Extended Framework and Evaluation for Multivariate Streaming Anomaly Detection with Machine Learning

Andreas Koch*†, Michael Petry*†, Martin Werner†

*Airbus Defence and Space GmbH,
Telecom and Navigation Processing Germany

†Technical University of Munich,
School of Engineering and Design,
Professorship for Big Geospatial Data Management

***Abstract*—Streaming anomaly detection in multivariate time series is an important problem relevant for automatic monitoring of various devices. This paper tackles the problem of streaming anomaly detection by extending a framework for the purpose of incorporating model-based approaches and evaluating previously uncombined methods for a total number of 26 distinct machine-learning-based algorithms. The framework identifies four fundamental components inherent to many streaming anomaly detection algorithms and one or more methods are presented for each component. It is found that a simple and computationally less expensive strategy for detecting concept drift yields almost identical results to the "KSWIN" strategy, when applied to measuring concept drift in a training set relevant for training a machine learning model. A secondary experiment supports the effectiveness of finetuning a machine learning model after the detection of concept drift for the purpose of detecting anomalies.**

***Index Terms*—Machine learning, anomaly detection, stream mining, online learning, multivariate time series**

I. Introduction

Time series are an increasingly important format of data and their analysis offers many insights. The subfield of anomaly detection in time series also shows much research interest by the number of publications. With higher automation and autonomy in vehicles and the proliferation of edge computing devices, real time anomaly detection on streams of data is very relevant. This paper aims to build on and extend the "SAFARI" framework [1] to formalize every component of a streaming anomaly detection algorithm. Thereby, a selection of algorithms will be presented, all of which include a machine learning (ML) model.

The differentiation of components of a streaming anomaly detection algorithm into data representation, learning strategy, nonconformity measure and anomaly scoring of SAFARI [1] will be adapted in order to accommodate model-based approaches. More specifically, the "reference group", consisting of a selection of instances deemed "normal", is extended to include model parameters that are allowed to change. The overall idea to this end is observing the changing reference group to detect concept drift. Once concept drift has been detected, the ML model is trained on the most recent reference group to maintain anomaly detection capability.

The experiments presented in this work encompass five different ML models being evaluated with different learning strategies. In total, this results in 26 different streaming anomaly detection algorithms. Evaluations are performed on a selection of multivariate time series anomaly detection benchmarks including Daphnet [2], Exathlon [3] and SMD [4]. To this end, the NAB score [5], volume under the surface (VUS) [6] and area under the precision-recall curve are used as metrics. The significance of training the ML model on the most recent reference group, once concept drift has been detected, is analyzed.

Our contribution can be summarized as following:

- We extend the SAFARI framework [1] to model-based approaches and
- provide the first evaluation of previously uncombined methods for multivariate streaming anomaly detection to a total number of 26 distinct algorithms.

In Section II, relevant previous work will be presented to provide important background knowledge before the adapted framework is defined in Section III. Section IV then continues to define and explain all methods that are implemented and Section V presents the experiments and the obtained results. Finally, Section VI summarizes this paper and gives an outlook on future research directions.

II. Related Work

Previous anomaly detection approaches can be classified into five distinct categories: statistical, similarity-based,

This activity has been funded by the ESA General Support Technology Programme within the project Machine Learning for Telecom Satellites (MaLeTeSa). Corresponding Author: Andreas Koch (Email: andreas.c.koch@airbus.com)

prediction-based, information theoretic, and grammar-based. Statistical approaches assume the time series originates from a known or unknown stochastic process. Most often, a statistical model is fitted to the normal data and a statistical test is applied on unseen data instances to determine the probability of them being generated by the model [7]. Note that this also includes several regression models, such as ARIMA. Similarity-based approaches require the computation of a distance metric between subsequences of the time series. Thereby, the nearest neighbor, density or distance to cluster centroids are used as indicators of normalness. Prediction-based approaches encompass both model-based regression and classification. The former includes forecasting, reconstruction or predicting custom nonconformity scores, such as in the isolation forest algorithm [8] or the hierarchical temporal model [9]. The latter includes all models that learn features in order to directly predict whether a subsequence is anomalous. Information theoretic approaches calculate the information contained in the overall dataset via different information theoretic measures, such as Kolmogorov complexity and entropy [7]. As it is assumed that anomalies induce irregularities and therefore increase the information content of the dataset, the goal is to find a set of instances, which when removed, lead to the greatest reduction of information. Lastly, grammar-based approaches discretize subsequences of a time series and apply grammar induction in order to identify substrings, which are rarely used in the grammar rules, as potential anomalies [10].

