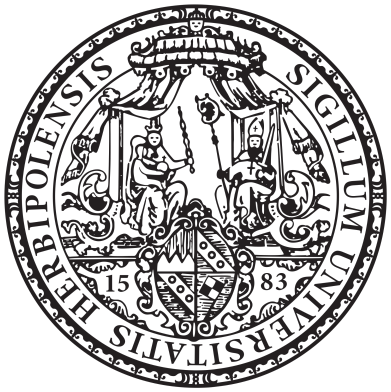
Time Series Anomaly Detection Benchmarking

Philip Spaier

3110375



Seminar Thesis

Lehrstuhl für Wirtschaftsinformatik und Künstliche Intelligenz im Unternehmen
Universität Würzburg

Supervisor: Prof. Dr. Gunther Gust
Assistant: Viet Nguyen

Würzburg, June 18, 2025

Contents

List of Figures

List of Tables

Abstract

The following seminar thesis provides four contributions to the current state of Time Series Anomaly Detection (TSAD) benchmarking. First, it includes a detailed literature review, explaining time series data, anomalies, detection methods, their use cases, and an overview of the current set of datasets and benchmarking tools. Secondly, a qualitative analysis of the Numeta Anomaly Benchmark (NAB) reinforces the commonly found claim that previous benchmarking datasets are highly flawed. Thirdly, I will replicate the TSB-AD benchmark and compare the authors' results with mine, looking at consistency between runs. Lastly, I provide four labeled high quality datasets that highlight a critical blindspot in current models' ability to detect non-peak anomalies. These require a good understanding of the entire time series, as opposed to just short local context windows.

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.

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 four key perspectives:

- Degree of supervision
- Architecture
- Technique
- Dimensionality

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).

1.3.4 Dimensionality

Detection methods can be classified as either *univariate* or *multivariate*. Univariate models can only use data with two dimensions. These are a value and a corresponding timestamp. Multivariate models can process data with multiple distinct values for a given time series. An example of a univariate time series would be a dataset recording CPU utilization over a specified period of time. A multivariate dataset could also include additionally recorded values for memory and GPU utilization or total electricity consumption.

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 Classification 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. Furthermore, 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 univariate 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. The dataset curation process systematically addresses issues with flawed time-series anomaly detection datasets through a three-step pipeline: (i) extensive collection of time-series data, (ii) flaw identification to exclude problematic series via manual inspection and anomaly detection algorithms, and (iii) label quality assessment to ensure high-quality labeling. 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 (Liu and Paparrizos, 2024, pp. 9-10; Paparrizos, Kang, et al., 2022, p. 1706).

2 Methodology

2.1 Analysis of Existing Datasets

Considering how much of the new benchmarking research is based on the assumption that previous datasets are flawed, I have conducted a short qualitative evaluation of the dataset quality found in the Numenta Anomaly Benchmark NAB (Lavin and Ahmad, 2015). As a general guideline, I will be referencing Wu and Keogh, 2021 and their described categories of issues. As a reminder, these are:

- Triviality

- Unrealistic Anomaly Density
- Mislabeled Ground Truth
- Run-to-failure Bias

I have chosen the NAB for this analysis, as it is commonly cited by many to be one of the earlierst attempts at creating a framework for TSAD benchmarking and includes a variety of different datasets. I will vizually inspecting all provided time series and select some examples representing the above mentioned flaws.

2.2 Replicating TSB-AD Benchmarking

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 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.

2.3 Dataset Creation

A common hurdle cited among 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.

2.3.1 Baseline Load and Anomaly Injection

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 continuous matrix multiplications, randomly selecting a matrix of size 3000, 3500, 4500, or 5000 and performing 3 multiplications of the selected size. This is followed by a sleep period of 4 seconds before another set of matrix multiplication resumes. Such a sequence of multiplications followed 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.

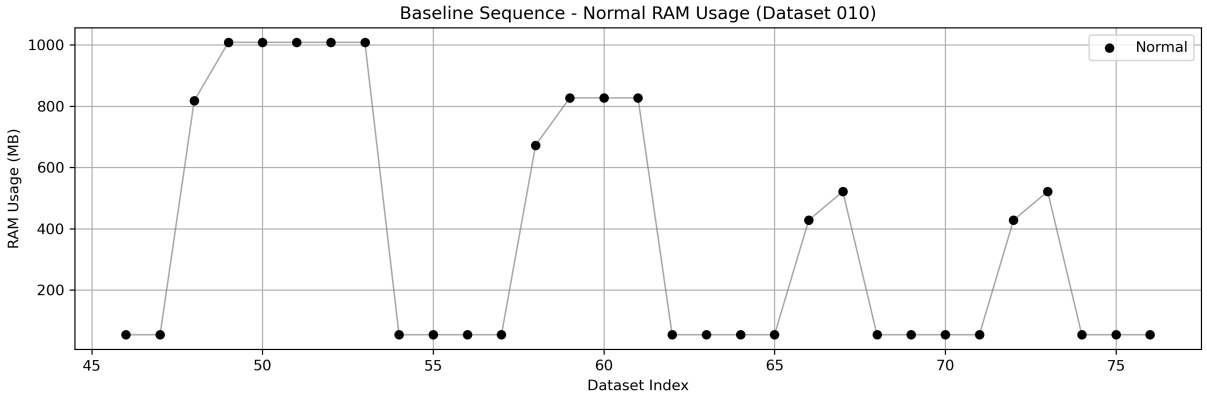


Figure 1: Baseline Load

Dataset 1 - Large Matrix Injection: For the first dataset, a particularly 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 idle 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

While being moderately easy to solve in theory, it remains a crucial scenario in many industries and fields. It simulates a sudden, unexpected resource demand (a "shock event"), common in DoS attacks or runaway processes.

