Streamify Time Series Anomaly Detection Methods

George Matlis[a], Ioannis Baraklilis[b]

Department of Computer Science Aristotle University of Thessaloniki
Thessaloniki, Greece

[a] *gmatl@csd.auth.gr, Student ID: 178*
[b] *ibarakl@csd.auth.gr, Student ID: 181*

Abstract

This study aims to explore the development of streaming versions of offline anomaly detection methods for time series data. Time series are prevalent across various scientific disciplines and industrial applications, making the analysis of such data crucial. Among the analytical tasks associated with time series, anomaly detection is particularly challenging due to the complexity of identifying outliers and the significant impact of this task on real-world applications. The objective of this study is twofold: first, to gain comprehensive experience with state-of-the-art benchmarks, methods, and evaluation measures in the field; second, to investigate how different anomaly detection techniques can be adapted for streaming scenarios where data is continuously received in real-time. Specifically, we propose streaming variants for the Isolation Forest and Matrix Profile, two non-streaming methods, and evaluate their performance on selected time series from the TSB-UAD benchmark. Finally, we compare the two non-streaming methods with their streaming variants, as well as with the state-of-the-art SAND algorithm.

Keywords:
Anomaly Detection, time series, SAND, Isolation Forest, Matrix Profile

1. Dataset & Data Pre-Processing

For the purposes of this analysis, we utilized the TSB-UAD time series dataset [1]. We selected a subset of time series from the original dataset using the following methodology: four domains were hand-selected, with the main criterion being that they are domains as thematically dissimilar to each other as possible. Further criteria for selection was the average series length and average number of anomalies, as reported in the description of TSB-UAD in the GitHub repository[1]. The goal was to select the appropriate time series so that they contain a sufficient number of anomalies to give the anomaly detection methods ample opportunities to identify anomalies, but to avoid being too large as to aid in the efficiency of the experiments. Additionally, the selection of time series from dissimilar domains was made to aid in evaluating the performance of methods in multiple domains as well as in time series with multiple normalities created by concatenating single-normality series.

The domains selected were: **ECG**, **IOPS**, **SMD**, and **Occupancy**. For each domain, two time series were selected, and were validated by visual inspection to be similar. One was used for evaluating the efficacy of the anomaly detection methods and another was used for selecting the parameters of the methods before evaluating them, i.e., for parameter tuning. So, two similar time series datasets were created. For each, identical preprocessing operations were conducted in an attempt to have the datasets as similar as possible while them not containing the same data. The time series selected for evaluation have a "1" identifier at the end of their names, while the series meant for tuning have a "2". For example, ECG1 for evaluation and ECG2 for tuning. For the ECG time series, the whole time series selected was placed for evaluation, but a smaller variant of it, created by selecting the first 20000 samples was also evaluated as an individual time series. They are named ECG1_20k and ECG2_20k for evaluation and tuning.

Multiple normality time series were artificially selected by concatenating multiple time series from the different domains selected. For the purposes of creating multiple normality time series that include ECG data, only the ECG_20k series were considered. This is to make sure that, because the ECG series happen to be quite large, they don't dominate the resulting time series. To create double normality time series, two sets of series were concatenated: ECG_20k with IOPS and SMD with Occupancy creating ECG+IOPS and SMD+Occupancy. Similarly, to create triple normality time series, two sets of three series were concatenated: ECG_20k with IOPS and Occupancy, and SMD with ECG_20k and Occupancy creating ECG+IOPS+Occupancy and SMD+ECG+Occupancy. Finally, all time series were concatenated to create a quadruple normality series: EEG+IOPS+SMD+Occupancy.

2. Anomaly Detection & Experimental Setting

Our anomaly detection experimentation consisted of evaluating the following methods:

[1] https://github.com/TheDatumOrg/TSB-UAD?tab=readme-ov-file#benchmark

- **SAND** [2] — SAND is a novel online algorithm suitable for domain-agnostic anomaly detection. We use the implementation present in TSB-UAD [1] implemented models list.

- **Matrix Profile** — Matrix Profile (MP) is an offline unsupervised discord-based method of identifying anomalies based on the distance to their nearest neighbors [2, 3]. The algorithm implementation we used for computing the matrix profile in each case was the one present in the STUMPY[2] library.

