TU Delft

Delft University of Technology

# Is your anomaly detector ready for change? adapting aiops solutions to the real world

Poenaru-Olaru, Lorena; Karpova, Natalia; Cruz, Luis; Rellermeyer, Jan S.; Van Deursen, Arie

**DOI**
10.1145/3644815.3644961

**Publication date**
2024

**Document Version**
Final published version

**Published in**
Proceedings - 2024 IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, CAIN 2024

**Citation (APA)**
Poenaru-Olaru, L., Karpova, N., Cruz, L., Rellermeyer, J. S., & Van Deursen, A. (2024). Is your anomaly detector ready for change? adapting aiops solutions to the real world. In *Proceedings - 2024 IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, CAIN 2024* (pp. 222-233). (Proceedings - 2024 IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, CAIN 2024). ACM. https://doi.org/10.1145/3644815.3644961

**Important note**
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

**Copyright**
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

*This work is downloaded from Delft University of Technology.*
*For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.*



# Is Your Anomaly Detector Ready for Change? Adapting AIOps Solutions to the Real World

Lorena Poenaru-Olaru
L.Poenaru-Olaru@tudelft.nl
Delft University of Technology
Netherlands

Natalia Karpova
karp.n.b@gmail.com
Delft University of Technology
Netherlands

Luis Cruz
L.Cruz@tudelft.nl
Delft University of Technology
Netherlands

Jan S. Rellermeyer
rellermeyer@vss.uni-hannover.de
Leibniz University Hannover
Germany

Arie van Deursen
arie.vandeursen@tudelft.nl
Delft University of Technology
Netherlands

## ABSTRACT

Anomaly detection techniques are essential in automating the monitoring of IT systems and operations. These techniques imply that machine learning algorithms are trained on operational data corresponding to a specific period of time and that they are continuously evaluated on newly emerging data. Operational data is constantly changing over time, which affects the performance of deployed anomaly detection models. Therefore, continuous model maintenance is required to preserve the performance of anomaly detectors over time. In this work, we analyze two different anomaly detection model maintenance techniques in terms of the model update frequency, namely blind model retraining and informed model retraining. We further investigate the effects of updating the model by retraining it on all the available data (full-history approach) and only the newest data (sliding window approach). Moreover, we investigate whether a data change monitoring tool is capable of determining when the anomaly detection model needs to be updated through retraining.

## KEYWORDS

anomaly detection, AIOps, model monitoring, model maintenance, concept drift detection

**ACM Reference Format:**
Lorena Poenaru-Olaru, Natalia Karpova, Luis Cruz, Jan S. Rellermeyer, and Arie van Deursen. 2024. Is Your Anomaly Detector Ready for Change? Adapting AIOps Solutions to the Real World. In *Conference on AI Engineering Software Engineering for AI (CAIN 2024), April 14–15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3644815.3644961

## 1 INTRODUCTION

The field of AIOps refers to applying artificial intelligence (AI) techniques on large-scale operational data to solve challenges derived from operational workflows. AIOps aims to increase the productivity of IT Ops, DevOps, and software reliability engineering teams by predicting the behavior of large-scale software systems and improving the software architectural decision-making processes [9]. The term AIOps solutions is used to describe machine learning (ML) systems that learn from operational data. In the past years, AIOps solutions have witnessed a fast adoption within different industries. The most popular AIOps solutions are failure prediction and anomaly detection [26].

Given that any AIOps solution is an ML system, the quality of its predictions is strongly dependent on the data it was trained on and the data it is evaluated on after being deployed into production. If the training data is significantly different than the evaluation data, there is a high chance that the performance of the ML system will be affected [28]. Previous work on failure prediction AIOps solutions has observed the evolving character of operational data, which implies that this data is continuously changing over time [22], [16], [18], [21]. The continuous data changes, also known as concept drift, influence the performance of AIOps solutions during their lifecycle. They become less accurate over time, which tremendously affects their reliability among practitioners. Although the concept drift cannot be prevented since it is caused by external hidden factors, AIOps practitioners need to constantly ensure that their AIOps solutions are up to date.

One solution that ML practitioners adopt to handle the evolving character of data is retraining/updating ML models over time [25]. Periodical model retraining has also been studied for failure detection AIOps solutions [23], [22] and has proved that continuous model updates achieve better performance over time compared to non-updated models. However, the effects of continuously updated models have only been studied for failure prediction models [22], [23].

In [27] the authors proposed a solution to handle concept drift in failure prediction AIOps solutions, which implies that the model is updated only when the concept drift is detected, instead of periodically. Therefore, their solution contains a monitoring part responsible for identifying concept drift in the evaluation data using a concept drift detector. However, this framework was just proposed, but it was never assessed on any AIOps solution.

When it comes to periodic model retraining, this technique has been studied for classification problems such as failure predictions AIOps solutions. Lyu et al. [23] suggested that it also needs to be studied on other AIOps solutions, such as anomaly detection on time series data. By doing so, AIOps practitioners could have a



This work licensed under Creative Commons Attribution International 4.0 License.

*CAIN 2024, April 14–15, 2024, Lisbon, Portugal*
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0591-5/24/04.
https://doi.org/10.1145/3644815.3644961

better understanding of whether they can mitigate the effects of concept drift by constantly updating anomaly detectors. Regarding concept drift monitoring-based model retraining, this technique was not previously applied to any AIOps solution. Furthermore, previous work suggests that organizations do not have monitoring infrastructure to detect drift in production [25], [24] and they only perform periodic model retraining based on human decisions. Examining the impact of drift detection monitoring tools is the first step toward automating the maintenance of machine learning in production. Therefore, it could help in understanding whether concept drift detectors could be quality and reliability indicators for deployed ML models.

To mitigate the effects of concept drift on anomaly detection AIOps solutions, in this paper, we study different model adaptation techniques. Our contributions can be summarized as follows:

(1) We examine the effect of periodically updating models on the performance of anomaly detection AIOps solutions.
(2) We assess and report the limitations of the state-of-the-art anomaly detection models when being evaluated on different data sizes.
(3) We investigate the effects of retraining anomaly detection AIOps models based on the output of a concept drift detector.
(4) A publicly available replication package is provided including the implementation of a concept drift detector for time series.

## 2 BACKGROUND AND RELATED WORK

### 2.1 Background AIOps Solutions & Anomaly Detection

AIOps solutions aim to identify issues in large software systems and then help with mitigating them or providing recommendations to engineers. When it comes to detecting possible issues in the system, a variety of different AIOps solutions were proposed for failure prediction tasks, namely predicting job failure [33], node failure [19], disk failure [3], incident [6] or outage [7]. Besides failures, plenty of attention has been paid to identifying abnormal system behavior, such as performance anomalies [14], [20], anomalies in system logs [13], [17], or internet traffic anomalies [31].

Although plenty of anomaly detection techniques were proposed in the literature, regarding AIOps solutions for anomaly detection, previous work has focused chiefly on *unsupervised* and *semi-supervised* models. This is a technique that practitioners use to handle the label availability challenge and high cost of obtaining true labels [31]. The best-performing and most popular techniques to detect anomalies in univariate AIOps data [5], [31], [39] belong to the *signal reconstruction models* group [34].

Anomalies are detected using methods that encode the time series into a latent space, such as Fast *Fourier Transform (FFT)* [30], *Spectral Residuals (SR)* [31] or *Prediction Confidence Interval (PCI)* [42]. These techniques usually have low computational costs, but they lose information during the encoding process [34]. To preserve more information, a type of artificial neural network called Auto-Encoder (AE) is employed to transform the time series data into a latent space. Thereby, plenty of anomaly detection techniques based on Auto-Encoders were derived, namely *Long Short-Term Memory Autoencoder (LSTM-AE)* [29] or *DONUT* [37]. Furthermore, Microsoft presents a more complex anomaly detector, *Spectral Residuals Convolutional Neural Networks (SR-CNN)* [31] that learns from multiple time series, generates synthetic anomalous samples, and trains a Convolutional Neural Network (CNN) to distinguish between anomalous and non-anomalous samples.

