB-AD benchmark results, the following core components were used:

- AMD Ryzen 5 7500F, 6C/12T, 3.70-5.00GHz
- Nvidia 3060TI 8 GB GDDR6 VRAM
- 32 GB DDR5 RAM
- OS: Windows 11

For the actual implementation, the TSB-AD Github repository was cloned and instructions provided by the authors on how to set up the correct development environment were followed. Given the high number of datasets, the benchmark was performed primarily running 3 models in parallel. Two non-resource intensive models using only the CPU (Statistical Methods as classified by the authors of TSB-AD) and one GPU utilizing model (Neural Network-based Method or Foundation Model-based Method). GPU utilization would frequently reach 100%, whereas CPU utilization rarely even exceeded 60%.

Despite extensive efforts, I was not able to get all provided anomaly detection models to run reliably. These are the models which I could not successfully implement:

- **MomentFM, TimesFM, and TimesFT** require packages that cause unresolvable dependency incompatibilities with TSB-AD.

- **Chronos and OFA** have required packages that do not work on my system for an unidentified reason.
- **Lag-Lama** has a required file which I can not find in the provided Github repository.
- **EIF** causes the system to run out of memory and crash.
- **Donut** creates a for me unidentifiable error, seemingly stemming from the donut.py file itself.
- **NORMA** and Series2Graph are commercially licensed models.

3.2 Results

3.2.1 Univariate Results

Since Liu and Paparrizos, 2024 only publish the full VUS-PR results for each individual dataset and model, the comparison will focus on this metric. Table 2 provides information about the differences found between both benchmark evaluations for univariate detection models.

Model	Avg. Abs. Diff.	Max. Abs. Diff.	File ID	Num. of Anom.	Datasets >5% Diff.	Datasets >25% Diff.	Datasets >50% Diff.
IForest	0.163	0.999	552	1	195	81	24
LSTMAD	0.108	0.951	849	2	118	36	12
USAD	0.106	0.982	849	2	111	44	27
TranAD	0.093	0.934	865	2	109	47	19
OmniAnomaly	0.082	0.966	865	2	101	44	13
CNN	0.079	0.987	645	1	125	29	12
AnomalyTrans	0.053	0.603	191	2	26	10	5
KShapeAD	0.048	0.999	780	1	71	14	8
SAND	0.047	0.994	680	1	70	17	7
FITS	0.038	0.450	579	1	86	6	0
MOMENT_ZS	0.007	0.413	579	1	9	1	0
AutoEncoder	0.006	0.152	658	1	8	0	0
LOF	0.002	0.075	811	3	2	0	0
POLY	0.001	0.288	534	1	2	1	0
MatrixProfile	5.96e-05	0.015	017	1	0	0	0
SR	8.24e-06	0.002	534	1	0	0	0

Statistical Methods
Neural Network-based Method
Foundation Model-based Method

Table 2: Summary of absolute differences in VUS-PR for univariate datasets.

For most models, the average difference stayed firmly below 0.100, with IForest, LSTMAD,

and USAD being the main exceptions. Beyond these averages, roughly half of all models had at least one dataset where the absolute score differed by more than 0.950. Furthermore, over a third of all models had more than 10 datasets where the difference between the benchmark evaluation by the original authors and the benchmark evaluation run by me differed by at least 50%. These high discrepancies typically occurred for datasets with a low anomaly count, making even small variations in detected anomalies strongly noticeable in the scores. When comparing model types, statistical methods generally showed higher consistency between runs than Neural Network-based methods. This is somewhat to be expected, as the inherent complexity, numerous parameters, and sensitivity to initialization or even minor data variations in Neural Networks can lead to greater variability in outcomes, even when attempting to fix seeds. The notable exception here was IForest; its inconsistency was somewhat surprising, especially given it was configured with 200 n_estimators and a consistent seed, factors which should generally promote stability. Interestingly, Moment (zero-shot) demonstrated remarkably high consistency, outperforming even 3 statistical methods and significantly surpassing most other neural network models. This high consistency is largely attributable to its fundamental zero-shot nature: as a pre-trained model, it involves little re-training randomness in each run, relying instead on fixed, deterministic weights for its inferences. Multiple datasets appeared twice as an entry for the highest absolute difference. These are: #849, #865, #579, #534.

3.2.2 Multivariate Results

Table 3 provides information about the differences found between both benchmark evaluations for multivariate detection models.