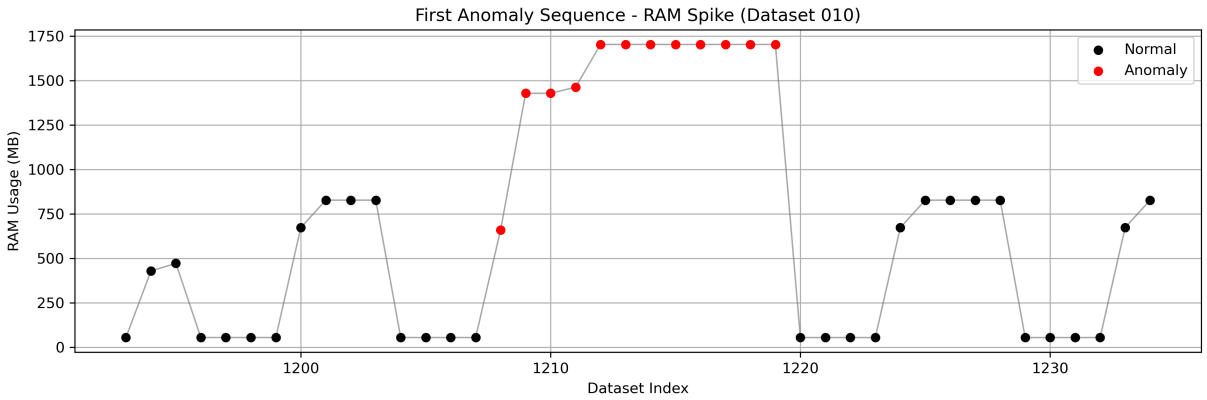


Figure 2: Large Matrix Injection

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

This anomaly type simulates a process "hanging" or an unexpected period of system inactivity, which is a critical but non-peak anomaly often missed by simple thresholding.

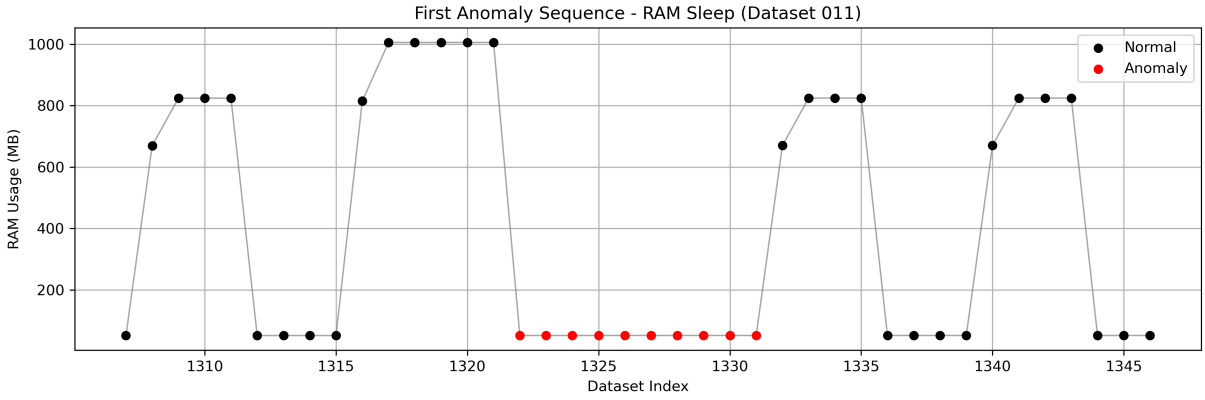


Figure 3: Sleep Time Injection

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

The third anomaly type simulates a subtle but abnormal change in workload. This would not activate peak alerts but still deviates from the established operational rhythm. This tests a model's ability to understand the normal operating range in a larger context. A real world example could be the occasional execution of malicious code, designed to hide between regular activities, never exceeding peak values in order to avoid detection.

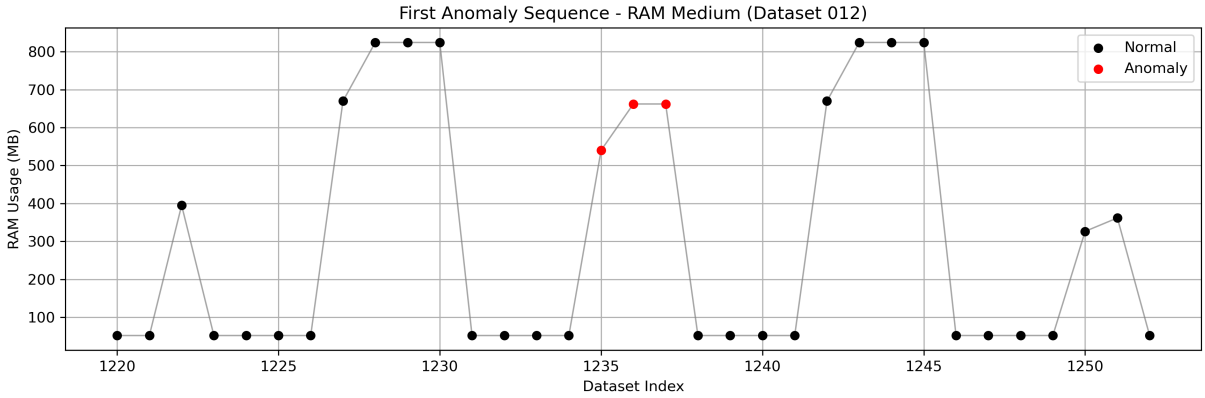


Figure 4: Medium Matrix Injection

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 idle 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

With all three anomalies representing distinctly different scenarios that require unique detection methods, a time series that challanges a model to detect all of them punishes methods that are highly specialized for a specific anomaly type.

While these datasets are based on RAM usage, the underlying principle of creating a baseline workload and injecting distinct, peak and non-peak anomalies is highly transferable to other domains like network traffic, CPU utilization, or sensor data.

2.3.2 Logging

With real data, the biggest challenge is not to find extensive datasets, it's to find datasets with accurate labels. Without those, (semi-)supervised models cannot function and no models can 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 accurate 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) continuously 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 utilization has yet occurred. The 0.3 seconds delay ensures that the first datapoint after a workload has started is correctly labeled as "working". This is particularly important for when an anomaly has been injected. Similarly, when it detects 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 occur, the logger takes a snapshot every 1.0 seconds. The 0.3 seconds and 0.1 seconds were 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.

2.3.3 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

3 Results

3.1 Flaws in Nunmeta Anomaly Benchmark (NAB)

In all four figures, the red dots are data points labeled as anomalies.