The anomaly detection models were previously evaluated using a delay metric. The reason for this is that in real-world applications, anomalies can occur either as single points or as segments of anomalies (group of continuous anomalies) [35]. According to [31], [40], [5], in AIOps solutions detecting any anomaly point in a segment of anomalies with a relatively small delay is considered as successful as detecting all anomaly points belonging to the same segment. Therefore, a delay-based evaluation strategy was presented in previous anomaly detection studies [31], [40], [5].

### 2.2 Concept Drift in Operational Data and Model Adaptation Techiques

Previous work observed the evolving character of operational data, which is responsible for changes over time and, therefore, for concept drift [22], [9], [18], [21]. The presence of concept drift results in the degradation in performance of the failure prediction models. Therefore, AIOps solutions need to be constantly maintained over time by continuous retraining [22], [23].

Gama et al. [11] propose two machine learning retraining techniques from the perspective of the retraining frequency, namely *blind retraining* and *informed retraining*. Blind retraining is the equivalent of periodic retraining, where the model is retrained after a predefined period. This technique was previously used by previous work on failure prediction models [18], [21], [22], [23], [32] and proved to be beneficial for preserving the model's performance over time. Informed retraining implies the existence of a data monitoring tool called a *concept drift detector* which indicates when the model is outdated due to changes in data. However, there is currently no study on the effects of blind and informed retraining on anomaly detection models.

Lyu et al. [23] propose two retraining techniques from the perspective of retraining data, namely the *full-history approach* and the *sliding window approach*. The full-history approach constantly enriches the training dataset with the newest data and retrains the model, while the sliding window approach retrains the model only on the most recent data, discarding old samples. These methods have been studied in failure prediction models, but there is currently no work on the effects of the full-history approach and sliding window on anomaly detection models.

### 2.3 Concept Drift Detection

Concept drift is monitored using some algorithms that can capture the moment when data shift occurs called *concept drift detectors* [28]. Although there are plenty of drift detectors available for multivariate data used in classification purposes when it comes to time series, the number of available drift detection techniques is significantly lower [2]. According to Bayram et al. [2], the reason for this is the lack of available open-source relevant time series datasets to study drift detection. Despite the potential of concept drift detectors to monitor data against concept drift, they have not been yet applied to the AIOps domain. Furthermore, there is no study on concept

drift monitoring techniques for anomaly detection on operational data.

Out of the existing concept drift detectors for time series we can mention *Feature Extraction Drift Detection (FEDD)* [4] and *Entropy-Based Time Domain Feature Extraction (ETFE)* [10]. Both these detectors identify drift by extracting features from a given time series window and observing their similarity with the features extracted from the reference window. FEDD is extracting six linear (autocorrelation, partial autocorrelation, variance, skewness coefficient, kurtosis coefficient, and turning point rate) and two non-linear features (bicorrelation and mutual information) and computing the dissimilarity between features extracted in the current window and the reference window. The drift is detected through a change detector which analyses the exponentially weighted moving average (EWMA) of the computed dissimilarities. ETFE is extracting solely entropy-related features (approximate, fuzzy, sample, permutation, increment, and weighted permutation entropy) from the decomposed time series. Concept drift is detected from the features extracted from the reference and the current time series using the GLR statistical test.

## 3 MOTIVATIONAL EXAMPLE

In Figure 1, we depict an example of two different time series related to internet traffic from the Yahoo dataset, a popular benchmark for AIOps data related to internet traffic [39], [31], [5]. In the upper plot, the behavior of operational data is significantly changing after a certain timestamp, while in the lower plot, it remains relatively constant. Thus, different time series have different behaviors over time.

We assume that we train two anomaly detection models on the data corresponding to the left of the vertical line and we keep the rest for testing. In the upper time series, we notice that the testing data is significantly different than the training data, which means that the model requires updating over time. Otherwise, the model would signal plenty of false alarms, which need to be verified by the operation and service engineers. Thus, the advantage of saving data monitoring time using anomaly detection AIOps solutions is diminished by the amount of time engineers need to spend searching for the causes of erroneously identified anomalies. Furthermore, the engineers would doubt the reliability of the model and they will be more reluctant to employ it. However, we notice that the data of the lower time series is not changing over time, thus the model should be able to capture anomalies.

If we would maintain our models using *blind retraining*, the model corresponding to the lower time series would be retrained although their performance might not be drastically improved since there were no changes in the data. Thus, we might encounter unnecessary retraining costs, which could be avoided by employing *informed retraining* where we initially verify whether data has changed and then update the model. When it comes to the model corresponding to the upper time series, both blind retraining and informed retraining might be beneficial.

Using a full-history approach in the case of the upper time series might result in poor anomaly detection performance for the upper time series since the past data no longer resembles current data. However, in the case of the lower time series, both techniques

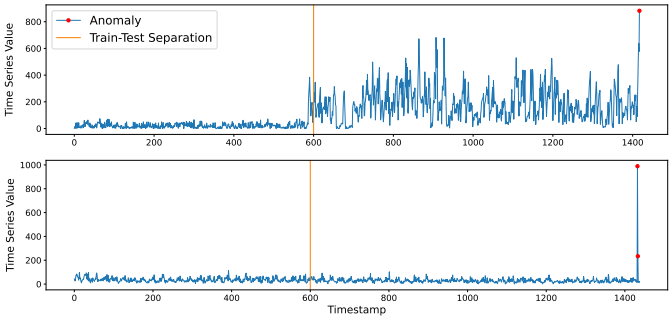



**Figure 1: The time series to the left of the vertical line represents the training data, while the one to the right represents the evaluation/testing data.**

sliding window and full-history approaches might lead to similar results.

Given that time series are different, they might require different maintenance techniques. Therefore, in our study, we address the effects of updating different models over time on their performance and the implications of each maintenance technique in terms of how accurately they detect anomalies and how many false alarms they raise.

## 4 RESEARCH QUESTIONS

We begin by assessing the performance of state-of-the-art anomaly detection models on operational data. Moreover, we aim to understand whether the size of the testing set affects the performance of these models. We further aim to investigate the lifecycle of these models in the situation in which the models are maintained over time compared to when they are never updated. We analyze these aspects by evaluating the model retraining effects from the perspective of the retraining data and retraining frequency.

(1) What is the performance of state-of-the-art anomaly detection models on operational data?
   (a) How robust are the state-of-the-art models to the testing set size?
(2) What is the impact of the two model retraining techniques from the perspective of the retraining data (full-history vs sliding window approach)?
(3) What is the impact of the two model retraining techniques from the perspective of the retraining frequency (blind vs informed retraining)?

## 5 EVALUATION METHODOLOGY

In this section, we present the evaluation methodology that we employ to answer our research questions. We begin by describing the employed datasets and anomaly detection models. We continue by presenting different retraining (maintenance) techniques and the concept drift detector monitoring tool that signals changes in the data over time. Lastly, we explain the used evaluation metrics together with the delay tolerance.