Another aspect to anomaly detection approaches is their capability for online processing of data streams. Many approaches have been adapted to improve this capability, such as in PCB-iForest [11] or in an online version of the ARIMA model [12]. Calikus et al [1] implement multiple statistical and nearest-neighbor-based approaches, while Munir et al [13] offer a comparison of statistical, nearest-neighbors-based and prediction-based deep learning algorithms. However, both evaluations are performed on univariate datasets only. In contrast to this, Mulinka et al [14] evaluate four ML-based approaches on a multivariate network traffic dataset, under consideration of the possibility for concept drift to occur. FuseAD [15] combines an ARIMA model with a prediction-based CNN as a streaming anomaly detection approach, although it does not take concept drift into account. Basheer et al [16] develop an online density-based approach to anomaly detection, which clusters data samples and updates clusters at every time step, thus being able to handle concept drift. A similar approach is pursued with SAND [17]. It updates clusters at every time step and calculates new centroids after two clusters are merged, without having to keep cluster elements in memory. Ribeiro et al [18] use the similarity-based OutlierDenStream algorithm to detect intrusions in software defined networks (SDN). DeepStream [19] leverages the learnt representations of an autoencoder to cluster them in an online manner. The method is however struggling with concept drift due to an overabundance of clusters. The density of a sample is estimated by randomized space trees in RS-Forest [20] to identify low density samples as anomalies. Ko and Comuzzi [21] present an online anomaly detection algorithm for categorical data and Belacel et al [22] use a prediction-based approach to reconstruct an adaptive sliding window (ADWIN). They detect concept drift if the rate of detected anomalies crosses a threshold. Consequently, the adaptive window is shrinked and the model is fine-tuned on the new window. Lastly, Wang et al [23] implement an online K-means algorithm in Apache Spark, which keeps a fixed number of clusters with clusters being rebuilt at every time step based on a sliding window.

III. Extended Framework

In this section, we will adapt the components of anomaly detection algorithms, defined in [1], to accommodate for model-based algorithms, which rely on learning a set of parameters from the data.

A. Adapted fundamental tasks

Definition III.1 (Data representation)**.** Given a stream $S \coloneqq \{s_1, ..., s_t \mid s_i \in \mathbb{R}^N\}$, the values of the last w time steps are transformed into a feature vector x_t by a data representation function D,

$$x_t \coloneqq D(s_{t-w+1}, ..., s_t) \tag{1}$$

with $x_t \in \mathbb{R}^d$.

The feature vector length d thereby remains unspecified to allow a wide range of representations.

The learning strategy defined in SAFARI [1] has the goal of compiling a reference group of feature vectors, which is updated at every time step. In this paper, the definition will be generalized to a set of reference parameters θ. This change in definition allows for the inclusion of model parameters, which are also to be updated at every time step. In the special case that θ consists of only feature vectors, the original definition is recovered.

Definition III.2 (Learning strategy)**.** Let $\theta_0 \in \mathbb{R}^{l_0}$ be a set of reference parameters at time step 0. A learning strategy L is a function that updates θ at every time step,

$$\theta_t = L(x_t, \theta_{t-1}). \tag{2}$$

This definition allows θ_i to be of a different length l_i for every time step, which is important for methods relying on accumulating feature vectors. Accordingly, the nonconformity measure from Calikus et al [1] is changed to consider the reference parameters instead of the reference group.

Definition III.3 (Nonconformity measure)**.** Given the reference parameters θ_t and feature vector x_t, a nonconformity measure A is a function that produces a nonconformity score

$$a_t = A(x_t, \theta_t), \tag{3}$$

which is representative of the "strangeness" of feature vector x_t, according to the reference parameters θ_t and measure A.

Finally, a window of k nonconformity scores are considered in the calculation of the anomaly score f_t.

Definition III.4 (Anomaly scoring)**.** An anomaly scoring function F maps k nonconformity scores $\{a_{t-k+1}, ..., a_t\}$ to the final anomaly score,

$$f_t = F(a_{t-k+1}, a_t). \tag{4}$$

IV. Methods

This section will introduce one or more methods for every task previously mentioned. Thereby, the reference parameters for the ML models will include the model parameters, as well as a training set of stream vectors,

$$\theta_i = \{\theta_{model}, R_{train,i}\}. \quad \forall i \in 0, ..., t \tag{5}$$

A. Data representation

As machine learning models learn various representations internally, the only data representation used in this paper will consist of the past w stream vectors,

$$x_t = [s_{t-w+1}, s_{t-w+1}, ..., s_t]^T.$$

Accordingly, a feature vector is of the dimensionality $x_t \in \mathbb{R}^{w \times N}$.

B. Learning strategy

While the models and their parameter update mechanism specify how the model parameters θ_{model} should be updated based on the training set of stream vectors R_{train}, the role of a learning strategy in this paper is to decide

1) how and when the training set should be updated and
2) when the **fine-tuning** process should be executed in order to update the model parameters.