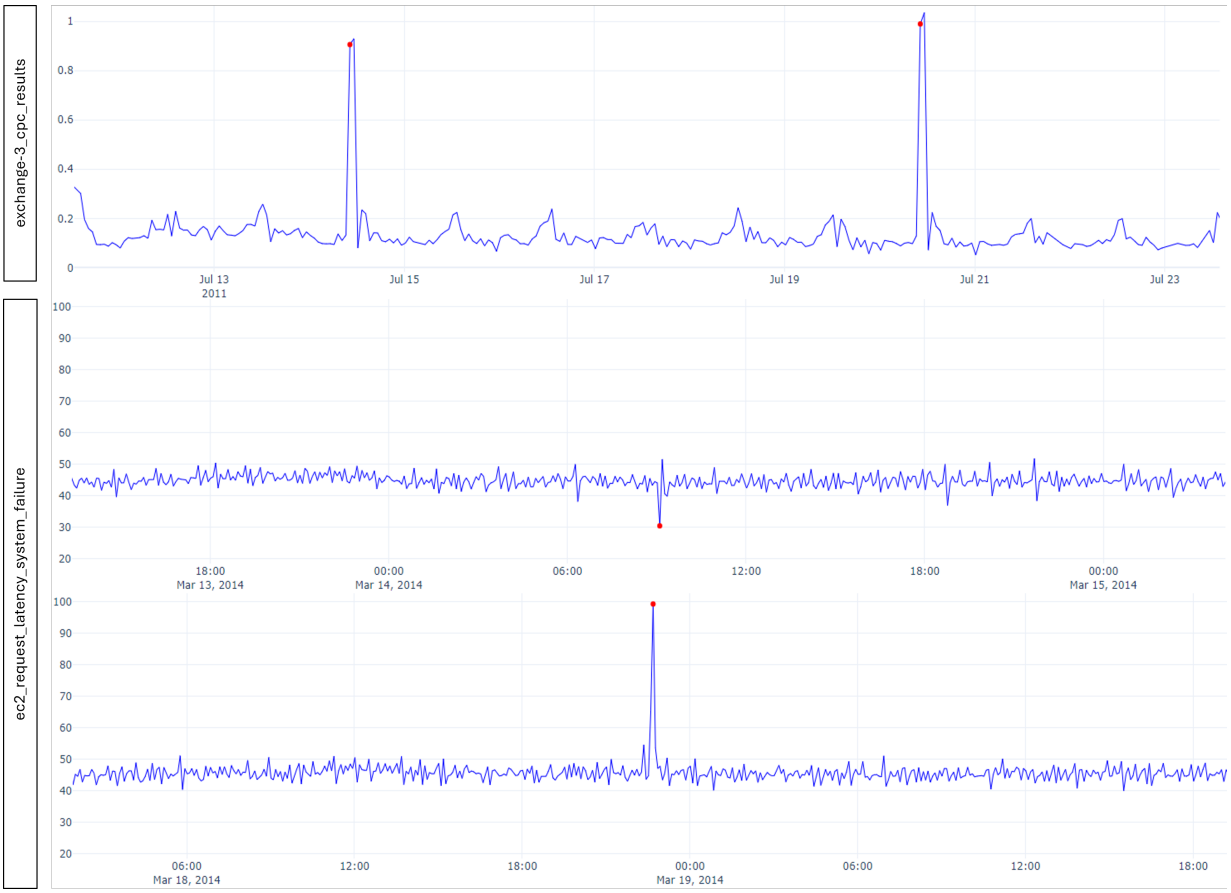


Figure 5: NAB Trivial

A sizable portion of all anomalies found in the NAB dataset fit into the category of **trivial anomalies** and can be detected using simple thresholding functions. Examples of that can be found in Figure 5.

Equally common are **mislabeled anomalies**. Examples can be found in Figure 6. The dataset at the time labels two points as anomalies, while completely ignoring the obvious peaks, which are unusual even when comparing with the entirety of the dataset. The time series at the bottom only highlights one seemingly normal datapoint, while again ignoring the unusual peak directly to its right.

A subcategory of mislabeled anomalies are sequence-based anomalies being labeled as only a single point. Examples can be found in Figure 7. In the upper time series only the first point of an unusual flatline-like sequence is labeled as an anomaly. In the lower