## 5.1 Datasets

Although there are various benchmarks for anomaly detection, very few benchmarks for AIOps anomaly detection exist since research on AIOps is mostly performed using proprietary production data that is not publicly released [8]. Therefore, we selected two popular operational datasets containing univariate time series with different data collection granularity and lengths as presented in Table 1, that were previously used to build anomaly detection AIOps solutions, namely Yahoo S5[1] and the Numenta Anomaly Benchmark (NAB)[2]. Although the publicly available benchmarks received criticism in the way they are labeled [38], labeling anomalies in real-world AIOps datasets is complex, labor-intensive, and prone to error due to manual labeling and the subjectivity of the annotator [8]. For instance, some annotators correlate anomalies with the moment an incident occurs [8], without taking into account that the points before and after the incident are identical as also pointed out by [38]. Furthermore, currently, the AIOps time series data are collected from various sources, which is the reason why anomalies exhibit different behaviors and they are not consistent across time-series [8]. Therefore, the anomaly detection AIOps datasets benchmarks reflect real-world characteristics of operational data, but as mentioned in [8], [38], the labeling process requires more transparency by understanding the human decision-making process while labeling and the cause of the anomaly.

**Yahoo** is a dataset released by Yahoo Lab which contains both synthetic and real data collected from *internet traffic* generated by Yahoo services. In our experiments, we solely considered real-world data (the Yahoo A1 benchmark), which we are further referring to as Yahoo. This dataset is composed of 67 time series collected with a granularity of 1 hour for 31 days (the shortest time series) and 61 days (the longest time series). The anomalous and non-anomalous points are labeled by domain experts.

**NAB** is a publicly available dataset previously used as a benchmark to assess the performance of anomaly detection models [36]. The NAB corpus contains multiple types of time series, such as metro traffic, tweets, etc. For our experiments, we solely select datasets referring to operational data, namely the realAWSCloudwatch time series that contain real-world server metrics (e.g. CPU utilization, Network Bytes In, Disk Read Bytes) collected by the AmazonCloudwatch service. For simplicity, we are further referring to these time series as NAB. This dataset is composed of 17 time series collected with a granularity of 5 minutes for 5 days (the shortest time series) and 17 days (the longest time series). To obtain the labels, we employ the ground truth labeling approach (labeling only the ground truth as an anomaly) instead of labeling the entire region to avoid raising too many false alarms to engineers [1].

*Data-Splitting And Preprocessing.* In our study, we considered the same data-splitting scenario from previous works [31], [5], namely for each time series from each dataset the first half is used for training and the second half is used for testing. For our experiments, we assume that the anomaly labels are known for the training set.

[1]Yahoo Data Source: https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomaly
[2]NAB Data Source: https://github.com/numenta/NAB/tree/master/data

**Table 1: Overview of the datasets characteristics.**

| Dataset | NAB | Yahoo A1 |
|---|---|---|
| Number of Time Series in Dataset | 17 | 67 |
| Granularity | 5 min | 1 hour |
| Approx. Min Data Collection Time | 5 days | 31 days |
| Approx. Max Data Collection Time | 17 days | 61 days |

## 5.2 Models

*5.2.1 State-of-the-Art Anomaly Detection.* We replicate five popular unsupervised anomaly detection models based on signal reconstruction, namely *FFT* [30], *SR* [31], *PCI* [42], *LSTM-AE* [29], and *SR-CNN* [31]. We selected our models based on both their popularity and their dissimilarity in techniques to detect anomalies. Furthermore, these models were previously used for anomaly detection in AIOps [31], [5], [39]. Therefore, we employ three models that work by solely applying different mathematical operations of the given time series to detect anomalies (FFT, PCI, and SR) and two models that require learning the behavior of the time series (LSTM-AE, and SR-CNN).

The **FFT** method firstly transforms the time series from the time domain into the frequency domain using the Fourier transform and then transforms the time series back into the time domain to estimate a fitted curve of the data. Anomalies are detected by finding differences between data points and the fitted curves. In our study, we use the implementation of FFT from the original paper [30], which is publicly available[3]. Given the unavailability of a threshold that differentiates anomalies from non-anomalies, we empirically computed the threshold using the training data. Therefore, we remove all existing anomalies from the training data and we extract the maximum anomaly score on the training data. Sample with a higher anomaly score than the threshold are classified as anomalies.

The **SR** method is composed of three steps: the Fast Fourier Transform, which transforms the time series from the time domain into the frequency domain to calculate the log amplitude spectrum, the spectral residuals calculation and the Inverse Fourier Transform transforms the signal from the frequency domain back into the time domain. After these three steps, the saliency map is computed and used to detect anomalies through a threshold. In our study, we use the publicly available implementation of SR[4] from the original paper [31]. Optimal hyperparameters are determined by performing a grid search starting from the parameters suggested in [31] until similar results are obtained on Yahoo. The NAB dataset is not assessed in the original paper and, thus, we perform a grid search to identify the optimal parameters.

The **PCI** method relies on the k-nearest neighbors of one data point to identify if that specific data point is an anomaly. The k-nearest neighbors are the closest continuous data points to the targeted data point in the time series. After calculating the nearest neighbors, the prediction confidence interval is calculated. The current data point is classified as an anomaly if its value is outside

[3]FFT Open Source Implementation: https://github.com/HPI-Information-Systems/TimeEval-algorithms/tree/main/fft
[4]SR Open Source Implementation: https://github.com/y-bar/ml-based-anomaly-detection

of the prediction confidence interval. We use the publicly available implementation of PCI[5] from the original paper [42]. Optimal parameters are computed for each dataset through grid search.

The **LSTM-AE** method uses an autoencoder architecture to learn the representation of a time series in the time domain. An anomaly is detected when the error between the reconstructed and true time series is higher than a predefined threshold. Due to the lack of publicly available implementations of LSTM-AE, we implement it based on the architecture presented in [15]. We remove anomalous samples from the training set [5] and perform interpolation between the closest non-anomalous samples is performed to preserve the continuity of the time series. Our implementation is publicly available in our replication package[6].

The **SR-CNN** method is composed of two main parts: the synthetic data generation and the CNN. It uses the saliency map calculated through spectral residuals as features for the CNN. The synthetic data generator injects fake anomalous points into the saliency map to reduce the high imbalance between anomalous and non-anomalous points. The augmented data is used to train the CNN and distinguish between anomalies and non-anomalies. We used the publicly available implementation[7] of SR-CNN from the original paper [31]. We perform a grid search to determine the optimal parameters of SR-CNN on both datasets.

*5.2.2 Testing Window Size Assessment.* Previous work reported results by splitting the original time series into half and using the entire first half for training and the entire second half for testing. We are further referring to this experimental setup as the SoTA setup. However, since we aim to understand the effect of retraining techniques on the available datasets, we also need to ensure that the model's predictions are not impacted by a smaller testing window. Thus, we split the testing sets into smaller subsets, and we evaluate the same model on each of the subsets. We further combine the predictions on the subsets and compare them to the prediction obtained from the SoTA setup. We are further referring to this setup as the Window Size Setup. The testing subsets correspond to around 168 samples (one week) for Yahoo and 225 samples (one day) for NAB and they also represent the retraining periods. We use natural time intervals, one week and one day for Yahoo and NAB, respectively, since they are commonly used in model retraining for AIOps solutions [22], [18], [21], [41]. Moreover, these exact periods were chosen based on the amount of available data points in each time series (see Table 1). The same subsets are used in the model retraining experiments.

## 5.3 Retraining Techniques

*5.3.1 Retraining Data.* From a training data perspective, we compare three scenarios, namely the scenario when the model is never updated, which we are referring to as the *static approach* and two scenarios in which the model is updated over time, namely the *full history approach* and *sliding window approach*.