- **Isolation Forest** [4] — The Isolation Forest (IF) is an offline unsupervised learning algorithm designed to detect anomalies by isolating observations in a dataset, leveraging the fact that anomalies are few and different, making them easier to isolate. We use the implementation present in TSB-UAD implemented models list.

All three of these methods, work on the subsequence level. So, in our experiments, we process the source time series with a single step window function to create a number of subseries which we then evaluate. To determine the size of the windows, we use each signal's period estimated using the autocorrelation function following the example of [1]. Also, for the evaluations we used the evaluation set of time series (like ECG1, SMD1, etc.).

During the base (not tuned) evaluations, we used the algorithms' default parameters. For the Matrix Profile, the (default) parameters used were k set to 1 (the anomaly scores are the distance to the closest neighbor) and the distances were normalized. For the Isolation Forest, the parameter of the number of estimators was 100 (the default). As an exception, for SAND we selected the same parameters as indicated in the SAND's usage example in the TSB-UAD repository [3]. For SAND, we set a to 0.5, subsequence_length (l_θ) to 4 times the window length, k to 6, init_length (initial batch length) to 5000, batch_size to 2000, and overlapping_rate (number points separating subsequences in the time series) to 4 times the window length.

We evaluate Matrix Profile and Isolation Forest on both offline and online settings. The offline setting was applied as the original methods intended, with access to the whole time series at once. The online settings were modifications on the way the methods have access to and evaluate the time series, without changing the way the methods themselves behave. In other words, they are method agnostic. Additionally, we evaluated SAND's performance on the same series as another baseline. Shortly, the approaches we evaluate are: 1) Offline setting, 2) Online setting Variant 1: Naive batching, 3) Online setting Variant 2: Batching with access to recent history, 4) Online setting Variants 3 and 4: Dynamic batching based on single data point characteristics.

[2] https://stumpy.readthedocs.io/en/latest/api.html#stumpy.stump

[3] https://github.com/TheDatumOrg/TSB-UAD/blob/main/example/notebooks/test_anomaly_detectors.ipynb

2.1. Offline Evaluation

As the first baseline, we apply the selected algorithms in an offline manner. This means that they have access to the subseries of the whole time series. This is the ordinarily expected usage of the algorithms we selected. Additionally, as the time series labels are defined in sample-based case, and we evaluate subseries, we pad the results on the left with the score of the first subseries and on the right with the score of the last subseries that could be generated. This assures consistent lengths of output scores and ground truth labels. This also helps to offset the center of the subseries toward the index where the label for the ground data is, reducing the lag introduced by the manufacturing of subseries with sliding windows as described in [1]. The total amount of padding is $window\_length - 1$, half in the left and half on the right of the scores.

2.2. SAND Evaluation

As the second baseline, we evaluate SAND on the selected time series. With this method, unlike the others, we did not add any padding as the algorithm ensures the consistency of length of results to the number of points in the time series.

2.3. Variant 1: Naive batching

As the first variant to be evaluated, we adapted the offline methods to online with a simple approach. In the naive streaming setting, we segment the time series data into partitions, where each holds 150 subseries. With the batches formed, we analyze each partition step-by-step and independently. This simulates a streaming scenario where data points arrive, overlapping subseries are extracted, and when a batch is formed, it gets evaluated.

The splitting is done in the level of subseries created by the window function. So, the last window of one batch and the one first window of the next batch are two adjacent elements of the list of all extracted subseries by the single-step window and overlap in all elements but one, shifted by one element. This means that the subseries evaluated and the number of scores generated is identical to the offline settings, but are done incrementally. In some cases, the last batch has fewer data points than the selected subsequence length. In such cases, these points are rejected, and a batch is not formed.

As with the offline evaluation, the total results are padded.

2.4. Variant 2: Batching with history

In the streaming variant with batch history, the time series are initially partitioned. Each partition (except the first) is combined with the preceding partition (can also be extended to incorporate additional preceding partitions) to incorporate historical context. This historical context helps smooth transitions between partitions by averaging the anomaly scores at the intersection of the two partitions. This builds upon the naive streaming variant, emphasizing the criticality of incorporating historical context in time series analysis for the following reasons:

- Possible enhancement of prediction accuracy — By considering a broader historical context, the algorithm has the potential of making predictions more effectively, potentially improving accuracy by leveraging past trends and patterns.
- Possible improvement of distribution shift detection — Detecting shifts and anomalies in data distributions can become more robust. When the distribution of the current batch/partition deviates from the past, integrating historical partitions allows for more informed decision-making in processing scores across different segments.