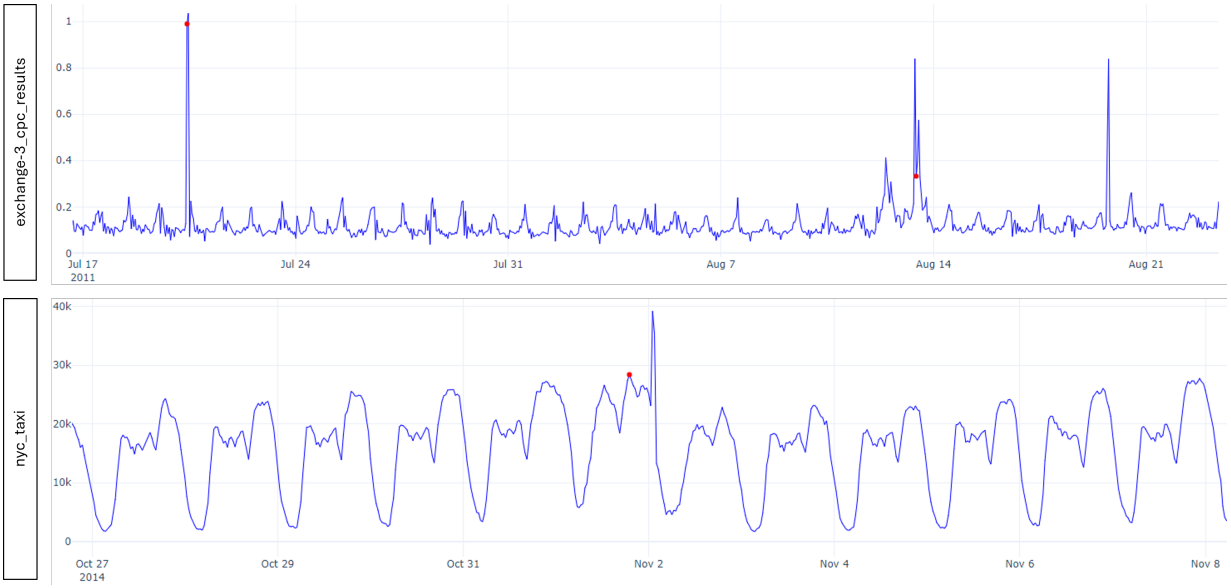


Figure 6: NAB Mislabeled

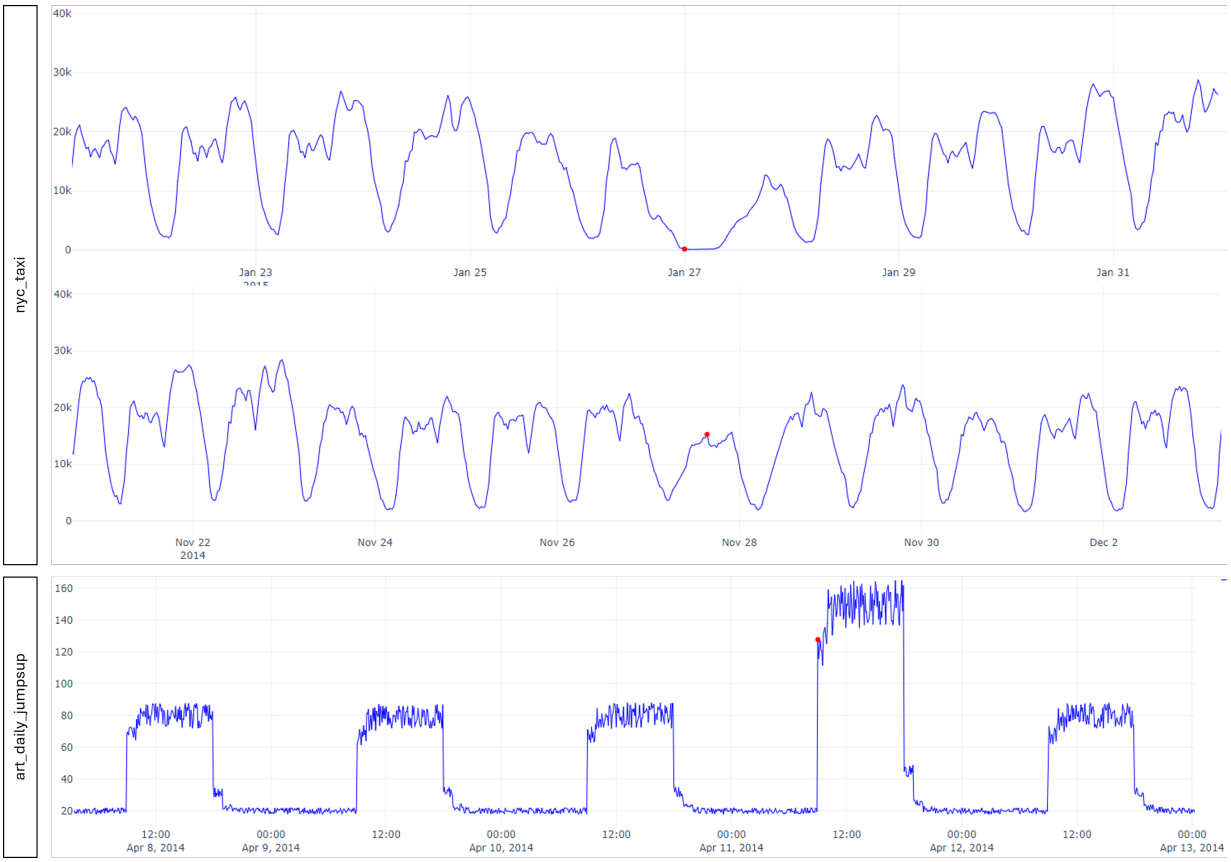


Figure 7: NAB Mislabeled | Point Instead of Sequence

time series only the first point of an elevated sequence is labeled as an anomaly. The time series in the middle highlights only a single point out of an entire sequence as an anomaly. Without context, it is hard to understand why the rest of the sequence should be considered as normal and only this one specific point is anomalous. In such cases, Wu and Keogh, 2021, p. 4-5 suggests to label the entire sequence as an anomaly.

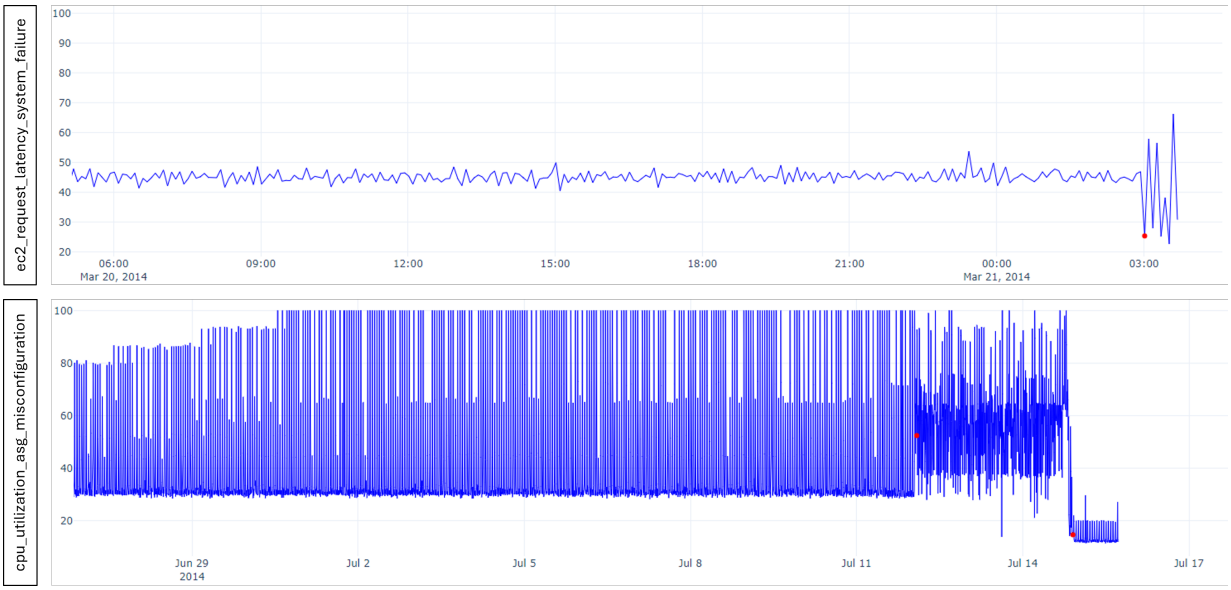


Figure 8: NAB Run-To-Failure Bias

Not as commonly found as the previous two flaws but still present are datasets with a **run-to-failure bias**, as described by Wu and Keogh, 2021. Examples of those can be found in Figure 8. In both cases, the time series end on an unusual pattern, representing hardware failures before the systems stop being monitored, These are rightfully marked as anomalies but can introduce a bias towards models that simply flag the end of each time series as an anomaly.

3.2 Consistency Between TSB-AD 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.

3.2.1 Univariate Results

Table 2 provides information about the differences found between both benchmark evaluations for univariate detection models.

For most models, the average difference stayed firmly below 0.100, with (Sub-)IForest, Sub-LOF, LSTMAD, and USAD being the main exceptions. Beyond these averages, almost half of all models had at least one dataset where the absolute score differed by more than 0.950.

Model	Avg. Abs. Diff.	Max. Abs. Diff.	File ID	Num. of Anom.	Datasets >5% Diff.	Datasets >25% Diff.	Datasets >50% Diff.
Sub-IForest	0.231	0.966	405	1	210	115	67
Sub-LOF	0.177	0.960	603	1	208	89	38
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
KMeansAD	0.012	0.268	419	1	21	1	0
Sub-MCD	0.009	0.316	788	1	20	1	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
Sub-KNN	0.002	0.216	538	3	3	0	0
POLY	0.001	0.288	534	1	2	1	0
Sub-PCA	0.001	0.071	534	1	1	0	0
Sub-HBOS	0.001	0.094	531	1	1	0	0
Sub-OCSVM	0.001	0.082	534	1	2	0	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.