[5]PCI Open Source Implementation: https://github.com/HPI-Information-Systems/TimeEval-algorithms/tree/main/pci
[6]Replication Package: https://github.com/LorenaPoenaru/anomaly_detection
[7]SR-CNN Open Source Implementation: https://github.com/microsoft/anomalydetector




**Figure 2: Training and testing data in case of the static approach, full-history approach, and sliding window approach.**

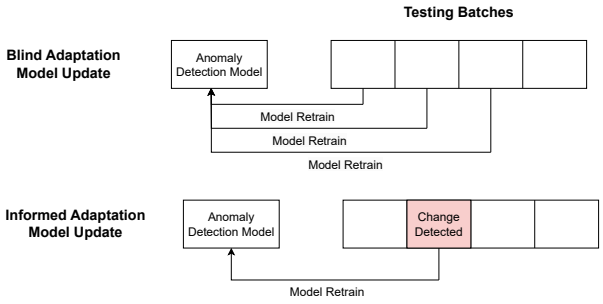



**Figure 3: Blind vs Informed Retraining.**

The testing data is split into equal testing batches corresponding to the predefined period. The experimental setup for the 3 scenarios can be observed in Figure 2. According to each scenario, the model update is done as follows:

**Static Approach (S):** The model is trained only once on the training data and tested on the testing batches.

**Full History Approach (FH):** The model is retrained periodically by constantly enriching the training set. Thus, when testing the model on the testing set corresponding to batch *t*, the model is trained on all the available data until batch *t-1*.

**Sliding Window Approach (SW):** The model is retrained periodically by constantly replacing old data with new data. Thus, when testing the model on the testing set corresponding to batch *t*, the model is retrained on only the newest data. In this approach, the training set size remains constant over time.

*5.3.2 Retraining Frequency.* From a retraining frequency perspective, we compare *blind retraining* with *informed retraining* updated models. The blind retraining technique implies that the model is retrained on a periodic basis by constantly including testing batches in the training data. Informed retraining implies that the model is retrained on all the data including the batch when the change occurs, only when a change detector identifies changes in data. A visual representation of the two is given in Figure 3.

## 5.4 Concept Drift Detectors

For this study, we only consider FEDD as a concept drift detector since it was evaluated on both synthetic and real-world data [12]. For our experiments, we extract the implementation of FEDD from

the replication package[8] provided by the authors of [12]. Furthermore, we translate the code into English and include the FEDD implementation in our replication package together with an example of how to run it on any time series such that practitioners can use it for their data.

## 5.5 Model Evaluation

*5.5.1 Evaluation Metrics.* We use F1-score, Precision, and Recall to evaluate the performance of anomaly detectors. We do not consider the computation time as one of our evaluation metrics since the evaluated time series are relatively short and, thus, meaningful conclusions cannot be drawn. The Precision metric shows the ability of the anomaly detector not to label a non-anomaly as an anomaly and, therefore, not to raise too many false alarms. The Recall metric shows the ability of the anomaly detector to find all the anomalies and, therefore, to correctly identify all the anomalies. The F1-Score shows the compromise between Precision and Recall and, therefore, how many anomalies are correctly identified with respect to how many false alarms are triggered. The metrics are defined by the following equations:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

where TP and FN represent the number of True Positives and False Negatives, respectively.

*5.5.2 Delay Metric.* As aforementioned, AIOps anomaly detection models were evaluated considering a predefined prediction tolerance called delay. The detection delay is the maximum tolerable number of anomalous points in the anomaly segment that the detector can omit. If the detectors find any anomaly in the segment within this delay, the detection is considered correct. Otherwise, the detector fails to capture the anomaly. We depict in Figure 4 an example of metric computation including the delay.

*5.5.3 Statistical Test.* We report the metrics averaged over all time series belonging to one dataset as done in previous work [31], [40], [5]. To understand whether the differences between the averaged metrics are statistically significant we employ the Wilcoxon signed-rank statistical test and assess whether the difference between the metrics obtained for each time series in one scenario and the metrics obtained for each time series in another scenario is statistically significant. The Wilcoxon signed rank is a non-parametric statistical test, thus it does not make any assumption about the data distributions. Its null hypothesis is that the two populations analyzed come from the same distribution and, thus, there is no statistically significant difference between the two scenarios. We use a confidence interval of 90% to assess whether the null hypothesis is accepted or rejected. To avoid bias in our experiments, we employ 5 random seeds in the case of LSTM-AE and SR-CNN, which suffer from randomness. We apply the statistical test on each random seed to assess the significance.

[8]FEDD Open Source Implementation: https://github.com/GustavoHFMO/IDPSO-ELM-S/blob/master/detectores/FEDD.py

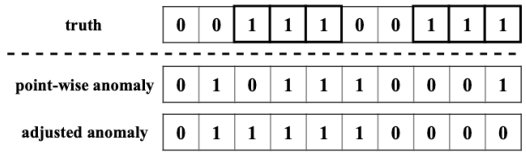



**Figure 4: In this example we show the labels for 10 data points corresponding to one time series, where 1 indicates an anomaly and 0 indicates a non-anomalous point. The first row shows the ground truth, the second row shows the original predictions of one anomaly detection model and the third row shows the adjusted predictions considering a delay of 1. In the ground truth, there are 2 anomalous segments, each containing 3 anomalies. Since the model managed to predict the second anomaly in the sequence of anomalies and we tolerate a delay of 1 sample, the adjusted anomaly treats the entire first anomaly segment as a correct prediction. However, the second sequence of anomalies is treated as an incorrect prediction since even with a delay of one the anomaly on position 8 is not reported in time, while with a delay of 2, it would be reported in time. [31]**

# 6 EXPERIMENTAL RESULTS

## 6.1 Anomaly Detection Models on Operational Data

*6.1.1 State-of-the-Art Setup.* The first set of experiments aims to assess the performance of state-of-the-art anomaly detection models on operational data. We initially train the model on the first half of each time series and test it on the second half (SoTA setup), similar to the state-of-the-art [31], [40], [5].

**Table 2: Results of anomaly detectors in SoTA setting. With bold we highlight the best scores. The obtained results are obtained by averaging the metric over all time series from each dataset and over the 5 random seeds.**

| | Model | F1-Score | Precision | Recall |
|---|---|---|---|---|
| Yahoo | FFT | 0.07 | 0.04 | 0.39 |
| | SR | **0.25** | **0.28** | **0.36** |
| | PCI | 0.09 | 0.06 | 0.39 |
| | LSTM-AE | **0.37** | **0.38** | **0.49** |
| | SR-CNN | **0.57** | **0.54** | **0.61** |
| NAB | FFT | 0.04 | 0.02 | 0.12 |
| | SR | 0.05 | 0.15 | 0.33 |
| | PCI | 0.03 | 0.02 | 0.32 |
| | LSTM-AE | **0.31** | **0.37** | **0.44** |
| | SR-CNN | **0.82** | **0.70** | **0.99** |

We depict our findings in Table 2 where we can observe that the performance of anomaly detection models is overall relatively low, which could be a reason for the labeling problems identified in [38]. FFT and PCI models obtained the lowest F1-score for both

datasets. When it comes to PCI, we can observe that it tends to classify most data points as anomalies given its high recall and low precision on both datasets. The same tendency can be observed when it comes to FFT on the Yahoo dataset and the SR on the NAB dataset. Given the low performance of FFT and PCI on both operational datasets, these two models are no longer considered in the following experiments. Moreover, the SR model is no longer considered in the following experiments on the NAB dataset for the same reason. The best-performing models on both datasets are LSTM-AE and SR-CNN.

In Table 2 we show the results of each model when performing a strict evaluation, where no delay is tolerated (delay=0). Thus, the model is penalized when it does not manage to detect the anomaly at the exact moment when it occurs. As aforementioned, AIOps models were evaluated in previous studies [39], [31], [5] with a maximum delay of 7 points. Therefore, in Figure 5 we show the results for the same models with a delay of up to 7 points.