These two aspects will be addressed independently as Task 1 and Task 2 in the following. The trivial overall learning strategy naturally is keeping the training set fixed without any fine-tuning.

$$\theta = \{\theta_{\text{model}}, R_{\text{train}}\} = \text{const} \quad \forall i \in 0, ..., t$$

As the training set is equivalent to the reference group defined in SAFARI [1], all of their presented learning strategies for updating the reference group are applicable to the training set. This includes the sliding window (SW), uniform reservoir (URES) and anomaly-aware reservoir (ARES). The **sliding window** [1] keeps the m most recent feature vectors at all times,

$$R_{\text{train},t} = \begin{cases} R_{\text{train},t-1} - \{x_{t-m}\} + \{x_t\} & \text{if } t > m, \\ R_{\text{train},t-1} + \{x_t\} & \text{otherwise.} \end{cases}$$

Similarly, the **uniform reservoir** [1] strategy also keeps a set of m feature vectors as its training set, although the newest feature vector is only added with a uniformly drawn probability $P \in [0, 1]$ being lower than $\frac{m}{t}$,

$$R_{\text{train},t} = \begin{cases} R_{\text{train},t-1} + \{x_t\} & \text{if } t \leq m \\ R_{\text{train},t-1} - \{x^*\} + \{x_t\} & t > m, P < \frac{m}{t} \\ R_{\text{train},t-1} & \text{otherwise,} \end{cases}$$

and the disregarded feature vector x^* is chosen randomly. Lastly, the **anomaly-aware reservoir** [1] considers a feature vector's anomaly score and the anomaly scores of all feature vectors in the training set in order to retain the most normal feature vectors. Let p_t be a "priority" for x_t,

$$p_t = u^{\frac{\lambda_1}{\exp -\lambda_2 f_t}},$$

with $u \in [0, 1]$ being drawn uniformly and $\lambda_1, \lambda_2 > 0$. The function is monotonically decreasing and thus feature vectors with higher priorities are more "normal", according to their anomaly scores. Although, the randomly drawn base u prevents the reservoir from converging to a fixed set of feature vectors. For this paper's experiments, the parameters were further restricted to $u \in [0.7, 0.9]$ and $\lambda_1 = \lambda_2 = 3$. With the helper function

$$c(ps, p_t) = \underset{p_j}{argmin}\{p \in ps | p < p_t\},$$

the priorities of a training set are kept in

$$ps = \begin{cases} ps + \{(p_t, t)\} & \text{if } t \leq m \\ ps - \{(p_i, i)\} + \{(p_t, t)\} & \exists\ i = c(ps, p_t) \\ ps & \text{otherwise.} \end{cases}$$

Resulting is a training set update rule that replaces a feature vector x_i, if its priority is found to be lower than the priority of the most recent feature vector x_t,

$$R_{\text{train},t} = \begin{cases} R_{\text{train},t-1} + \{x_t\} & \text{if } t \leq m \\ R_{\text{train},t-1} - \{x_i\} + \{x_t\} & \exists\ i = c(ps, p_t) \\ R_{\text{train},t-1} & \text{otherwise.} \end{cases}$$

Regarding Task 2, a simple solution is to retrain the model parameters after a regular time interval. This will be the **"regular fine-tuning"** strategy, with fine-tuning occurring after every m time steps

$$\theta_{model,t} = \begin{cases} \theta_{model,t-1} - \mathit{grads} & \text{if } t \pmod m = 0 \\ \theta_{model,t-1} & \text{otherwise,} \end{cases}$$

with *grads* referring to all iterative gradient updates summed together and *Opt* being an optimizer function,

$$\mathit{grads} \coloneqq \sum_{\forall x_j \in R_{\text{train},\,t-1}} \mathit{Opt}\left(\frac{\partial \mathcal{L}(x_j)}{\partial \theta_{model,t-1,j}}\right).$$

The second strategy for Task 2, **"μ/σ-Change"**, will involve monitoring the mean and standard deviation of the training set and fine-tuning the ML model if one of these measures exceed a certain threshold. Keeping a running mean as

$$\mu_t = \begin{cases} \mu_i & \text{if } t = i, \\ \mu_{t-1} + \frac{1}{N}(x_t - x^*) & \text{for } R_{\text{train},t-1} \setminus \{x^*\} \cup \{x_t\}, \\ \frac{N-1}{N}\mu_{t-1} + \frac{1}{N}x_t & \text{for } R_{\text{train},t-1} \cup \{x_t\}, \\ \mu_{t-1} & \text{if } R_{\text{train},t} = R_{\text{train},t-1} \end{cases}$$

and a running standard deviation σ_t, as well as the mean μ_i and standard deviation σ_i of the training set used for the last

training session at time step i, fine-tuning will occur if the difference in means exceeds σ_i or if the standard deviation changes to a factor of 2,

$$c(\mu_i, \sigma_i, \mu_t, \sigma_t) = (d(\mu_i, \mu_t) > \sigma_i) \vee (\frac{1}{2}\sigma_i > \sigma_t > 2\sigma_i)$$

$$\theta_{model,t} = \begin{cases} \theta_{model,t-1} - \textit{grads} & \text{if } c(\mu_i, \sigma_i, \mu_t, \sigma_t), \\ \theta_{model,t-1} & \text{otherwise.} \end{cases}$$