The results for multivariate detections models are in many ways similar to those of univariate detections models. Only 5 out of 20 models display an average difference between runs of more than 0.100, with those being TranAD, PCA, OmniAnomaly, USAD, and IForest. 1/4th of all models had more than 10 datasets where the difference between the benchmark evaluation by the original authors and the benchmark evaluation run by me differed by at least 50%. These high discrepancies typically occurred for datasets with a low anomaly count. As with the univariate models, when comparing model types, statistical methods generally showed higher consistency between runs than Neural Network-based methods. The exceptions here being the statistical models PCA and IForest (ranking 2nd and 5th), and the Neural Network-based model AutoEncoder (ranking 14th). HBOS was the only model to have its largest difference occur during a dataset with multiple hundred anomalies. Given the small absolute difference (0.00000291), this seems insignificant. Most prone to differences was the dataset #175, appearing 5 times as the dataset with the highest absolute difference, followed by #156 with 4 appearances.

3.2.3 Benchmarking Conclusion

Across both univariate and multivariate anomaly detection models, the observed trends regarding benchmark evaluation consistency are largely parallel. While most models dis-

Model	Avg. Abs. Diff.	Max. Abs. Diff.	File ID	Num. of Anom.	Datasets >5% Diff.	Datasets >25% Diff.	Datasets >50% Diff.
TranAD	0.185	0.910	175	4	83	36	28
PCA	0.155	0.855	156	1	91	44	17
OmniAnomaly	0.148	0.910	175	4	110	36	11
USAD	0.147	0.909	175	4	109	36	8
IForest	0.122	0.942	156	1	85	29	12
CNN	0.095	0.922	187	6	70	15	12
LSTMAD	0.086	0.952	175	4	57	10	6
FITS	0.065	0.479	007	1	64	11	0
AnomalyTransformer	0.036	0.614	175	4	33	5	1
CBLOF	0.030	0.870	194	5	8	7	5
KMeansAD	0.019	0.566	046	1	19	1	1
RobustPCA	0.015	0.396	014	1	17	1	0
EIF	0.015	0.095	008	2	1	0	0
AutoEncoder	0.009	0.318	017	1	6	2	0
KNN	0.008	0.504	156	1	8	1	1
MCD	0.007	0.544	194	5	2	1	1
LOF	0.004	0.340	156	1	2	1	0
OCSVM	0.004	0.370	017	1	2	2	0
HBOS	1.71e-08	2.91e-06	123	463	0	0	0
COPOD	9.61e-18	2.22e-16	008	2	0	0	0

Statistical Methods
Neural Network-based Method

Table 3: Summary of absolute differences in VUS-PR for multivariate datasets.

played relatively small average differences between the two evaluations, a significant portion exhibited substantial discrepancies on individual datasets, particularly those with low anomaly counts where even minor variations in anomaly detection profoundly impacted scores. A consistent finding was that statistical methods generally demonstrated higher consistency than Neural Network-based models. However, notable exceptions exist, such as IForest often showing surprising inconsistency despite its configuration, and conversely, zero-shot models like Moment achieving remarkable stability due to their pre-trained, less randomizable nature. Given that most of the highest absolute differences can be found on recurring datasets, it is reasonable to assume that the specific type of anomalies and data sequences significantly influences the consistency between benchmarking runs. Furthermore, since a large portion of datasets with equally low anomaly counts are not present on this list, it can be ruled out that the higher differences between runs are entirely attributable solely to their low anomaly count.

However, it's important to stress that these observations - for univariate and multivariate models alike - are based on a very limited sample size of just two runs (the original authors' and my own). Therefore, while these results are noteworthy, this comparison doesn't provide any conclusive results regarding the general reproducibility of these models without further, more extensive investigation.

4 Dataset Setup

A common hurdle cited amon Time Series Anomaly Detection researchers is the lack of high quality, accurately labeled data (Liu and Paparrizos, 2024, p. 2; Wu and Keogh, 2021, p. 1-6; Paparrizos, Kang, et al., 2022, p. 1). Out of the existing datasets that fulfill the desired criteria, most are either entirely synthetic or contain artificially inserted anomalies that attempt to mimic real world scenarios based on statistical assumptions. To contribute to the current state of research, I will be providing 4 real, high quality, univariate datasets with accurate labels.

4.1 Dataset Description

The provided datasets contain information about RAM utilization during matrix multiplication. While the first 3 datasets each include 1 unique type of anomaly, the fourth dataset combines all 3 anomalies.