2.5. Variant 3-4: Dynamic batching

In the streaming variant with change point detection, we aim to improve upon the simplistic and arbitrary choice of partition/batch size. Instead of using a fixed partition size which may not capture the intrinsic characteristics of the time series, we dynamically partition the time series according to detected abrupt abnormal changes. This approach ensures that the partitioning process better reflects the actual properties and behavior of the time series.

We start by creating an initial partition of a predefined length to establish its minimum and maximum points. This partition serves as a baseline assumption without anomalies or outliers, which is typical in real-time series analysis. We define parameters before proceeding to subsequent partitions, such as:

- The number of points p to consider after an abrupt change has been detected.
- A threshold beyond which a point is considered abnormal.
- A threshold beyond which a distribution shift is detected.

As new points arrive following the points comprising the initial partition, each new point undergoes scrutiny for deviations from the maximum and minimum values multiplied by a threshold. By multiplying the max and min values by a threshold, we effectively define upper and lower bounds beyond which a point is considered abnormal (indicating a change has been detected). If deviations exceed these bounds, suggesting a potential change, we accumulate subsequent points until a sufficient number p of post-change points is gathered. This approach accommodates abrupt changes spanning multiple points rather than mere spikes, ensuring robust detection of significant distribution shifts. If a substantial proportion of post-change points surpasses the bounds (as defined by a threshold), we adjust the maximum and minimum values to reflect the revised data distribution and integrate the current partition into our segmented data list. This process is repeated for subsequent partitions. This is variant 3. A variant of this methodology involves utilizing the 5% and 95% percentiles to detect abrupt changes. The overall approach remains largely the same; however, instead of scrutinizing new points for deviations from predefined upper and lower bounds, we check whether the points fall within or beyond the 5% and 95% percentile range. This approach makes the assumption that even the abnormal points (anomalies), are expected even beyond the 5%-95% range. This is variant 4.

Following the dynamic partitioning approach, anomaly detection is conducted by iterating through the time series partitions and employing the Matrix Profile and Isolation Forest methods.

2.6. Tuning

To investigate the effect of the parameter selection on the performance of the algorithms, aside from the methodologies described in the earlier subsections, we tuned these parameters before preforming evaluation for the case of Batching with history. For this purpose, we selected a few parameters to tune and a list of values for each one, for both Isolation Forest and Matrix Profile. So, a grid of all the combinations of parameters was formed.

To determine the best set of parameters for each algorithm, or in other words to tune it, for each combination of parameters in the grid, the algorithm was evaluated for a time series and the ROC-AUC score was determined. The best parameter combination was that one with the highest AUC score.

Multiple parameters were considered during the tuning process. For the Matrix Profile (stump), the parameters chosen were the sliding window length (subsequence length), the number of the closest neighbor to calculate the distance to (instead of the closest) and whether to normalize the subsequence distances or not. For the Isolation Forest, the parameters chosen were the sliding window length (subsequence length), and number of estimators (number of random trees in the ensemble). In total, 60 combinations of parameters were evaluated for the Matrix Profile and 9 for the Isolation Forest.

In practice, for each time series, we determined the best combination of parameters in the tuning version of the time series, and evaluated the evaluation version as usual, but with the optimized parameters selected.

3. Experimental Results

In this section, we present the results from our experiments. As the evaluation metric, we use the ROC-AUC. This metric allows us to evaluate the anomaly detection methods in a way that we don't have to rely on a way to set a score threshold, beyond which subseries are tagged as anomalies. At the same time, we can assess the ability of each method to distinguish between normal and anomalous subseries.

3.1. Outlier Detection Methods Evaluation

The results of the streaming variants (see Table 1), including the baseline methods (Static IF and MP, and SAND), shows that no method consistently outperforms others across all time series as the performance of each method varies significantly, making it difficult to draw definitive conclusions. We encourage readers to look deeper into the detailed performance metrics for selected time series to gain a more comprehensive understanding and draw conclusions based on the specific use case.