From Figure 5 we can notice that although on the Yahoo dataset, a higher delay can significantly influence the performance of anomaly detection models, on the NAB dataset it has almost no impact. In the case of Yahoo, the explanation behind this result is that 73.13% of the time series belonging to the Yahoo dataset contain sequences of anomalies (group anomalies). When it comes to NAB, only 29.41% of the time series contain sequences of anomalies. Therefore, the dataset containing samples related to internet traffic (Yahoo) contains significantly more group anomalies compared to the dataset containing samples related to CPU utilization (NAB), which shows the different behavior of different operational datasets.

Another important observation is that in the case of Yahoo, the SR model performs better after a delay of 2 points compared to LSTM-AE. However, if we assess the two anomaly detection models according to how well they detect the exact moment anomalies occur, the difference in the F1-score is almost 10%. In our work, we aim to understand how well these methods perform when detecting the exact anomalous point is crucial. Therefore, for the rest of our experiments, we are solely considering a delay of 0 points when evaluating different anomaly detection models.

**Table 3: Assessment of anomaly detectors' robustness towards testing window size (Window Size Setup). The obtained results are obtained by averaging the metric over all time series included in one dataset and over the 5 random seeds.**

| | Model | F1-Score | Precision | Recall |
|---|---|---|---|---|
| **Yahoo** | LSTM-AE | 0.37 | 0.38 | 0.49 |
| | SR-CNN | 0.14 | 0.14 | 0.14 |
| **NAB** | LSTM-AE | 0.31 | 0.37 | 0.44 |
| | SR-CNN | 0.05 | 0.05 | 0.05 |

*6.1.2 Window Size Setup.* To fully answer our first research question, we need to understand whether models are robust towards the testing data size. We, therefore, employ the Window Size Setup described in Section 5.2.2 in the case of SR-CNN and LSTM-AE and compare it to the performance of the model when using the SoTA setting. However, as part of the functionality of SR, this model requires to be evaluated on a continuous time series. Therefore, this model could only be assessed for robustness on the sub-testing set immediately following the training set. From this experiment, we noticed that SR is not influenced by the size of the testing lengths and, thus, is robust to changes in the testing length. For SR-CNN and LSTM-AE we show results in Table 3.

Throughout our experiments, we noticed that the performance of LSTM-AE in both scenarios is the same. Therefore, the testing window size does not influence this anomaly detector. As it can be observed from Table 3, SR-CNN is drastically affected by being evaluated on smaller window sizes on both datasets. Its performance drops from an F1-score of 0.82 to 0.05 on the NAB dataset and from an F1-score of 0.57 to 0.14 for the Yahoo dataset.Thus, we consider that only SR and LSTM-AE are robust toward different testing sizes.

## 6.2 Full-History vs Sliding Window

In this set of experiments, we aim to answer our second research question and understand the differences in performance between the model that was never updated (S) compared to the models that were updated using a full-history (FH) vs a sliding window (SW) approach. In Table 4 we show the results of this experiment. The results displayed for SR-CNN are the results obtained through the testing window size setup experiment since this experiment implies a division of the entire testing data into smaller subsets and retraining the model accordingly.

From Table 4 we can notice that the performance of LSTM-AE is significantly improving when the model is updated over time. The highest performance is achieved when the model is updated using the sliding window (SW) technique. When it comes to the SR model, its performance degrades when being updated with the SW technique, but it does not statistically change when being updated with the FH technique. The SR-CNN performance benefits from being updated using the FH technique since it manages to find more anomalies (higher recall) and also to label more non-anomalies correctly (higher precision). In terms of SR, its performance degrades when old data is discarded.

## 6.3 Drift Detection based Retraining

*6.3.1 Drift Detection Results.* We apply the concept drift detector FEDD on a period basis (weekly for Yahoo and daily for NAB) for each time series from each dataset. Figure 6 shows the percentage of time series affected by concept drift according to FEDD during each period. Due to the lengths of time series corresponding to each dataset, the evaluation time series within the Yahoo dataset could be split into a maximum of five periods (five weeks), while those within the NAB dataset could be split into a maximum of nine periods (nine days).

From Figure 6 we can see that both for Yahoo and NAB, more than 50% of the time series are affected by concept drift in the first period (P1). Furthermore, in the case of NAB, during the second period (P2), there are more than 60% of the time series affected by concept drift according to FEDD.

*6.3.2 Blind vs Informed Retraining.* With this experiment, we aim to understand whether an anomaly detection model's performance can be preserved when retraining based on a drift detector's output (informed retraining) and how it compares to periodically retraining the model (blind retraining). Since the most significant benefits of

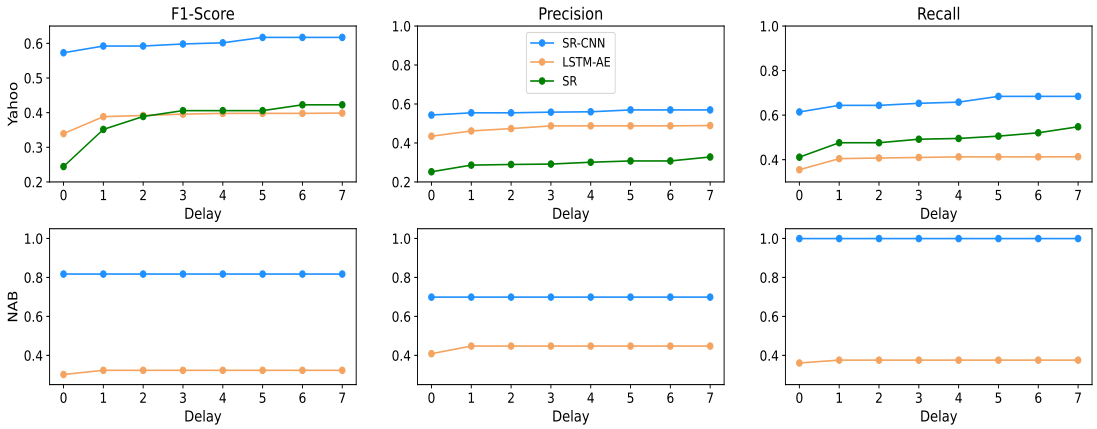



**Figure 5: Delay metric applied to F1-score, precision and recall on Yahoo and NAB.**

**Table 4: Results of anomaly detectors for blind model retraining. With bold we highlight the situation in which the difference in performance between the S model and the updated model (FH, SW) is significant. The displayed results are obtained by averaging the metric over all time series included in one dataset and over the 5 random seeds. (S-Static, FH-Full-History, SW-Sliding Window)**

| | | F1-Score | | | Precision | | | Recall | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Model | S | FH | SW | S | FH | SW | S | FH | SW |
| Yahoo | SR | 0.25 | 0.21 | **0.19** | 0.28 | **0.19** | **0.17** | 0.36 | **0.23** | **0.23** |
| | LSTM-AE | 0.37 | **0.42** | **0.46** | 0.38 | **0.42** | **0.46** | 0.49 | **0.53** | **0.59** |
| | SR-CNN | 0.14 | 0.10 | 0.14 | 0.14 | 0.09 | 0.17 | 0.14 | **0.26** | 0.22 |
| NAB | LSTM-AE | 0.31 | **0.34** | **0.37** | 0.37 | 0.37 | **0.45** | 0.44 | **0.46** | **0.48** |
| | SR-CNN | 0.05 | 0.07 | 0.03 | 0.05 | **0.15** | 0.07 | 0.05 | **0.13** | 0.04 |

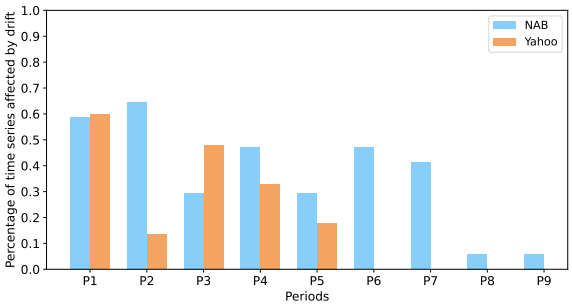



**Figure 6: Percentage of time series affected by concept drift during each period according to FEDD.**

periodic model retraining were observed in the case of LSTM-AE, this experiment is solely performed on this anomaly detection model. In addition, this experiment is performed with both data retraining techniques, full history (FH) and sliding window (SW).