Furthermore, 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 exceptions here were (Sub-)IForest and Sub-LOF. Their inconsistency were somewhat surprising, especially given that (Sub-)IForest were configured with (150) 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. The dataset #534 appeared 4 times. Multiple datasets appeared twice as an entry for the highest absolute difference. These are: #849, #865, #603, #579.

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 detection models are in many ways similar to those of univariate detection 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 Univariate and Multivariate Compared

Across both univariate and multivariate anomaly detection models, the observed trends regarding benchmark evaluation consistency are largely parallel. While most models displayed relatively small average differences between the two evaluations, a significant por-

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
AnomalyTrans	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.

tion 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.

3.3 Results From Novel Datasets

3.3.1 Statistical Overview

	Dataset 1	**Dataset 2**	**Dataset 3**	**Dataset 4**
Anomaly Type	Large Matrix	Added Sleep	Medium Matrix	Combined
Length	5308	5320	5319	9452
Total Anomaly Sequences	8	11	11	21
Total Anomalous Points	91	110	40	194
Average Anomaly Length	11.38	10.0	3.64	9.24
Longest Anomaly	13	10	4	16
Shortest Anomaly	10	10	3	4
Anomaly Ratio (Point based)	0.0171	0.0207	0.0075	0.0205
Anomaly Ratio (Sequence based)	0.0015	0.0021	0.0021	0.0022
RC Score	473.079	505.050	395.150	597.475

Anomaly Ratio (Point based): Total Anomalous Points divided by Length
Anomaly Ratio (Sequence based): Total Anomalous Sequences divided by Length

Table 4: Statistical summery of datasets 1-4

Most datasets used for the TSB-AD benchmark are between 1000 and 50000 datapoints in length. Based on this, I decided to keep datasets 1-3 at around 5000 datapoints. This ensures a sufficiently large sample size, while keeping it small enough for me to manually go through the entire set and check for potential problems or mislabeled datapoints.

Since dataset 4 contains 3 separate anomaly types and I intended to maintain a consistent anomaly ratio between all sets, I increased its size. Otherwise, when maintaining the same anomaly ratio, it would be possible for some anomaly type to not occur at all. Schmidl, Wenig, and Papenbrock, 2022, pp. 8 define an acceptable anomaly ratio to be one below 10%, as anything above it would be better considered to be a classification problem. The median anomaly ratio in the TSB-AD dataset is 4.85% and Wu and Keogh, 2021, p. 4 consider the adequate number of anomalies to be exactly 1. Given his range of opinions, I settled on a range of between 0.7% and 2.2% for my datasets. The differences between datasets can be explained by the average length of an anomalous sequence depending on anomaly type. The actual anomaly ratio based on anomalous sequences to total sequences is kept steady between datasets. Maintaining the sequence ratio ensures each dataset presents a consistent number of anomalous events to detect, which is more important for a fair comparison than the point ratio that can be skewed by the varying lengths of those events.

For the calculations of the difficulty measures, I used Relative Contrast (RC), as described by Paparrizos, Kang, et al., 2022, pp. 1704. Since the authors did not specify their preferred Shape Based Distance (SBD) metric, I applied the commonly used Dynamic Time Warping (DTW), using the cdist_dtw implementation provided by the tslearn library (Tavenard et al., 2020). Paparrizos, Kang, et al., 2022, pp. 1704 do not clearly state how to decide on a sliding window for the comparison of individual sequences and only state to "split the time series by its period" (Paparrizos, Kang, et al., 2022, p. 1704). I interpret the term "period" to mean a repeating pattern, as is common in time series analysis. With all 4 of my created datasets, a repeating period can be defined as a sequence of matrix multiplications followed by an idle state, which tends to be between 8 and 12 datapoints long. With that in mind, I selected the value 10 for my sliding window. Since I neither know the exact RC implementation of Paparrizos, Kang, et al., 2022, nor their used SBD metric, it is impossible to compare the here produced values with those provided in the TSB-UAD paper. Therefore, a comparison can only be made between my own datasets. A low RC score should indicate that its difficult to effectively cluster the points in a dataset, making anomaly detection theoretically more challenging. Based on this assumption, dataset 4 should provide the easiest to detect anomalies. As will be shown later, this does not hold up.

3.3.2 Benchmarking Results on New Dataset

Table 5 provides a summary of results when evaluating datasets 1-4 with the TSB-AD benchmark. Unsurprisingly, dataset 1, which essentially just has a larger peak load as its anomalies, had the best results. Sub-PCA managed to achieve almost perfect scores for both VUS-PR (0.989) and VUS-ROC (0.999), highlighting its ability to detect the existing anomalies precisely and with great sensitivity. Most other models, statistical and neural network-based alike, also performed decently well with average scores (VUS-ROC) of 0.887 and 0.767 respectively. When looking at VUS-PR score averages, most detection models did not reach a score of above 0.500, with neural-network based models surprisingly outperforming statistical models. Since all anomalies are sequence-based and not point-based, one would expect the opposite (Liu and Paparrizos, 2024, p. 10; Paparrizos, Kang, et al.,

	Dataset 1	**Dataset 2**	**Dataset 3**	**Dataset 4**
Anomaly Type	Large Matrix	Added Sleep	Medium Matrix	Combined
VUS-PR				
Best Model	Sub-PCA	KShapeAD	LOF	OmniAnomaly
Best Score	0.989	0.562	0.062	0.394
Avg. Score	0.376	0.096	0.018	0.084
Avg. Score (St. Mod.)	0.253	0.144	0.024	0.062
Avg. Score (NN Mod.)	0.472	0.059	0.014	0.108
VUS-ROC				
Best Model	Sub-PCA	SAND	LOF	OmniAnomaly
Best Score	0.999	0.949	0.926	0.878
Avg. Score	0.832	0.586	0.554	0.615
Avg. Score (St. Mod.)	0.887	0.572	0.634	0.613
Avg. Score (NN Mod.)	0.767	0.604	0.506	0.659

Table 5: VUS-PR and VUS-ROC: Summary of benchmarking results for datasets 1-4.

2022, p. 1706).

For dataset 2, which introduces anomalies by extending the idle time, the results are more varied. The best VUS-ROC score was a high 0.949 achieved by SAND, indicating a strong ability to detect these anomalies. However, the best VUS-PR score was only 0.562 from KShapeAD, showing that while models can find the anomalies, they struggle with precision. The average VUS-ROC scores were middling, with neural network models (0.604) slightly outperforming statistical ones (0.572). In contrast, the average VUS-PR scores were poor across the board, with statistical models (0.144) performing better than neural network models (0.059).