Our aim is to assess the overall performance of the methods and answer whether the proposed modifications to the offline (variants) truly outperform the baseline (static and SAND)

Time Series	Static		Variant 1		Variant 2		Variant 3		Variant 4		SAND
	IF	MP	IF	MP	IF	MP	IF	MP	IF	MP	SAND
Single Normality											
ECG1	**0.96**	0.64	0.86	0.70	0.83	0.85	0.82	0.71	0.81	0.69	0.77
ECG1_20k	0.97	0.86	0.86	0.69	0.83	0.97	0.81	0.65	0.78	0.72	**0.99**
IOPS1	0.53	**0.72**	0.51	0.60	0.48	0.46	0.46	0.68	0.53	0.41	0.70
SMD1	**0.85**	0.47	0.37	0.56	0.32	0.46	0.38	0.52	0.49	0.51	0.57
Occupancy1	**0.87**	0.17	0.71	0.19	0.76	0.19	0.85	0.16	0.78	0.24	0.71
Average	**0.84**	0.57	0.66	0.55	0.64	0.59	0.66	0.55	0.68	0.51	0.75
Double Normality											
ECG1+IOPS1	0.81	0.76	0.68	0.71	0.67	**0.90**	0.67	0.64	0.65	0.68	0.87
SMD1+Occupancy1	**0.83**	0.38	0.50	0.44	0.48	0.37	0.53	0.42	0.60	0.41	0.59
Average	**0.82**	0.57	0.59	0.57	0.58	0.64	0.6	0.53	0.63	0.54	0.73
Triple Normality											
ECG1+IOPS1 +Occupancy1	**0.88**	0.77	0.76	0.50	0.76	0.69	0.84	0.60	0.77	0.64	0.77
SMD1+ECG1 +Occupancy1	0.69	0.58	0.65	0.54	0.62	0.57	0.61	0.58	0.69	0.55	**0.78**
Average	**0.79**	0.67	0.71	0.52	0.69	0.63	0.73	0.59	0.73	0.60	0.78
Quadruple Normality											
ECG1+IOPS1 +SMD1+Occupancy1	0.65	0.61	0.60	0.53	0.57	0.59	**0.76**	0.58	0.62	0.57	0.52
Average	0.65	0.61	0.60	0.53	0.57	0.59	**0.76**	0.58	0.62	0.57	0.52

Table 1: The AUC scores resulting from the evaluations of all the time series, across multiple normalities, algorithms, and variants. IF refers to the Isolation Forest algorithm, while MP refers to Matrix Profile.

	Static	Variant 1	Variant 2	Variant 3	Variant 4	SAND
Single Normality	0.71	0.61	0.62	0.61	0.60	**0.75**
Double Normality	0.70	0.58	0.61	0.57	0.59	**0.73**
Triple Normality	0.73	0.62	0.66	0.66	0.67	**0.78**
Quadruple Normality	0.63	0.56	0.58	**0.67**	0.62	0.60

Table 2: The averages of the averages AUC scores for the IF and MP for all variants and normalities.

methods, as well as the naive conversion of the offline to online. We compare variants 2,3, and 4 with the static (offline) execution, variant 1, and SAND which act as the baselines. For ease of comparison between the methods, as there are two algorithms evaluated for each variant, we average the averages for each variant for each normality in Table 2. This assists us in comparing the proposed modifications (variants) themselves, instead of the performance of each algorithm under its variant. The variants are algorithm-agnostic; hence, we evaluate their performance across algorithms and not individually for each algorithm.

According to Table 2, our streaming variants (variant 2-4) do not outperform the static methods and the SAND baseline in all normalities but the quadruple. Additionally, variant 2, in all normalities, outperforms variant 1 which also acts as a baseline. Variants 3 and 4 perform similarly to variant 1 in single and double normalities and better in triple and quadruple.

Moreover, the state-of-the-art SAND algorithm outperforms all other variants, including the static approaches, for single, double, and triple normalities. However, for quadruple normality, the performance is significantly lower, and the third variant surpasses it. Apart from SAND being the top-performing algorithm overall, the static approaches rank next in terms of performance. If instead of evaluating the average performance of methods, we considered the performance of each method-algorithm pair individually, it is apparent that Isolation Forest in a static setting generally (in most time series) outperforms all other cases.