Table 5 shows that overall the performance of informed model retraining is higher than the performance of a model that was never updated (static model). However, blind (periodic) model retraining generally achieves better results than informed model retraining.

**Table 5: Results of LSTM-AE when comparing blind with informed model retraining. The displayed results are obtained by averaging the metric over all time series included in one dataset and over the 5 random seeds. Bold means that informed retraining achieves significantly better results compared with the static model. *Italic* means that informed retraining achieves significantly similar results with blind retraining. (S-Static, FH-B/I-Full History Blind/Informed, SW-B/I-Sliding Window Blind/Informed.)**

| | | S | FH-B | FH-I | SW-B | SW-I |
|---|---|---|---|---|---|---|
| Yahoo | F1 | 0.37 | 0.42 | **0.40** | 0.46 | **0.41** |
| | Precision | 0.38 | 0.42 | **0.40** | 0.46 | ***0.45*** |
| | Recall | 0.49 | 0.53 | 0.50 | 0.59 | **0.55** |
| NAB | F1 | 0.31 | 0.34 | ***0.34*** | 0.37 | ***0.36*** |
| | Precision | 0.37 | 0.37 | **0.39** | 0.45 | ***0.45*** |
| | Recall | 0.44 | 0.46 | 0.45 | 0.48 | **0.46** |

Similar to the periodic model retraining experiment, the highest results were achieved using the sliding window approach.

From Table 5 it can be noticed that the informed model retraining achieved similar precision to the blind model retraining for both datasets when the model was retrained using the sliding window

approach. However, in all situations informed retraining results in lower recall compared with blind model retraining. When it comes to the full-history approach retraining technique, all performance metrics calculated for the blind model retraining are always higher than the performance metrics calculated for the informed model retraining on the Yahoo dataset. However, for the NAB dataset, the F1 score for informed model retraining is similar to the blind model retraining, while the precision is slightly higher.

## 7 DISCUSSIONS AND ANSWERS TO RESEARCH QUESTIONS

***Research Question 1:*** What is the performance of state-of-the-art anomaly detection models on operational data and how does the testing window size influence it?
***Answer 1:*** The more complex models (LSTM-AE and SR-CNN) perform significantly better on both operational datasets compared to the simpler models (FFT, PCI, SR). However, the evaluation window size significantly impacts SR-CNN but does not affect LSTM-AE or SR.

**State-of-the-Art Anomaly Detection:** Throughout our experiments, we noticed that PCI and FFT tended to label every point in the time series as an anomaly. This shows the complexity of operational data also described in [8], which was only captured by LSTM-AE and SR-CNN. Moreover, we noticed a significant difference of almost 10% between LSTM-AE and SR anomaly detectors when the delay metric is considered. We believe that the delay could be misleading for practitioners who want to use the anomaly detection models and, thus we recommend researchers to initially report the results with a delay of 0 and afterward perform an additional experiment with a variable delay.

One interesting finding of our experiment is that SR-CNN is highly influenced by the size of the data it is evaluated on. When performing the experiments to assess this aspect, we noticed a significant decrease in performance when comparing the SR-CNN tested on the original setup described in the paper [31] to a setting in which the testing set is smaller. However, in both settings, we respected the original SR-CNN architecture in which the training and testing datasets are separated before performing the time series spectral transformation. This could be due to the functionality of SR-CNN since this model relies on decomposing time series into another domain and extracting features that are further used to train a CNN. Therefore, when shortening the time series, these features might not be significant enough for CNN to properly distinguish between anomalies and non-anomalies.

**Implications for Practitioners:** In the light of our findings we strongly recommend practitioners that when performing anomaly detection model selection they should consider a delay of 0. Thereafter, they can adjust the delay according to the requirements of their application. Given our findings regarding the performance drop of SR-CNN when a smaller testing size is employed, we further suggest that during model selection, AIOps practitioners should consider its robustness to different time series lengths and how often the inference is going to be performed. For instance, when identifying anomalies in a time series with a big granularity (e.g. 1 hour) that needs to be evaluated daily, employing an anomaly detector that is sensitive to short testing sets could lead to erroneous results. Moreover, sometimes incidents could occur and data collection could be interrupted, which can also shorten the available testing data.

***Research Question 2:*** What is the impact of the two model retraining techniques from the perspective of the retraining data (full-history vs sliding window approach)?
***Answer 2:*** The sliding-window approach is beneficial to models that learn from the time domain (LSTM-AE) but can impact the performance of models decompose the time series into another domain (SR and SR-CNN). Retraining using the full-history approach achieved higher performance than retraining in most of the cases.

We empirically prove that the LSTM-AE anomaly detector benefits from being periodically updated in both datasets, being able to identify more anomalies and distinguish better between anomalous and non-anomalous points. The performance of this model is higher when employing the sliding window approach than the full-history approach. This could show that old samples are no longer relevant to the current behavior of the data and retraining the LSTM-AE only on the most recent data eliminates non-relevant data points. When it comes to SR, which is not affected by the window size as shown in previous experiments, we noticed that a sliding window approach significantly impacts its performance. This can be a consequence of lowering the size of the time series and discarding important spectral information.

**Implications for Practitioners:** Based on our findings, periodically retraining anomaly detection models can significantly improve their performance over time. Thereby, AIOps practitioners should consider periodic model maintenance after deployment as part of the anomaly detection model lifecycle. Furthermore, we noticed that models that identify anomalies from the original time series (LSTM-AE) benefit from a sliding-window retraining approach, while the ones that detect anomalies by transforming the time series in other domains (SR) benefit from a full-history approach. Therefore, the retraining technique should be chosen according to the employed anomaly detector.

***Research Question 3:*** What is the impact of the two model retraining techniques from the perspective of the retraining frequency (blind vs informed retraining)?
***Answer 3:*** Blind (periodic) retraining usually achieves higher performance than informed retraining. However, when a sliding window retraining technique is employed, the precision is statistically similar.

This experiment shows that retraining a model based on the output of a drift detector (FEDD) achieves a higher performance than a model that was never updated. This shows that the informed retraining setting is a promising research path and the FEDD drift detector could be employed as an anomaly detection model degradation monitoring tool. However, a model that is retrained periodically achieves slightly higher performance than a model retrained based on the output of a drift detector. This could be caused by the fact that not all drifts are captured by FEDD, which leaves room for improvement of concept drift detectors for time series.

We noticed from our experiments that for both datasets the precision obtained through blind retraining is comparable to the one obtained through informed retraining when a sliding window approach is employed. This shows that retraining based on a drift detector helps reduce the number of false alarms triggered by an anomaly detector as much as periodic retraining does. However, with a periodic model retraining more anomalies are captured compared to the blind model retraining given the higher recall.