Dataset 3, featuring the 'Medium Matrix' anomalies, was clearly the most difficult for the benchmarked models. This is especially evident in the VUS-PR scores, which were exceptionally low. The LOF model performed best on both metrics, yet it highlights a significant performance gap: it achieved a strong VUS-ROC score of 0.926 but a dismal VUS-PR score of only 0.062. On average, VUS-ROC scores were moderate, with statistical models (0.634) performing noticeably better than neural network models (0.506). This trend was even more pronounced for the VUS-PR scores, where the average for all models was a near-zero 0.018, confirming the difficulty of this dataset. It is also the only dataset where the best VUS-PR and VUS-ROC score where not achieved by the same model.

Dataset 4, which aggregates all anomaly types, was intended to be the most challenging. The neural network-based OmniAnomaly model achieved the best scores for both VUS-PR (0.394) and VUS-ROC (0.878). While the VUS-ROC score demonstrates good overall sensitivity, the sub-0.400 VUS-PR score again points to a significant challenge in maintaining

precision across varied anomaly types. The average VUS-ROC scores were the highest outside of dataset 1, with neural network models (0.659) holding a slight edge over statistical models (0.613).

These results were verified with 2 additional benchmarking runs. The scores remained almost identical throughout all three runs.

4 Discussion

4.1 The Need For Better Benchmarking Tools

The examples shown in 3.1 are not cherry-picked results; they represent systemic flaws found throughout the entire NAB dataset. The analysis consistently reveals two main problems. The first is the large number of trivial anomalies that can be detected with simple thresholds. The second is significant mislabeling of the ground truth, where entire unusual sequences are often marked as just a single anomalous point. Interestingly, these labeling errors are not just in the real-world data but also appear in some of the artificially created time series. This suggests a potential flaw in how the benchmark was fundamentally designed and labeled. This brief analysis therefore confirms the critiques made by previous authors. It supports their claims that many of the early TSAD datasets are compromised, limiting their usefulness for rigorously comparing modern anomaly detection algorithms. The evidence here validates their conclusion that there is a clear need for new benchmarking tools built on a foundation of accuracy and realistic complexity.

4.2 Benchmark Reproducability

Beyond the issue of dataset quality, the replication of the TSB-AD benchmark revealed another challenge in the field: reproducibility between different systems. The results showed that while most models were consistent on average, their scores could differ significantly on specific datasets. This was particularly true for neural network-based methods and models with a low anomaly density where just a few mistakes can drastically change evaluation scores. A single performance score, especially on a new or challenging dataset, should be treated with care. The randomness in some models means that poor performance might be due to a bad initialization, not just a bad model design. It also puts into queastion the claim that the ideal number of anomalies per dataset should always be exactly 1 (Wu and Keogh, 2021, p. 4). With only a single anomaly per time series, even a single mistake could result in a vastly worse model evaluation. It forces benchmarks to drastically increase the total number of time series in order to compensate for this behaviour and average out scores. Considering the high reproducibility of my own datasets between runs, the observed differences are likely to be caused by differences between systems and potentially setups.

4.3 Model Failure on Contextual Anomalies

While the quality and reproducibility of benchmarks present foundational challenges, the most critical finding of this thesis emerged from evaluating models on the newly developed datasets. The results point to a significant failure on contextual anomalies.

Other than the results for dataset 1, these scores presented in 3.3.2 are unexpectedly bad. While an average VUS-ROC score of around 0.500 for datasets 2 and 3 could potentially be deemed acceptable, when combined with the horrendous VUS-PR scores, it becomes questionable whether most of the models provided any value at all. This applies even more so to dataset 3, where not a single model was able to achieve a usable level of precision. Given that the anomalies were correctly labeled, this begs the obvious question: Were the anomalies simply too difficult to detect? In order to answer this, I have applied a modified process introduced by Wu and Keogh, 2021, who attempted to detect anomalies with only a single line of code in order to determine whether they are trivial. Since the goal here is not to look for triviality but instead an acceptable level of difficulty, I have attempted to write a simple script, made up of only basic if-statements, which can, with high accuracy and precision, detect all existing anomalies. *Dataset 2 Pseudo Code:*

```
For i from 0 to len(data)-1:
    If data[i] < 60:
        Count consecutive values < 60
        If count > 7:
            Mark those indices as anomalies
Return anomaly labels
```

When appling this simple algorithm, it returns 110 True Positives, 0 False Positives, 0 False Negatives, and therfore a precision and sensitivity of 1 are achieved.

Dataset 3 Pseudo Code:

```
For i from 0 to len(data)-1:
If data[i] in (530, 700):
    Count consecutive values in (530, 700)
    If count == 3 or 4:
        Mark those indices as anomalies
Return anomaly labels
```

Similarly here, the algorithm returns 40 True Positives, 0 False Positives, 0 False Negatives, and therefore a precision and sensitivity of 1.

This is obviously not a perfect test, as it requires me to have prior knowledge about the nature of the anomaly. Despite that, if an algorithm of such simplicity can detect anomalies with complete precision and sensitivity, I will argue, that any other sufficiently advanced detection method should be able to at least reach a VUS-PR score of above 0.1 to be deemed functional.

The anomaly ratios (<3%) are firmly within the established consensus for what an acceptable ratio is, the datapoints are accurately labeled with the help of an automated logger and a manual inspection, the anomalies represent real changes in workload, and the anomalies can be easily detected using a few lines of code. Based on these aspects I am willing to state that the results do not represent poor quality of the datasets but instead a blindspot in the current set of detection models included by the TSB-AD benchmark.

Since Dataset 4 is made up two-thirds of the anomalies established in Datasets 2 and 3, it is only natural that the average scores are also rather unimpressive and lie somewhere between those of Aataset 1 and Dataset 4. The inclusion of multiple anomaly types seems to have slightly degraded peak VUS-ROC performance compared to all other datasets. This is most likely the result of highly specialized models that are good at detecting certain anomalies (e.g. peak values), struggling more with other types of anomalies. A neural network-based model performing best in both metrics is another notable occurrence.

4.4 Implications for TSAD Research

The results indicate that current models struggle with non-peak anomalies, especially those which mimic known activities but at unusual levels between the otherwise present minimum and maximum values. As previously stated, such anomalies could appear in cases of fraud, where malicous activities are designed to avoid standing out. The injected anomalies in dataset 3 can be easily detected with a few lines of code or human evaluation. Despite that, most models struggled with their sensitivity, and all models failed entirely to showcase any acceptable level of precision. In real cases of fraud, attackers would most likely do a better job than to inject malicous acitivities conveniently placed at a level exactly between already existing processes and instead aim to replicate output levels more closely. If existing models failed on my simple simulations, they would certainly fail in more sophisticated scenarios.