Baseline Load: All four datasets have the same baseline load for their default state. A script performs continous matrix multiplications, randomly selecting a matrix of size 3000, 3500, 4500, or 5000 and performing 3 multiplications of the selected size. This is follwed by a sleep period of 4 seconds before another set of matrix multiplication resumes. Such a sequence of multiplications follwed by a sleep time will be defined as an "baseline instance" going forward. Each dataset will first run for a predetermined amount of time without any anomalies, which can be used as the training split for semi-supervised models.

Dataset 1 - Large Matrix Injection: For the first dataset, a particularily large matrix multiplication is performed as the anomaly. The generation starts with a large matrix multiplication to prepare the memory and avoid any changes in baseline RAM utilization after the first anomaly injection.

- *Baseload* – Matrix Size: [3000, 3500, 4500, 5000] | Number of Multiplications: 3 | Sleep Time: 4 sec
- *Runtime* – Total: 90 min | Anomaly-free: 20 min
- *Preparation Matrix* – Matrix Size: 6000 | Multiplications: 4
- *Anomaly* – Type: Large Matrix Injection | Matrix Size: 6000 | Multiplications: 4 | Frequency: 1/50 baseload instances

Dataset 2 - Sleep Time Injection: For the second dataset, an additional sleep time is performed instead of a matrix multiplication. This results in a sequence consisting of: regular sleep time -> anomaly sleep time -> regular sleep time. Given that the anomaly does not require more memory than the baselaod calculations, no preparation is requried at the start of the dataset generation.

- *Baseload* – Matrix Size: [3000, 3500, 4500, 5000] | Number of Multiplications: 3 | Sleep Time: 4 sec
- *Runtime* – Total: 90 min | Anomaly-free: 20 min
- *Anomaly* – Type: Sleep Time Injection | Sleep Time Duration: 2 sec | Frequency: 1/50 baseload instances

Dataset 3 - Medium Matrix Injection: For the third dataset, a matrix multiplication is performed as the anomaly. The size of the matrix falls between the already existing matrices in the baselaod calculations. Given that the anomaly does not require more memory than the baselaod calculations, no preparation is requried at the start of the dataset generation.

- *Baseload* – Matrix Size: [3000, 3500, 4500, 5000] | Number of Multiplications: 3 | Sleep Time: 4 sec
- *Runtime* – Total: 90 min | Anomaly-free: 20 min
- *Preparation Matrix* – Matrix Size: 4000 | Multiplications: 3
- *Anomaly* – Type: Medium Matrix Injection | Matrix Size: 4000 | Multiplications: 3 | Frequency: 1/50 baseload instances

Dataset 4 - Combined Anomaly Injection: For the fourth dataset, all three anomalies are injected. The generation starts with a large matrix multiplication to prepare the memory and avoid any changes in baseline RAM utilization after the first anomaly injection. The same combined anomaly frequency is maintained as in the previous three datasets. The total length of the dataset is adjusted upwards.

- *Baseload* – Matrix Size: [3000, 3500, 4500, 5000] | Number of Multiplications: 3 | Sleep Time: 4 sec
- *Runtime* – Total: 160 min | Anomaly-free: 25 min
- *Preparation Matrix* – Matrix Size: 6000 | Multiplications: 4
- *Anomaly 1* – Type: Large Matrix Injection | Matrix Size: 6000 | Multiplications: 4

- *Anomaly 2* – Type: Sleep Time Injection | Sleep Time Duration: 2 sec
- *Anomaly 3* – Type: Medium Matrix Injection | Matrix Size: 4000 | Multiplications: 3
- *Combined Anomaly Frequency* 3

4.2 Logging

With real data, the biggest challenge is not to find extensive datasets, its to find datasets with accurate labels. Without those, (semi-)supervised models can not function and no models can be be properly evaluated. Given this challenge, creating an accurate logging mechanism that will correctly label datapoints as either normal or anomalous was at the core of this entire seminar paper.

The logging system employs a dual approach that provides a comprehensive view of each action executed and acurate monitoring of Resident Set Size (RSS) memory usage. It captures both discrete events and continuous system state. At its center is a centralized, thread-safe function called log_entry which appends timestamped records to a CSV file. It utilizes a global threading.Lock to serialize file access, preventing race conditions and ensuring that multiple logging instances can write log entries without corruption.