3.2. Tuning Evaluation

In Table 3, we present the results of the parameter-tuned and untuned Isolation Forest and Matrix Profile algorithms. The Isolation Forest exhibits worse performance when parameter tuned using different time series from the same domain compared to using the default parameters. This clearly indicates

Time Series	Default Parameters		Tuned Parameters	
	IF	MP	IF	MP
Single Normality				
ECG1	0.83	0.70	0.66	**0.93**
ECG1_20k	0.83	0.69	0.63	**0.98**
IOPS1	0.48	0.60	0.67	**0.71**
SMD1	0.32	0.56	0.24	**0.90**
Occupancy1	0.76	0.19	0.70	**0.86**
Average	0.64	0.55	0.58	**0.88**
Double Normality				
ECG1+IOPS1	0.67	0.71	0.54	**0.88**
SMD1+Occupancy1	0.48	0.44	0.42	**0.90**
Average	0.58	0.57	0.48	**0.89**
Triple Normality				
ECG1+IOPS1 +Occupancy1	0.76	0.50	0.65	**0.89**
SMD1+ECG1 +Occupancy1	0.62	0.54	0.53	**0.79**
Average	0.69	0.52	0.59	**0.84**
Quadruple Normality				
ECG1+IOPS1 +SMD1+Occupancy1	0.57	0.53	0.50	**0.79**
Average	0.57	0.53	0.50	**0.79**

Table 3: The AUC scores for the default and tuned algorithms for the second variant (Batch with history).

that, in our experiments, the Isolation Forest cannot optimize effectively using related time series.

Conversely, the Matrix Profile demonstrates an exceptional performance gain when leveraging tuned parameters using same-domain time series. This finding is particularly noteworthy as it indicates that tuning the Matrix Profile consistently yields significantly better results than tuning the method in isolation with default parameters. Leveraging time series from the same domain to tune methods is practical and feasible, giving the Matrix Profile a distinct advantage and a greater potential over the Isolation Forest method.

4. Conclusions

To summarize, this study develops streaming versions of offline anomaly detection methods for time series data. It aims to adapt offline anomaly detection techniques to real-time scenarios. Streaming variants for Isolation Forest and Matrix Profile are proposed and evaluated on selected time series of the TSB-UAD benchmark time series, comparing their performance with non-streaming methods and the SAND algorithm. While our streaming variants performed adequately across various time series, they still did not manage to consistently outperform SAND and the static context.

5. Future Work

Future work could explore a more advanced version of the dynamic batching approach by iterating through dynamically partitioned time series data. For any non-initial partition, a subset of the previous partitions could be incorporated, specifically starting from the point of detected abrupt changes and including the next p points up to the end of the previous partition. The anomaly score of the current partition, would then be compared with the anomaly score of the previous partition and by intersecting the two scores, significant discrepancies could be identified. The previous score corresponding to the subset could then be updated with the new score from the current partition. This iterative approach might enable robust monitoring and detection of anomalies within the evolving time series data. This approach has theoretical implications for two key reasons:

1. If an anomaly exists within the intersection subset of points and both the current and previous partitions (excluding the intersection) follow similar distributions, both the current and previous classifiers will detect the anomaly. This is because the distributions are similar, making the points in the subset stand out significantly from their respective distributions.

2. If a distribution shift occurs in the current partition and the previous classifier detects an anomaly in the subset, the two classifiers (current and previous) will disagree on the subset. However, because the distribution shift in the current partition is natural and not an anomaly, the anomaly score assigned by the previous classifier will be replaced by the accurate score from the current classifier, reflecting the new distribution information.

References

[1] J. Paparrizos, Y. Kang, P. Boniol, R. S. Tsay, T. Palpanas, M. J. Franklin, Tsb-uad: an end-to-end benchmark suite for univariate time-series anomaly detection, Proceedings of the VLDB Endowment 15 (8) (2022) 1697–1711.