The third strategy for Task 2 is called **"KSWIN"** [24] and it applies the two-sample Kolmogorov-Smirnov (KS) test to detect significant changes between two sets of data. The two-sample KS test is a non-parametric test for two random variables and it tests whether their underlying distributions are the same. In the case of two-dimensional feature vectors $\{x_1, ..., x_t\}$, two training sets at different time steps $R_{\text{train},i}$ and $R_{\text{train},t}$ and their corresponding probability distributions $F_i(x)$ and $F_t(x)$, both statistical hypotheses can be written as

$$\textbf{Null hypothesis } N_0 : F_i(x) = F_t(x),$$
$$\textbf{Alternative hypothesis } N_1 : F_i(x) \neq F_t(x).$$

The corresponding test statistic is defined as

$$dist_{i,t} = \sup_x |F_{i,r_i}(x) - F_{t,r_t}(x)|,$$

with a critical value $c(\alpha) = \sqrt{\ln \frac{2}{\alpha}}$. $F_{i,r_i}(x)$ and $F_{t,r_t}(x)$ denote the empirical cumulative distribution functions of each training set, where r_i and r_t are the respective training set sizes. The null hypothesis can be rejected to a significance level of α, if the test statistic fulfills

$$dist_{i,t} > c(\alpha)\sqrt{\frac{r_i + r_t}{r_i r_t}}.$$

When dealing with multidimensional data, Raab et al [24] perform a KS test on every dimension individually. For simplicity, this approach will be adopted as well for our implementation and the test will be performed for every channel dimension $j \in \{1, .., N\}$ individually. Furthermore, the required distance to trigger the test decreases with larger training set sizes, leading to many false positives when repeatedly testing. For this reason, [24] applies a correction to obtain a new significance level of $\alpha^* = \frac{\alpha}{r}$ in the case of an equal length of training sets $r = r_i = r_t$. The model parameters will be updated if the difference between the training set at time step t differs from the training set at time step i to a significance level of α according to the KS test,

$$\theta_{model,t} = \begin{cases} \theta_{model,t-1} - \textit{grads} & \text{if } dist_{i,t} > c(\alpha)\sqrt{\frac{r_i+r_t}{r_i r_t}}, \\ \theta_{model,t-1} & \text{otherwise.} \end{cases}$$

C. Machine learning models

As the machine learning models' predictions are required for the nonconformity scores, this section will introduce all ML models before moving on to the nonconformity scores.

The first model is going to be an online version of the **autoregressive integrated moving average** (ARIMA) model [12]. It consists of an autoregression (AR) model, where a data point in a time series is the result of a linear combination of the past q data points and a zero-mean random noise term ϵ_t. It further includes a moving average (MA) model, which is a linear regression of the past q' noise terms. In case the polynomial of the AR model has a unit root of multiplicity d, a linear regression of both AR and MA is able to emulate a non-stationary process. The resulting ARIMA(q, d, q') model can be expressed as

$$\nabla^d s_t = \sum_{i=1}^{q'} \beta_i \epsilon_{t-i} + \sum_{i=1}^{q} \alpha_i \nabla^d s_{t-i} + \epsilon_t,$$

with the differencing operator $\nabla s_t = s_t - s_{t-1}$. As in the adaptation for an online learning problem in [12], this model is approximated by an ARIMA(q+m, d, 0) model without any noise terms to give the prediction

$$\tilde{s}_t(\gamma_t) = \sum_{i=1}^{q+m} \gamma_{t,i} \nabla^d s_{t-i} + \sum_{i=0}^{d-1} \nabla^i s_{t-1},$$

with $\gamma_t \in \mathbb{R}^{q+m}$ as the only model parameter $\theta_{model} = \{\gamma_t\}$. The factors resulting from applying the differencing operator multiple times can be calculated via the binomial coefficient as

$$\nabla^d s_t = \sum_{i=0}^{d} (-1)^i \binom{d}{i} s_{t-i}.$$

As it stands, the online ARIMA model covers only univariate time series and assumes the data points s_i to be consecutive, originating from the same underlying process. As the optimization process via gradient descent is iterative, the model can in fact be applied to a window of consecutive data points. Our data representation x_t fits this description and its length restricts the model parameters as $w = q + m + d$. When applied to multivariate streams, this model will not take any correlations into consideration. Instead, it will simply learn the behavior of all channels at once, as if they were part of the same univariate stream.