To address this, future detection methods must be improved to better incorporate the global context of a time series. However, for such model development to be effective, I propose that a more nuanced dataset categorization is needed. Instead of only classifying anomalies as point-wise or sequence-wise, it would be beneficial to create an extended framework that also defines the contextual nature of the anomaly. This framework could include primary categories like 'Deviation from Local Pattern' and 'Deviation from Global Pattern,' with subcategories such as the types explored in this thesis: 'Peak-Value,' 'Medium-Value,' and 'Low-Value' anomalies. Adopting such a taxonomy would allow researchers to identify which anomaly classes their models struggle with and enable targeted improvements.

4.5 Limitations

The broad scope of this thesis required a breadth-focused approach, meaning an in-depth exploration of every subtopic was not possible.

Specifically, the analysis of existing datasets was qualitative and limited to a single bench-

mark suite (NAB). This means my findings might be biased or not fully represent the entire body of existing work. Furthermore, the TSB-AD benchmark replication was performed only once. With such a small sample size, the results may not be statistically significant and could be due to chance. These runs were also done on consumer hardware in a non-isolated environment, which might have affected the outcomes. Additionally, minor unrecognized differences in implementation or updates to the used third party packages and libraries could have caused some of the observed differences.

For the dataset creation, two main limitations exist. First, despite efforts to isolate the simulation environment using a Virtual Machine and disabling background processes, undetected effects from the host hardware are still possible. A completely isolated system, like one running Linux From Scratch, would offer more control but requires extensive specialized knowledge. However, visual inspection suggests no significant data quality issues, likely due to the stable nature of memory allocation. Second, the anomalies themselves are simulations. While designed to be challenging, they only roughly represent real-world issues. Achieving exact replications of naturally occurring anomalies would require injecting actual malicious code or inducing real hardware failures.

5 Conclusion

This seminar paper embarked on a comprehensive evaluation of the current state of Time Series Anomaly Detection (TSAD) benchmarking, navigating from a literature-driven critique to empirical analysis and contribution. The literature review established a clear picture: while time series anomaly detection is an established discipline in many industries, playing an essential role in areas such as fraud detection and healthcare, the openly published scientific progress has only recently picked up pace. The lack of a generally recognized benchmarking environment with labeled high quality datasetshas posed a significant challenge to the development and evaluation of better detection methods. While eariler datasets like the Yahoo S5 dataset or the Numeta Anomaly Benchmark (NAB) provided a decent starting point, substantial issues with triviality, label quality, and run-to-failure bias have highlighted the need for better benchmarks. TSB-AD, and its predecessor TSB-UAD, have persued this goal and created a new standard for anomaly detection benchmarking that can be expanded on.

The attempt to replicate the results provided by the TSB-AD benchmark has shown that many hurdles exist. In particular, running all provided models turned out to be a challenging endevour. For researchers with a more traditional background without strong programming foundations, this could become roadblock in the advancement of TSAD methods. Consistency between the original results an my run has been on average high but it is important to be careful when evaluating model performance on datasets with a low number of anomalies. The results from statisitcal model tend to be more reproducable than those of neural network-based ones.

The central contribution of this paper was the creation of four new, high-quality, real-world univariate datasets, designed specifically to address the identified flaws of previous benchmarks. By meticulously logging RAM usage during controlled workload scenarios, I produced accurately labeled datasets with a justifiable anomaly ratio and distinct, non-trivial anomaly types. A noteworthy result emerged from benchmarking established algorithms on these new datasets. While models performed expectadly well on the 'Large Matrix Injection' anomalies, which present as clear statistical outliers, their performance degraded dramatically on the more subtle 'Medium Matrix Injection' and 'Sleep Time Injection' tasks. Average VUS-PR scores, in particular, reached unjustifiably low levels, indicating a near-total failure to precisely identify these anomalies, which were, by contrast, perfectly detectable with simple, rule-based scripts.

This suggest a blindspot in the current set of commonly discussed detection models. Their design may be overly specialized for detecting point-wise outliers (which could be explained by all the sequence-wise anomalies mislabeled as point-wise anomalies in earlier datasets like NAB) or significant deviations from a localized temporal window, while struggling with contextual anomalies that are only identifiable by understanding the broader patterns of the entire time series. The models failed to recognize that a period of inactivity or a moderately-sized workload, while normal-looking in isolation, was anomalous within the established operational rhythm of the system.

Ultimately, this thesis gives an overview of the current Time Series Anomaly Detection (TSAD) field and existing benchmarking methods. Further more, it provides four high quality datasets that highlight a critical blindspot in current detection methods. These datasets serve as a useful tool for future model improvements.

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.

Tavenard, Romain, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods (2020). "Tslearn, A Machine Learning Toolkit for Time Series Data". In: *Journal of Machine Learning Research* 21.118, pp. 1–6. URL: http://jmlr.org/papers/v21/20-091.html.

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 Full Results of Benchmarking the New Datasets

Table 6: Model Performance Comparison on 010_RAMspike_10_Hardware_tr_1200_1st_1210

Model	AUC-PR	AUC-ROC	VUS-PR	VUS-ROC
AnomalyTransformer	0.0171	0.4753	0.0221	0.4767
AutoEncoder	0.0099	0.1797	0.0119	0.1660
CNN	0.9465	0.9583	0.9475	0.9766
FFT	0.5835	0.9072	0.4995	0.9379
FITS	0.0318	0.5354	0.0375	0.5955
IForest	0.1081	0.9153	0.1322	0.9003
KMeansAD_U	0.1211	0.9174	0.1433	0.9022
KShapeAD	0.2319	0.9419	0.2119	0.9437
Left_STAMPi	0.0224	0.6119	0.0304	0.5943
LOF	0.0310	0.6006	0.0427	0.6669
LSTMAD	0.7009	0.9697	0.7084	0.9759
M2N2	0.9462	0.9683	0.9376	0.9877
MatrixProfile	0.1077	0.8046	0.1544	0.8910
MOMENT_ZS	0.1874	0.9628	0.2233	0.9670
OmniAnomaly	0.9791	0.9996	0.9866	0.9998
POLY	0.7869	0.9954	0.8133	0.9968
SAND	0.0614	0.8385	0.0743	0.8612
SR	0.4325	0.9028	0.4522	0.9378
Sub_HBOS	0.4562	0.8222	0.3928	0.8999
Sub_IForest	0.8427	0.9933	0.8795	0.9966
Sub_KNN	0.0648	0.8747	0.0867	0.8875
Sub_LOF	0.1619	0.9526	0.2121	0.9530
Sub_MCD	0.0108	0.2701	0.0132	0.2595
Sub_OCSVM	0.0648	0.7716	0.0749	0.7485
Sub_PCA	0.9883	0.9998	0.9893	0.9999
TranAD	0.8239	0.9961	0.8913	0.9976
USAD	0.1291	0.9453	0.1746	0.9462