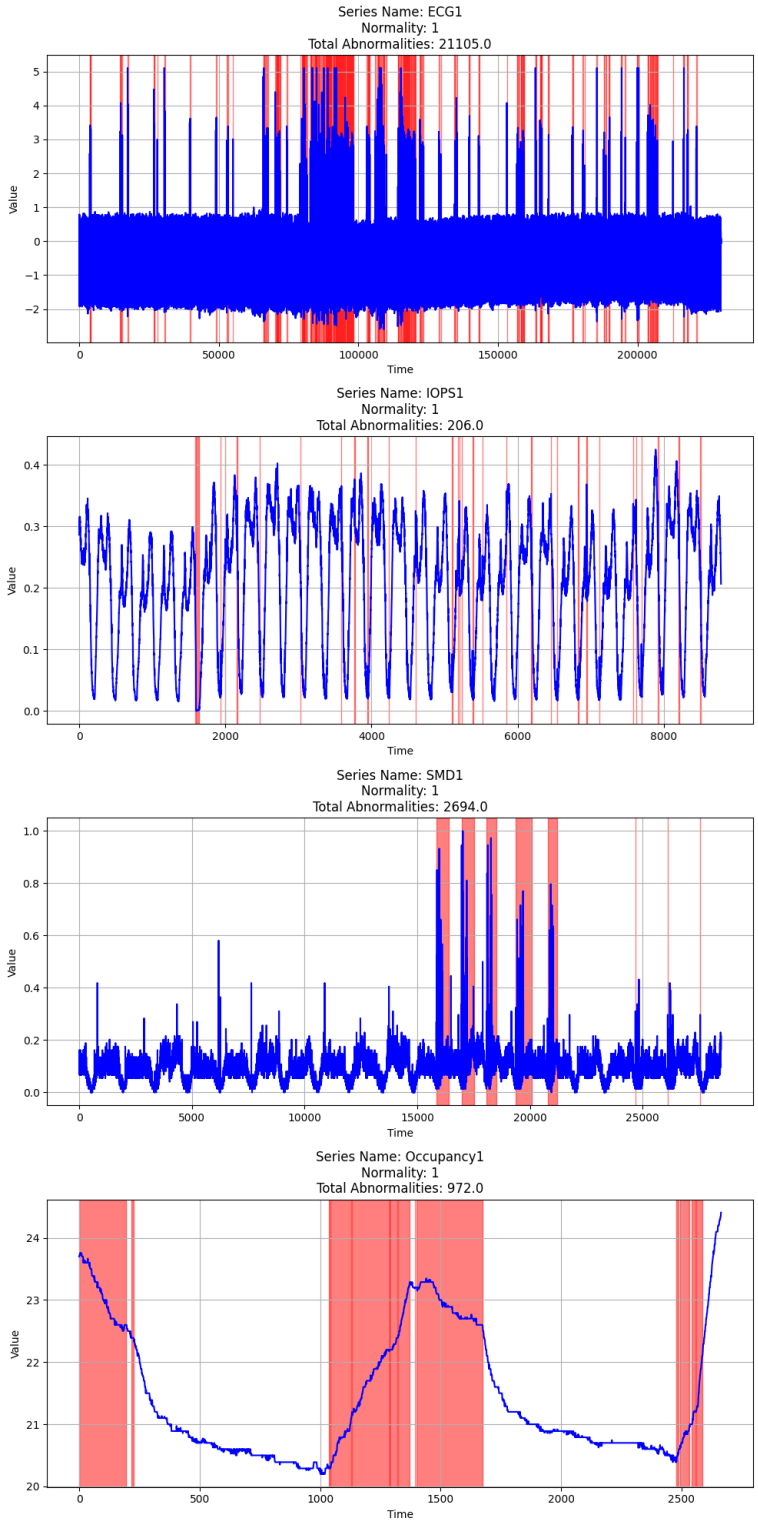
[2] P. Boniol, J. Paparrizos, T. Palpanas, M. J. Franklin, Sand: streaming subsequence anomaly detection, Proceedings of the VLDB Endowment 14 (10) (2021) 1717–1729.

[3] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, E. Keogh, Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016, pp. 1317–1322. doi:10.1109/ICDM.2016.0179.

[4] F. T. Liu, K. M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 eighth ieee international conference on data mining, IEEE, 2008, pp. 413–422.