An extension of autoregressive models to multivariate streams, taking correlations into account, exists with **vector-autoregressive** (VARs) models. A stream vector s_t with $N > 1$ is dependent upon the p previous data points,

$$s_t = \nu + \sum_{i=1}^{p} A_i s_{t-i} + \epsilon_t,$$

with coefficient matrices $A_i \in \mathbb{R}^{N \times N}$, intercept terms $\nu \in \mathbb{R}^N$ and noise terms $\epsilon_t \in \mathbb{R}^N$ [25]. The model parameters $\theta_{\text{model}} = \{\nu, A_1, .., A_p\}$ are estimated via least squares estimation. For this purpose, an excerpt of consecutive time series data is required and this restricts learning strategy Task 1 to the sliding window, as it contains the original time series. Once concept drift is detected, the model parameters are estimated for the most recent training set.

The second ML model examined is the **PCB-iForest** algorithm [11], which is an online version of the isolation forest

algorithm [26]. It rates the performance of individual isolation trees by their contribution to the total score produced by all trees and discards unnecessary trees once concept drift has been detected. It employs the extended isolation forest [27] algorithm to build an isolation forest model with the most recent sliding window of a stream of size w. A branch in the extended isolation forest algorithm is allowed to be diagonal and its branching criteria for a stream vector is

$$(s_t - p) \cdot n \leq 0,$$

with the slope $n \in \mathbb{R}^N$ and a random intercept point $p \in [min(x_t), max(x_t)]^N$. The branching process is repeated until a single stream vector is isolated or a maximum depth limit is reached. In order to judge the normalness of a new stream vector, it traverses down all trees and the average depth across the whole forest is the input to an anomaly scoring function, which is compared to a fixed threshold. The performance of individual trees can be assessed similarly by giving the anomaly scoring function only the depth of the respective tree i. If it contributed positively to the overall result, the tree's performance counter pc_i will increase by one and it will decrease correspondingly for a negative contribution. In order to detect concept drift Heigl et al [11] make use of the KSWIN method introduced in Section IV-B. Once a drift has been detected, only trees with $pc_i > 0$ will be retained and their performance counters will be reset. The model parameters of a PCB-iForest model can be summarized as

$$\theta_{\text{model}} = \{(\{(p_{i,j}, n_{i,j})\}, pc_i) | \forall i, j\},$$

with i indicating the number of trees and j being the number of branches within tree i. The second variant for building isolation forest models based on feature scoring evaluated by Heigl et al [11] is not covered in this paper.

A **two-layer autoencoder** (AE) reconstructing a feature vector x_t is going to be used as the third model. It serves the purpose of establishing a baseline for reconstruction-based approaches. The input is transformed by two fully connected layers to make a prediction $\hat{x}_t$ as

$$\hat{x}_t = r^{-1}(\sigma(r(x_t) * W_1 + b_1) * W_2 + b_2),$$

where $r(\cdot)$ refers to a reshaping operation, which reshapes its input to a one-dimensional vector, i.e., $r(x_t) \in \mathbb{R}^{Nw}$. The model parameters are $\theta_{\text{model}} = \{W_1, W_2, b_1, b_2\}$.

This simple reconstruction model is compared to the more complex **adversarial autoencoder** (USAD) from [28]. With $\text{FC}_i(x) = \sigma(x * W_i + b_i)$ as a fully-connected layer, one encoder

$$E = \text{FC}_{e,3} \circ \text{FC}_{e,2} \circ \text{FC}_{e,1},$$

is paired with two decoders

$$D_1 = \text{FC}_{d,1} \circ \text{FC}_{d,2} \circ \text{FC}_{d,3},$$
$$D_2 = \text{FC}_{d,4} \circ \text{FC}_{d,5} \circ \text{FC}_{d,6},$$

where $z_t = E(x_t^T)$ is the latent vector with $z_t \in \mathbb{R}^Z$, $Z << w$. Therefore, the model parameters are $\theta_{\text{model}} = \{W_{e,i}, b_{e,i}, W_{d,j}, b_{d,j} | \forall i, j\}$. Adversarial training is performed with the loss functions

$$\mathcal{R}_i = ||x_t - AE_i(x_t)||_2, \quad \mathcal{R}_{\text{both}} = ||x_t - AE_2(AE_1(x_t))||_2$$
$$\mathcal{L}_{AE1} = \frac{1}{n}\mathcal{R}_1 + \left(\frac{n-1}{n}\right)\mathcal{R}_{\text{both}}$$
$$\mathcal{L}_{AE2} = \frac{1}{n}\mathcal{R}_2 - \left(\frac{n-1}{n}\right)\mathcal{R}_{\text{both}}$$

where $AE_i = D_i \circ E$ and n refers to the number of training epochs. With progressively more training epochs, the impact of the pure reconstruction error diminishes in favor of the adversarial loss terms.