**Implications for Practitioners:** Since we show the potential of having a model retrained based on the output of a drift detector, we argue that these monitoring tools can be employed in real-world anomaly detection model maintenance pipelines. Furthermore, to understand the suitability of the employed drift detector and the effects of retraining based on its output, we suggest that practitioners allocate a period to evaluate the model quality over time. Within this period AIOps practitioners should constantly investigate the effects of different retraining techniques (blind vs informed and sliding-window vs full-history) and understand which maintenance technique is the most suitable for anomaly detection AIOps solutions with respect to its corresponding costs (labeling/labor costs, etc). Within this period multiple drift detectors can be evaluated .

# 8 CONCLUSIONS AND FUTURE WORK

In this paper, we investigate different maintenance techniques for popular anomaly detection models on two AIOps domains, internet traffic and CPU utilization. From a retraining frequency perspective, we analyzed *blind retraining*, retraining the model periodically, and *informed retraining*, retraining the model when a drift detector indicates it. With this study, we assess the potential of using drift detectors as a quality monitoring tool for anomaly detection models. From a retraining data perspective, we experimented with *full-history approach*, constantly enriching the training set, and *sliding window approach*, discarding old samples from the training set.

We began our study by replicating the most popular anomaly detection model and testing their performances on the two AIOps datasets. We observed that some models could not detect anomalies in this type of data, while others were sensitive to the length of the testing data size. Our study shows that generally, an updated anomaly detector achieves higher performance than an anomaly detector that was never updated. We observed that the sliding-window approach benefits models that identify anomalies from the original time series while the full-history approach benefits models that transform time series into another domain to detect anomalies. Furthermore, we empirically demonstrated that the performance of an anomaly detector retrained based on the output of a drift detector leads to better performance when compared to an anomaly detector that was never updated. This shows that drift detection-based model update is a promising research path and can be the beginning of automated model maintenance pipelines. In light of our findings, we offer recommendations to AIOps practitioners regarding anomaly detection model selection and assessment, as well as identifying the most appropriate model maintenance techniques.

Our work was limited by the availability of open-source AIOps datasets for time series anomaly detection. We, therefore strongly encourage AIOps practitioners to release more datasets that researchers can use as benchmarks when developing new models. Moreover, we also believe that research should focus on understanding and reporting the labeling process of experts. This can offer anomaly detection researchers more transparency and confidence in the data quality.

In our study, we solely targeted anomaly detection for CPU utilization and internet traffic data. A promising research path is analyzing the potential of a concept drift detection-based framework in other AIOps applications, such as job and disk failure prediction [22]. Furthermore, we noticed that the performance of retraining an anomaly detector based on FEDD is lower than periodically updating the anomaly detector. This could show that FEDD does not capture all drifts that lead to performance degradation. Therefore, future work should consider investigating more drift detectors suitable for time series beyond FEDD and designing more accurate drift detectors for time series.

# 9 THREATS TO VALIDITY

## 9.1 External Validity

An external threat to validity derived from our study is the generalizability of our results towards different AIOps solutions. In our study, we targeted detecting anomalies in solely two AIOps domains, namely, internet traffic and CPU utilization due to public availability. Moreover, in this study, we relied on the provided anomalous labels corresponding to each dataset. The anomaly detection models depend on the datasets they are applied to. Although our datasets contain solely real-world data, the data used to e.g. predict incidents using AIOps solutions might be different than the one we employed. Therefore, the anomaly detection AIOps solutions lifecycle should be explored for other AIOps domains.

In our study, we included a variety of popular anomaly detection models, which are different from each other in terms of the technique they use to detect anomalies. Since we noticed that maintenance techniques are model-dependent when employing other models they need to be verified accordingly.

## 9.2 Internal Validity

When computing the static model, we used the first half of each time series from each dataset as training and the second half of each time series as testing as previous work [31], [5].

We partitioned our testing data into periods of different sizes (weekly and daily) similar to prior work [22], [23]. We did not consider periods smaller than one day since not enough data is captured in a time shorter than one day. Periods bigger than one week were not considered given the size of the time series.

## 9.3 Construct Validity

We replicated anomaly detection models used in previous studies [42], [31], [30]. We employed hyperparameters used by previous studies. In the case when the hyperparameters were not provided, we used hyperparameters that led to similar results to what was reported in previous studies. In the case of LSTM-AE, we used the maximum reconstruction error on the training set as our anomaly threshold, which is derived from each time series. We employed multiple performance metrics to evaluate and interpret the model's outcome.

# REFERENCES


[1] Tao Ban, Ndichu Samuel, Takeshi Takahashi, and Daisuke Inoue. 2021. Combat security alert fatigue with ai-assisted techniques. In *Cyber Security Experimentation and Test Workshop* (CSET '21). Association for Computing Machinery, Virtual, CA, USA, 9–16. ISBN: 9781450390651. DOI: 10.1145/3474718.3474723.

[2] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. 2022. From concept drift to model degradation: an overview on performance-aware drift detectors. *Knowledge-Based Systems*, 245, 108632. DOI: https://doi.org/10.1016/j.knosys.2022.108632.

[3] Mirela Madalina Botezatu, Ioana Giurgiu, Jasmina Bogojeska, and Dorothea Wiesmann. 2016. Predicting disk replacement towards reliable data centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD '16). San Francisco, California, USA, 39–48. ISBN: 9781450342322.

[4] Rodolfo C. Cavalcante, Leandro L. Minku, and Adriano L. I. Oliveira. 2016. Fedd: feature extraction for explicit concept drift detection in time series. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 740–747.

[5] Run-Qing Chen, Guang-Hui Shi, Wan-Lei Zhao, and Chang-Hui Liang. 2021. A joint model for it operation series prediction and anomaly detection. *Neurocomputing*, 448, 130–139.

[6] Yujun Chen et al. 2019. Outage prediction and diagnosis for cloud service systems. In *The World Wide Web Conference* (WWW '19). Association for Computing Machinery, San Francisco, CA, USA, 2659–2665. ISBN: 9781450366748. DOI: 10.1145/3308558.3313501.

[7] Yujun Chen et al. 2019. Outage prediction and diagnosis for cloud service systems. In *The World Wide Web Conference* (WWW '19). San Francisco, CA, USA, 2659–2665. ISBN: 9781450366748.

[8] Qian Cheng, Doyen Sahoo, Amrita Saha, Wenzhuo Yang, Chenghao Liu, Gerald Woo, Manpreet Singh, Silvio Saverese, and Steven C H Hoi. 2023. AI for IT operations (AIOps) on cloud platforms: reviews, opportunities and challenges, (Apr. 2023). arXiv: 2304.04661 [cs.LG].

[9] Yingnong Dang, Qingwei Lin, and Peng Huang. 2019. Aiops: real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 4–5.

[10] Fengqian Ding and Chao Luo. 2019. The entropy-based time domain feature extraction for online concept drift detection. *Entropy*, 21, 12. DOI: 10.3390/e21121187.

[11] João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46, 4, Article 44, 37 pages. DOI: 10.1145/2523813.

[12] Gustavo H. F. M. Oliveira, Rodolfo C. Cavalcante, George G. Cabral, Leandro L. Minku, and Adriano L. I. Oliveira. 2017. Time series forecasting in the presence of concept drift: a pso-based approach. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 239–246. DOI: 10.1109/ICTAI.2017.00046.

[13] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. 2016. Experience report: system log analysis for anomaly detection. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, 207–218. DOI: 10.1109/ISSRE.2016.21.

[14] Ali Imran Jehangiri, Ramin Yahyapour, Philipp Wieder, Edwin Yaqub, and Kuan Lu. 2014. Diagnosing cloud performance anomalies using large time series dataset analysis. In *2014 IEEE 7th International Conference on Cloud Computing*, 930–933. DOI: 10.1109/CLOUD.2014.129.