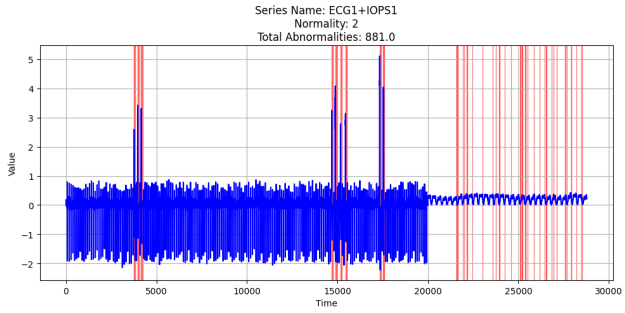
Appendix A. Data Availability and time series Plots

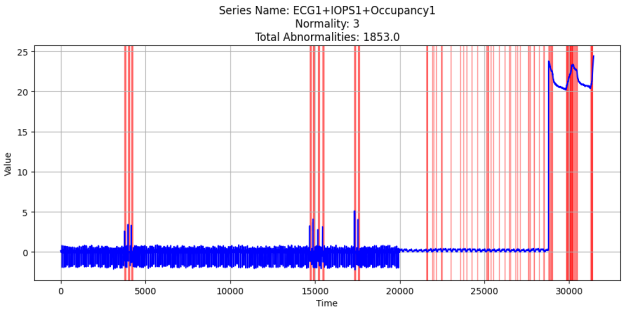
The full code for this project can be found in the following GitHub repository: `https://github.com/GeorgeM2000/Streamify-timeseries-Anomaly-Detection-Methods`.

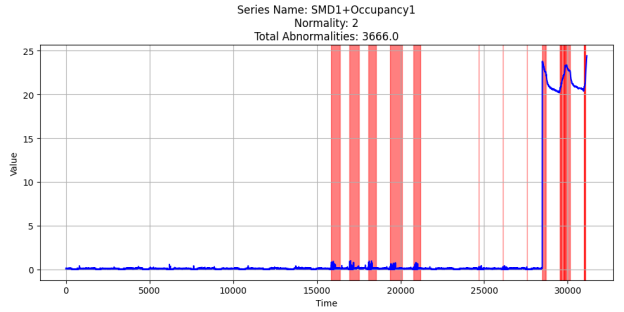


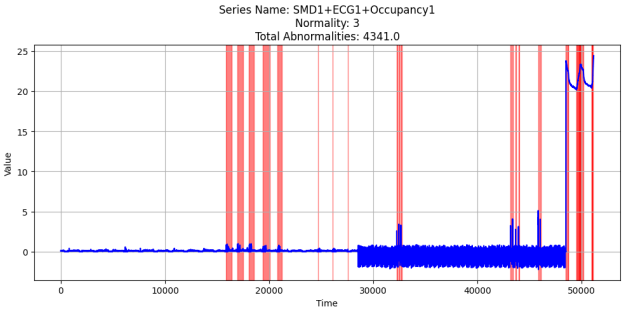
(a) Normality 1

Figure A.1: time series with normality 1. The anomalous data points are highlighted in red.



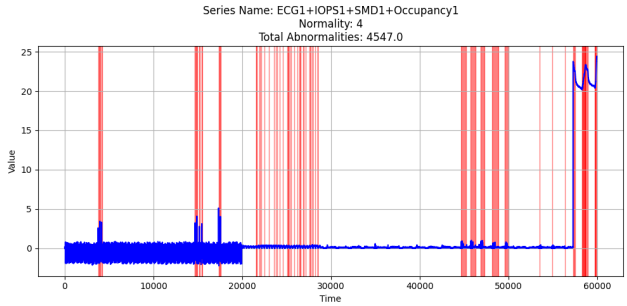


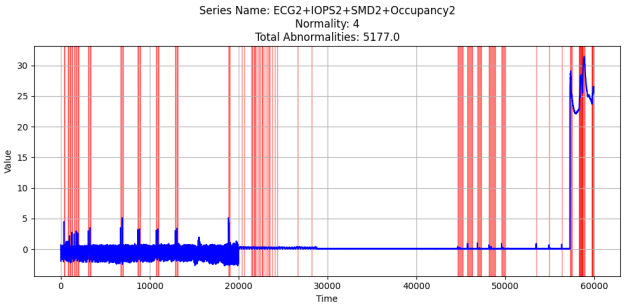




(a) Normality 2

(b) Normality 3





(c) Normality 4

(d) Normality 4

Figure A.2: time series with normality 2, 3, and 4. The anomalous data points are highlighted in red.