The fifth and final ML model is the neural basis expansion (**N-BEATS**) model by Oreshkin et al [29]. It relies on a stacked block design with double residual connections for better loss propagation. Each block produces both a forecast and a backcast as

$$h_l = \text{FC}_l(x_l),$$
$$\theta_l^b = \text{LINEAR}_l^b(h_l), \quad \theta_l^f = \text{LINEAR}_l^f(h_l),$$
$$\hat{x}_l = \sum_{i=1}^{\dim(\theta_l^b)} \theta_{l,i}^b v_i^b, \quad \hat{y}_l = \sum_{i=1}^{\dim(\theta_l^f)} \theta_{l,i}^f v_i^f,$$

here exemplified for the l-th block. The projection to a set of basis vectors v_i helps with interpretability since the weights can show the contribution of well known elements in time series analysis, such as seasonality and trend, when the appropriate basis functions are chosen. The model parameters can be summarized as

$$\theta_{\text{model}} = \{W_{l,i}, b_{l,i}, \theta_{l,j}^b, \theta_{l,k}^f, v_{l,j}^b, v_{l,k}^f | \forall l, i, j, k\}$$

and in this paper's streaming scenario, the model will forecast s_t based on the previous stream vectors $s_{t-w+1}, ..., s_{t-1}$, which are contained in the data representation x_t.

D. Nonconformity measure

The goal of a nonconformity score is to measure the "strangeness" of a feature vector. As an additional requirement, a nonconformity score should map this measurement to the interval $[0, 1]$ with 0 representing a feature vector judged to be normal and 1 giving an indication to an anomaly. Let $\hat{x}_t$ be a machine learning model's prediction for feature vector x_t. The first nonconformity score used for this work is going to be based on the **cosine similarity**,

$$a_t = 1 - \frac{x_t * \hat{x}_t}{||x_t||_2||\hat{x}_t||_2}, \quad \text{or} \quad a_t = 1 - \frac{s_t * \hat{s}_t}{||s_t||_2||\hat{s}_t||_2}$$

for forecasting models where the stream vectors at time step t are compared. Note that this only works for forecasting models in the multivariate case ($N > 1$).
Secondly, the **isolation forest** algorithm comes with its own nonconformity score

$$a_t = 2^{-E(h(x_t))/c(n)},$$

where $E(h(x))$ refers to the average tree depth across all isolation trees and $c(n)$ is the average depth to be expected given n samples were used to build the isolation forest.

TABLE I: Overview of all combinations to be evaluated

Learning strategy		**ML model**	**Nonconformity score**	**Anomaly score**
Task 1	*Task 2*			
SW URES ARES	μ/σ, KS	Online ARIMA	Cosine similarity	Average, Anomaly Likelihood
SW ARES	KS	PCB-iForest	iForest score	Anomaly Likelihood
SW URES ARES	μ/σ, KS	2-layer AE	Cosine similarity	Average, Anomaly Likelihood
SW URES ARES	μ/σ, KS	USAD	Cosine similarity	Average, Anomaly Likelihood
SW URES ARES	μ/σ, KS	N-BEATS	Cosine similarity	Average, Anomaly Likelihood

The first column of the learning strategy refers to methods for compiling and maintaining a training set of instances, where the options are sliding window (SW), uniform reservoir (URES) and anomaly-aware reservoir (ARES). The second column of the learning strategy consists of methods being applied to a training set in order to detect concept drift. These include observing the change in mean and standard deviation (μ/σ) or applying the two-sample Kolmogorov-Smirnov test (KS). Once concept drift has been detected, the ML model will be trained on the training set for one epoch. Note that in this context "cosine similarity" refers to the nonconformity score of $a_t = 1 - \text{cosine similarity}$.

E. Anomaly score

The last step remaining is to produce the final anomaly score f_t under consideration of the past k anomaly scores. The simplest approach to this end is to take the **average**

$$f_t = \frac{1}{k} \sum_{j=0}^{k-1} a_{t-j}.$$

As a second anomaly score, this paper will use the **anomaly likelihood** introduced by Lavin and Ahmad [9]. It compares the previous average, now denoted as μ_t, to a short term average $\tilde{\mu}_t$,

$$\mu_t = \frac{1}{k} \sum_{j=0}^{k-1} a_{t-j}, \quad \tilde{\mu}_t = \frac{1}{k'} \sum_{j=0}^{k'-1} a_{t-j},$$

where $k' << k$. With σ_t as the standard deviation of window k, the anomaly likelihood is defined as

$$f_t = 1 - Q\left(\frac{\tilde{\mu}_t - \mu_t}{\sigma_t}\right),$$

with $Q(x)$ as the Gaussian tail distribution function.

The original anomaly score presented by Calikus et al [1] will not be applied in this paper as it relies on the one-sample Kolmogorov-Smirnov test, which requires the feature vectors $x_1, ..., x_t$ to satisfy the i.i.d. condition. As this is not the case for our data representations, consisting of the past w stream vectors, $x_t = [s_{t-w+1}, ..., s_t]^T$, the score will not be applied in this work.