The first part is event-driven and activates whenever the script is started, an instance of matrix multiplication is started, or an instance of matrix multiplication concludes. These event logs provide a clear overview of all operations.
The second part of my logging system (continuous_logger_thread) continously samples current RSS memory usage while running in parallel to the rest of the working script. It monitors the current state of operations in 0.1 second intervals. If it detects the state changing from 'idle' to 'working', it waits for 0.3 seconds before recording RSS memory usage. During testing, I encountered the issue of a matrix multiplication having technically started but no changes in actual hardware utilisation has yet occured. The 0.3 seconds delay ensures that the first datapoint after a workload has started is correctly labeled as "working". This is particularily important for when an anomaly has been injected. Similarly, when it detect a state change from 'working' to 'idle', the logger waits 0.1 seconds before recording RSS memory usage. This exists to prevent situations where the matrix multiplication has already ended but the memory was not yet released. While none of these state changes occure, the logger takes a snapshot every 1.0 seconds. The 0.3 seconds and 0.1 seconds where each originally selected during extensive testing on a dataset that was supposed to track CPU usage. When used on memory, it continued to function as intended and was therefore kept.

4.3 Hardware Setup

The simulations are ran on the previously described system that was used for the replication of the TSB-AD benchmark. Since Windows is notorious for unannounced and unwanted background activities that can often neither be reliably prevented, nor accounted for, a VMWare Workstation Virtual Machine was used. The specific setup and resource allocations

were as follows:

- 4 Processor Cores (2 Processors with 2 Cores each)
- 8GB DDR5 RAM
- OS: Linux Ubuntu 24.04.2 LTS 64-bit
- Automatic Updates Disabled
- Networking Disabled

4.4 Statistical Overview

5 Conclusion

6 Zitieren und Referenzieren

7 Abbildungen

Abbildungen erfordern das package *graphicx*. Idealerweise verwendet man Vektorgrafiken oder hochaufgelöste Bitmaps. Eine gute Variante ist das Verwenden von PDFs.

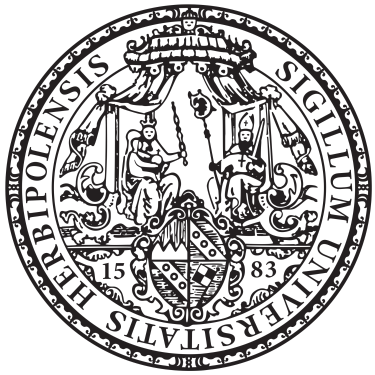


Figure 1: Siegel der Universität

8 Tabellen

Die Tabular-Umgebung gibt die Anzahl Spalten an, deren Orientierung, Breite und evtl. Zwischenlinien.

9 Formeln

$$\sum_{i=1}^{N} x_i \tag{1}$$

Table 4: Meine Tabelle

col1	col2	col3
Multiple row	cell2 cell5 cell8	cell3 cell6 cell9

References

Bommasani, Rishi et al. (2022). *On the Opportunities and Risks of Foundation Models*. arXiv: 2108.07258 [cs.LG]. URL: https://arxiv.org/abs/2108.07258.

Boniol, Paul, Qinghua Liu, Mingyi Huang, Themis Palpanas, and John Paparrizos (2024). *Dive into Time-Series Anomaly Detection: A Decade Review*. arXiv: 2412.20512 [cs.LG]. URL: https://arxiv.org/abs/2412.20512.

Bundesamt für Sicherheit in der Informationstechnik (2025). https://www.bsi.bund.de/EN/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/DoS-Denial-of-Service/dos-denial-of-service_node.html. Accessed: 2025-05-10.

Chalapathy, Raghavendra and Sanjay Chawla (2019). *Deep Learning for Anomaly Detection: A Survey*. arXiv: 1901.03407 [cs.LG]. URL: https://arxiv.org/abs/1901.03407.

Deutsche Börse (2025). https://www.deutsche-boerse-cash-market.com/dbcm-en/about-us/organisation-of-the-fwb/market-surveillance-in-germany/market-surveillance-21856?frag=249060. Accessed: 2025-05-10.

Fouladi, Ramin Fadaei, Orhan Ermiş, and Emin Anarim (2020). "A DDoS attack detection and defense scheme using time-series analysis for SDN". In: *Journal of Information Security and Applications* 54, p. 102587. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2020.102587. URL: https://www.sciencedirect.com/science/article/pii/S2214212620307560.