Table 7: Model Performance Comparison on 011_RAMsleep_11_Hardware_tr_1200_1st_1324

Model	AUC-PR	AUC-ROC	VUS-PR	VUS-ROC
AnomalyTransformer	0.0217	0.4882	0.0269	0.4826
AutoEncoder	0.0315	0.6456	0.0387	0.6871
CNN	0.0173	0.4475	0.0220	0.4622
FFT	0.0207	0.3123	0.0260	0.3319
FITS	0.0178	0.4364	0.0234	0.5103
IForest	0.0132	0.2683	0.0166	0.2112
KMeansAD_U	0.0629	0.6214	0.0485	0.5825
KShapeAD	0.5849	0.9209	0.5619	0.9445
Left_STAMPi	0.0583	0.7618	0.0702	0.7498
LOF	0.0239	0.4362	0.0272	0.4533
LSTMAD	0.0212	0.5416	0.0274	0.6055
M2N2	0.0184	0.4805	0.0220	0.5269
MatrixProfile	0.0216	0.4504	0.0244	0.4467
MOMENT_ZS	0.0148	0.3359	0.0178	0.2944
OmniAnomaly	0.2378	0.8521	0.2667	0.8712
POLY	0.0164	0.3716	0.0211	0.3459
SAND	0.5052	0.9371	0.4225	0.9487
SR	0.0258	0.5872	0.0325	0.6455
Sub_HBOS	0.6076	0.8314	0.5494	0.8736
Sub_IForest	0.0110	0.0892	0.0132	0.1317
Sub_KNN	0.0632	0.7234	0.0798	0.7677
Sub_LOF	0.0650	0.8238	0.0825	0.8612
Sub_MCD	0.0345	0.6672	0.0468	0.7134
Sub_OCSVM	0.0648	0.7607	0.0623	0.7934
Sub_PCA	0.0150	0.3463	0.0191	0.3698
TranAD	0.0219	0.5443	0.0310	0.6188
USAD	0.0275	0.5959	0.0362	0.5931

Table 8: Model Performance Comparison on 012_RAMmedium_12_Hardware_tr_1200_1st_1237

Model	AUC-PR	AUC-ROC	VUS-PR	VUS-ROC
AnomalyTransformer	0.0085	0.5179	0.0144	0.4915
AutoEncoder	0.0111	0.5630	0.0194	0.5603
CNN	0.0044	0.1373	0.0086	0.2994
FFT	0.0075	0.3364	0.0143	0.3396
FITS	0.0077	0.5123	0.0142	0.6153
IForest	0.0120	0.6676	0.0190	0.6548
KMeansAD_U	0.0175	0.5914	0.0229	0.5411
KShapeAD	0.0097	0.6309	0.0169	0.6655
Left_STAMPi	0.0174	0.6920	0.0276	0.6591
LOF	0.0503	0.9260	0.0621	0.9264
LSTMAD	0.0053	0.3307	0.0093	0.5000
M2N2	0.0047	0.2339	0.0088	0.4380
MatrixProfile	0.0096	0.5403	0.0282	0.6904
MOMENT_ZS	0.0057	0.3700	0.0092	0.3590
OmniAnomaly	0.0049	0.2509	0.0156	0.5425
POLY	0.0058	0.3636	0.0108	0.3504
SAND	0.0115	0.6955	0.0203	0.7291
SR	0.0053	0.3402	0.0101	0.5142
Sub_HBOS	0.0071	0.4415	0.0102	0.5115
Sub_IForest	0.0195	0.7884	0.0457	0.8525
Sub_KNN	0.0097	0.5965	0.0200	0.6327
Sub_LOF	0.0060	0.4246	0.0105	0.4209
Sub_MCD	0.0124	0.6283	0.0232	0.7032
Sub_OCSVM	0.0074	0.5224	0.0144	0.6283
Sub_PCA	0.0048	0.2216	0.0071	0.2922
TranAD	0.0133	0.7199	0.0170	0.7197
USAD	0.0064	0.3663	0.0103	0.3208

Table 9: Model Performance Comparison on 013_RAMmixed_13_Hardware_tr_2000_1st_2083

Model	AUC-PR	AUC-ROC	VUS-PR	VUS-ROC
AnomalyTransformer	0.0204	0.4819	0.0315	0.4867
AutoEncoder	0.0205	0.4877	0.0352	0.5242
CNN	0.3191	0.5636	0.0941	0.5556
FFT	0.1247	0.5532	0.0551	0.4276
FITS	0.0641	0.5869	0.0443	0.6666
IForest	0.1251	0.5930	0.0500	0.5329
KMeansAD_U	0.0243	0.5500	0.0336	0.4481
KShapeAD	0.0391	0.6980	0.0473	0.7077
Left_STAMPi	0.0351	0.6919	0.0484	0.7102
LOF	0.0177	0.4088	0.0319	0.5099
LSTMAD	0.1008	0.6478	0.0581	0.7293
M2N2	0.3219	0.6197	0.1040	0.6358
MatrixProfile	0.0735	0.6210	0.1033	0.7788
MOMENT_ZS	0.0966	0.6134	0.0761	0.5910
OmniAnomaly	0.5149	0.8398	0.3942	0.8782
POLY	0.2589	0.6070	0.1117	0.5163
SAND	0.0420	0.6928	0.0549	0.7198
SR	0.1094	0.5934	0.0673	0.6898
Sub_HBOS	0.2093	0.6745	0.1685	0.7796
Sub_IForest	0.3550	0.6704	0.1711	0.6827
Sub_KNN	0.0199	0.5114	0.0305	0.4698
Sub_LOF	0.0603	0.5957	0.0515	0.4912
Sub_MCD	0.0268	0.5209	0.0402	0.5227
Sub_OCSVM	0.0234	0.4790	0.0375	0.5490
Sub_PCA	0.3318	0.6393	0.1230	0.5795
TranAD	0.2698	0.6989	0.1359	0.7731
USAD	0.0757	0.7174	0.0746	0.6647

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 18. Juni 2025 Philip Spaier