V. Results

In total, the previous section introduced one data representation, three approaches for Task 1, three approaches for Task 2 of the learning strategy, five different ML models, two nonconformity scores and two anomaly scores. An overview of all combinations to be evaluated can be found in table I.

A. Evaluation metrics

With only one data representation and one nonconformity score for each approach, the subjects of the evaluation are the ML models and corresponding learning strategies. Furthermore, the effectiveness of anomaly scores can be judged by comparison to the raw nonconformity scores.

As the process of fine-tuning a ML model at distinct time steps goes beyond the analysis proposed so far, judging the effect that fine-tuning has is chosen as a secondary goal. Under the assumption that concept drift has occurred, a fine-tuning session should have the immediate impact of reducing the average nonconformity score, i.e., the difference between the ML model's prediction and the actual stream vector will be lower. By itself, this fact does not matter to the task of anomaly detection. The most important aspect to this end is the model's capability to distinguish between normal and anomalous behavior. Although, it seems plausible that a fine-tuning session would also lower the variance of the nonconformity scores, which would help in distinguishing anomalous nonconformity scores. An experiment to measure the impact of fine-tuning will be performed as such: Once concept drift has been detected by one of the methods for learning strategy Task 2, an artificial anomaly will be introduced shortly after the fine-tuning session. Both the previous model and the retrained version will produce the predictions and their differences to the actual stream vector, as well as their nonconformity scores, will be calculated.

As the task of anomaly detection in time series usually entails the majority of data points being nominal, considering true negatives (TN) in an evaluation metric can lead to an overall lower sensitivity to actual anomalies being detected, while giving a false sense of the algorithm's capability. For this reason, we opt to use the **precision-recall area under the curve** (PR AUC) instead of the receiver operating char-

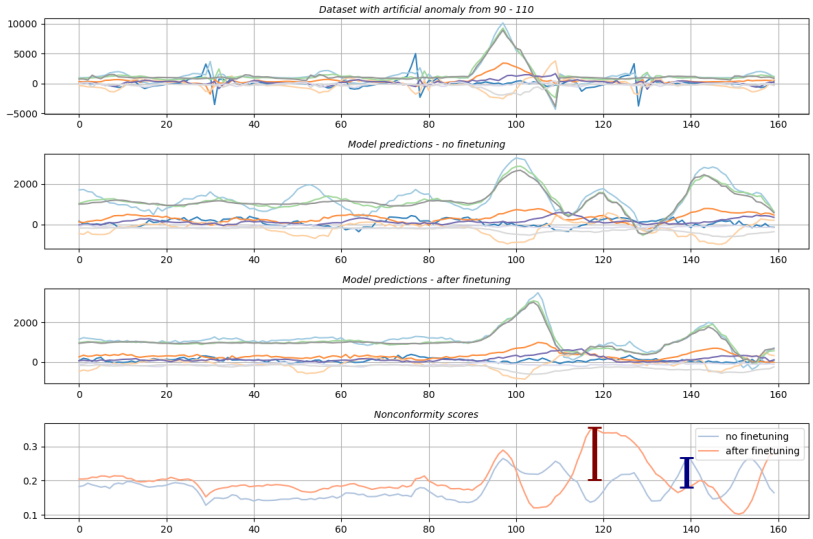

Fig. 1: Artificial anomaly inserted from $90 - 110$ after concept drift has been detected. The lowest plot contains two error bars indicating the difference between the average nonconformity score previous to the anomaly to the maximum score observed for the anomaly. Note that the maximum score could be observed as long as time step 210, since the data representation length is 100, i.e. the model gets the past 100 stream vectors as input. The red error bar of the model after finetuning is clearly larger than the error bar of the same model with no finetuning.

acteristic. True positives (TP), false positives (FP) and false negatives (FN) are defined to be sequences of time steps, following the definitions of Hundman et al [30]. Thereby, any positive prediction within an anomaly sequence is enough to be counted as a TP, whereas an anomaly sequence with no positive prediction is counted as a TN. Any predicted sequence with no overlap to an actual anomaly sequence is counted as a FP. The second evaluation metric is the **Numenta anomaly benchmark scoring function** (NAB) [5], which considers the surroundings of an anomaly sequence to identify TPs, FPs, and FNs and rewards early detection of the anomaly via a sigmoidal function weighting detections in and around an anomaly sequence. The third and final metric is the **volume under the surface** (VUS) [6]. It combines point-wise scores with the information of overlapping predicted and true anomaly sequences. Both the anomaly score threshold as well as a buffer region for true anomaly sequences are varied and the volume under the resulting surface is calculated in order to gain a parameter-free evaluation metric.