[15] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. 2017. Time-series extreme event forecasting with neural networks at uber. In *International conference on machine learning*. Vol. 34, 1–5.

[16] Heng Li, Tse-Hsun Chen, Ahmed E. Hassan, Mohamed Nasser, and Parminder Flora. 2018. Adopting autonomic computing capabilities in existing large-scale systems. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 1–10.

[17] Xiaoyun Li, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba Yu. 2020. Swisslog: robust and unified deep learning based log anomaly detection for diverse faults. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, 92–103. DOI: 10.1109/ISSRE5003.2020.00018.

[18] Yangguang Li, Zhen Ming (Jack) Jiang, Heng Li, Ahmed E. Hassan, Cheng He, Ruirui Huang, Zhengda Zeng, Mian Wang, and Pinan Chen. 2020. Predicting node failures in an ultra-large-scale cloud computing platform: an aiops solution. *ACM Transactions on Software Engineering and Methodology*, 29, 2, Article 13, 24 pages.

[19] Yangguang Li, Zhen Ming (Jack) Jiang, Heng Li, Ahmed E. Hassan, Cheng He, Ruirui Huang, Zhengda Zeng, Mian Wang, and Pinan Chen. 2020. Predicting node failures in an ultra-large-scale cloud computing platform: an aiops solution. *ACM Trans. Softw. Eng. Methodol.*, 29, 2, Article 13, 24 pages.

[20] Meng-Hui Lim, Jian-Guang Lou, Hongyu Zhang, Qiang Fu, Andrew Beng Jin Teoh, Qingwei Lin, Rui Ding, and Dongmei Zhang. 2014. Identifying recurrent and unknown performance issues. In *2014 IEEE International Conference on Data Mining*, 320–329. DOI: 10.1109/ICDM.2014.96.

[21] Qingwei Lin et al. 2018. Predicting node failure in cloud service systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (ESEC/FSE 2018). Association for Computing Machinery, Lake Buena Vista, FL, USA, 480–490. ISBN: 9781450355735.

[22] Yingzhe Lyu, Heng Li, Mohammed Sayagh, Zhen Ming (Jack) Jiang, and Ahmed E. Hassan. 2021. An empirical study of the impact of data splitting decisions on the performance of aiops solutions. *ACM Trans. Softw. Eng. Methodol.*, 30, 4, Article 54, 38 pages.

[23] Yingzhe Lyu, Gopi Krishnan Rajbahadur, Dayi Lin, Boyuan Chen, and Zhen Ming (Jack) Jiang. 2021. Towards a consistent interpretation of aiops models. *ACM Trans. Softw. Eng. Methodol.*, 31, 1, Article 16, 38 pages.

[24] N. Nahar, H. Zhang, G. Lewis, S. Zhou, and C. Kastner. 2023. A meta-summary of challenges in building products with ml components – collecting experiences from 4758+ practitioners. In *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*. IEEE Computer Society, Los Alamitos, CA, USA, 171–183. DOI: 10.1109/CAIN58948.2023.00034.

[25] Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. 2022. Collaboration challenges in building ml-enabled systems: communication, documentation, engineering, and process. In *Proceedings of the 44th International Conference on Software Engineering* (ICSE '22). Association for Computing Machinery, Pittsburgh, Pennsylvania, 413–425. ISBN: 9781450392211. DOI: 10.1145/3510003.3510209.

[26] Paolo Notaro, Jorge Cardoso, and Michael Gerndt. 2021. A systematic mapping study in aiops. In *Service-Oriented Computing – ICSOC 2020 Workshops*. Hakim Hacid, Fatma Outay, Hye-young Paik, Amira Alloum, Marinella Petrocchi, Mohamed Reda Bouadjenek, Amin Beheshti, Xumin Liu, and Abderrahmane Maaradji, (Eds.), 110–123. ISBN: 978-3-030-76352-7.

[27] Lorena Poenaru-Olaru, Luis Cruz, Jan S. Rellermeyer, and Arie van Deursen. 2023. Maintaining and monitoring aiops models against concept drift. In *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, 98–99. DOI: 10.1109/CAIN58948.2023.00024.

[28] Lorena Poenaru-Olaru, Luis Cruz, Arie van Deursen, and Jan S. Rellermeyer. 2022. Are concept drift detectors reliable alarming systems? - a comparative study. In *7th Workshop on Real-time Stream Analytics, Stream Mining, CER/CEP & Stream Data Management in Big Data*.

[29] Oleksandr Provotar, Yaroslav M.) Linder, and Maksym Veres. 2019. Unsupervised anomaly detection in time series using lstm-based autoencoders. *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, 513–517.

[30] Faraz Rasheed, Peter Peng, Reda Alhajj, and Jon Rokne. 2009. Fourier transform based spatial outlier mining. In *Proceedings of the 10th International Conference on Intelligent Data Engineering and Automated Learning*, 317–324. ISBN: 3642043933.

[31] Hansheng Ren et al. 2019. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (KDD '19). Association for Computing Machinery, Anchorage, AK, USA, 3009–3017. ISBN: 9781450362016. DOI: 10.1145/3292500.3330680.

[32] Andrea Rosà, Lydia Y. Chen, and Walter Binder. 2015. Catching failures of failures at big-data clusters: a two-level neural network approach. In *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, 231–236. DOI: 10.1109/IWQoS.2015.7404739.

[33] Nosayba El-Sayed, Hongyu Zhu, and Bianca Schroeder. 2017. Learning from failure across multiple clusters: a trace-driven approach to understanding, predicting, and mitigating job terminations. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 1333–1344. DOI: 10.1109/ICDCS.2017.317.

[34] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly detection in time series: a comprehensive evaluation. *Proc. VLDB Endow.*, 15, 1779–1797.

[35] Kamran Shaukat, Talha Mahboob Alam, Suhuai Luo, Shakir Shabbir, Ibrahim A. Hameed, Jiaming Li, Syed Konain Abbas, and Umair Javed. 2021. A review of time-series anomaly detection techniques: a step to future perspectives. In *Advances in Information and Communication*. Springer International Publishing, 865–877.

[36] Nidhi Singh and Craig Olinsky. 2017. Demystifying numenta anomaly benchmark. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 1570–1577. DOI: 10.1109/IJCNN.2017.7966038.

[37] Li Wei, Nitin Kumar, Venkata Lolla, Eamonn J. Keogh, Stefano Lonardi, and Chotirat Ratanamahatana. 2005. Assumption-free anomaly detection in time series. In *Proceedings of the 17th International Conference on Scientific and Statistical Database Management* (SSDBM'2005). Lawrence Berkeley Laboratory, Santa Barbara, CA, 237–240.

[38] Renjie Wu and Eamonn J. Keogh. 2023. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, 35, 3, 2421–2429. DOI: 10.1109/TKDE.2021.3112126.

[39] Haowen Xu et al. 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. *WWW '18: Proceedings of the 2018 World Wide Web Conference*, (Feb. 2018).

[40] Haowen Xu et al. 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference* (WWW '18). International World Wide Web Conferences Steering Committee, Lyon, France, 187–196. ISBN: 9781450356398. DOI: 10.1145/3178876.3185996.

[41] Yong Xu et al. 2018. Improving service availability of cloud systems by predicting disk error. In *Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association, USA, 481–493. ISBN: 9781931971447.

[42] Yu Yufeng, Yuelong Zhu, Shijin Li, and Dingsheng Wan. 2014. Time series outlier detection based on sliding window prediction. *Mathematical Problems in Engineering*, 2014, (Oct. 2014).