Guresen, Erkam and Gulgun Kayakutlu (2011). "Definition of artificial neural networks with comparison to other networks". In: *Procedia Computer Science* 3. World Conference on Information Technology, pp. 426–433. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2010.12.071. URL: https://www.sciencedirect.com/science/article/pii/S1877050910004461.

Ho, Thi Kieu Khanh, Ali Karami, and Narges Armanfard (2025). *Graph Anomaly Detection in Time Series: A Survey*. arXiv: 2302.00058 [cs.LG]. URL: https://arxiv.org/abs/2302.00058.

Huijse, Pablo, Pablo A. Estevez, Pavlos Protopapas, Jose C. Principe, and Pablo Zegers (2014). "Computational Intelligence Challenges and Applications on Large-Scale Astronomical Time Series Databases". In: *IEEE Computational Intelligence Magazine* 9.3, pp. 27–39. DOI: 10.1109/MCI.2014.2326100.

Jacob, Vincent, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul (2021). *Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series*. arXiv: 2010.05073 [cs.LG]. URL: https://arxiv.org/abs/2010.05073.

Jiang, Aofan, Chaoqin Huang, Qing Cao, Yuchen Xu, Zi Zeng, Kang Chen, Ya Zhang, and Yanfeng Wang (2024). *Anomaly Detection in Electrocardiograms: Advancing Clinical Diagnosis Through Self-Supervised Learning*. arXiv: 2404.04935 [cs.CV]. URL: https://arxiv.org/abs/2404.04935.

Lai, Kwei-Herng, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu (2021). *Revisiting Time Series Outlier Detection: Definitions and Benchmarks*. Ed. by J. Vanschoren and S. Yeung. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/ec5decca5ed3d6b8079e2e7e7bacc9f2-Paper-round1.pdf.

Lavin, Alexander and Subutai Ahmad (Dec. 2015). "Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark". In: *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE. DOI: 10.1109/icmla.2015.141. URL: http://dx.doi.org/10.1109/ICMLA.2015.141.

Liu, Qinghua and John Paparrizos (2024). "The elephant in the room: Towards a reliable time-series anomaly detection benchmark". In: *Advances in Neural Information Processing Systems* 37, pp. 108231–108261.

Paparrizos, John, Paul Boniol, Themis Palpanas, Ruey S. Tsay, Aaron Elmore, and Michael J. Franklin (July 2022). "Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection". In: *Proc. VLDB Endow.* 15.11, pp. 2774–2787. ISSN: 2150-8097. DOI: 10.14778/3551793.3551830. URL: https://doi.org/10.14778/3551793.3551830.

Paparrizos, John, Yuhao Kang, Paul Boniol, Ruey S Tsay, Themis Palpanas, and Michael J Franklin (2022). "TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection". In: *Proceedings of the VLDB Endowment* 15.8, pp. 1697–1711.

Schmidl, Sebastian, Phillip Wenig, and Thorsten Papenbrock (2022). "Anomaly Detection in Time Series: A Comprehensive Evaluation". In: *Proceedings of the VLDB Endowment (PVLDB)* 15.9, pp. 1779–1797. DOI: 10.14778/3538598.3538602.

Visa Acceptance Solutions (2025). https://www.visaacceptance.com/en-us/solutions/ai-driven-fraud-management.html. Accessed: 2025-05-10.

Wu, Renjie and Eamonn J Keogh (2021). "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress". In: *IEEE transactions on knowledge and data engineering* 35.3, pp. 2421–2429.

Yahoo (2015). https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70. Accessed: 2025-05-20.

Zamanzadeh Darban, Zahra, Geoffrey I. Webb, Shirui Pan, Charu Aggarwal, and Mahsa Salehi (Oct. 2024). "Deep Learning for Time Series Anomaly Detection: A Survey". In: *ACM Computing Surveys* 57.1, pp. 1–42. ISSN: 1557-7341. DOI: 10.1145/3691338. URL: http://dx.doi.org/10.1145/3691338.

Zhou, Xun, Cheng, Sicong, Zhu, Meng, Guo, Chengkun, Zhou, Sida, Xu, Peng, Xue, Zhenghua, and Zhang, Weishi (2018). "A state of the art survey of data mining-based fraud detection and credit scoring". In: *MATEC Web Conf.* 189, p. 03002. DOI: 10.1051/matecconf/201818903002. URL: https://doi.org/10.1051/matecconf/201818903002.

A Anhang A

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.
Ich erkläre weiterhin, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Würzburg, den 11. Juni 2025 VORNAME NACHNAME