B. Experimental results

Table III lists the results of every method for the three benchmark corpora. For all experiments, the initial training set was chosen to be constructed from the first 5000 time steps and the data representation length was chosen as 100. The first thing that is apparent is the difference between some very negative NAB scores to at the same time high precision and recall values. This is due to the fact that NAB scores are calculated point-wise, whereas precision and recall is calculated based on overlapping intervals. Therefore, a long interval of consecutive falsely predicted anomalies is counted as only one false positive for the precision and recall measure, while every time step in that interval contributes $-\frac{1}{|\text{anomalies}|}$ to the NAB score.

TABLE II: Mathematical operations for Task 2 methods

	μ/σ**-Change**	**KSWIN**
Additions	$6Nw$	$2Nmw$
Multiplications	$2Nw$	$2Nmw$
Comparisons	$3Nw$	$(1+4m)Nw\log_2(mw)+N$

Mathematical operations required at time step t for training set length m, data representation length w and channel size N. The KSWIN method requires significantly more operations as the empirical CDF for one channel consists of mw elements, compared to a size of Nw for the mean feature vector in the μ/σ-Change method. The increased comparisons are due to determining insertion points via binary search for each element in both the current and historical training sets, when compared to their concatenated array.

In many cases, a performance increase can be observed for the anomaly-aware reservoir. Furthermore, the two different methods for concept drift detection are almost identical. This motivates the use of the less computationally complex method, which is the μ/σ-Change strategy as displayed in table II. Lastly, the online ARIMA model shows an overall lower performance than the more complex non-linear models.

Comparing the results for different anomaly scores averaged over all algorithms, it can be observed that a better AUC and NAB score is achieved when considering an average of non-conformity scores compared to the raw nonconformity scores, which is again improved on by the anomaly likelihood. The decreasing VUS score on the other hand is an indication for more point-wise predictions being made within the buffered

TABLE III: Experimental results

			Daphnet					Exathlon					SMD				
			Prec	*Rec*	*AUC*	*VUS*	*NAB*	*Prec*	*Rec*	*AUC*	*VUS*	*NAB*	*Prec*	*Rec*	*AUC*	*VUS*	*NAB*
Online ARIMA	*SW*	μ/σ	0.79	0.53	0.35	0.29	0.05	**1.00**	0.31	0.30	0.14	0.08	**1.00**	0.20	0.25	0.13	0.00
		KS	0.74	0.58	0.35	0.28	0.06	**1.00**	0.31	0.30	0.14	0.08	**1.00**	0.20	0.25	0.13	0.00
	URES	μ/σ	0.80	0.53	0.36	0.27	0.04	**1.00**	0.31	0.30	0.13	0.08	**1.00**	0.21	0.24	0.13	0.00
		KS	0.76	0.57	0.37	0.27	0.05	**1.00**	0.31	0.29	0.13	0.08	**1.00**	0.21	0.25	0.13	0.00
	ARES	μ/σ	0.76	0.50	0.35	0.27	0.05	**1.00**	0.30	0.46	0.14	0.07	**1.00**	0.23	0.39	0.13	0.00
		KS	0.79	0.45	0.37	0.27	0.05	**1.00**	0.32	0.44	0.14	0.07	**1.00**	0.23	0.39	0.14	0.00
2-layer AE	*SW*	μ/σ	0.71	0.68	0.43	0.36	0.10	0.91	0.69	0.52	0.21	−106.10	0.91	0.39	0.34	0.22	0.04
		KS	0.73	0.68	0.42	0.36	0.09	0.92	0.70	0.53	0.22	−118.12	0.91	0.39	0.34	0.22	0.04
	URES	μ/σ	0.73	0.67	0.42	0.36	0.10	0.90	0.77	0.55	0.26	−168.93	0.98	0.30	0.34	0.20	0.03
		KS	0.74	0.66	0.42	0.36	0.10	0.93	0.71	0.54	0.25	−181.05	0.94	0.32	0.33	0.20	0.01
	ARES	μ/σ	0.71	**0.69**	**0.51**	0.31	0.10	0.98	0.59	0.60	0.27	0.35	0.91	0.43	0.46	0.27	0.02
		KS	0.69	**0.69**	0.49	0.32	0.10	0.98	0.59	0.58	0.25	0.35	0.95	0.36	0.42	0.26	0.01
USAD	*SW*	μ/σ	0.73	0.66	0.42	0.39	**0.14**	0.72	0.31	0.33	0.26	0.17	**1.00**	0.25	0.35	0.35	0.06
		KS	0.71	0.68	0.42	**0.41**	0.13	0.73	0.30	0.33	0.26	0.16	**1.00**	0.26	0.35	0.34	0.07
	URES	μ/σ	0.73	0.66	0.42	0.40	**0.14**	0.71	0.42	0.43	0.26	0.27	0.98	0.29	0.36	0.32	0.10
		KS	0.74	0.65	0.42	0.40	0.13	0.71	0.42	0.44	0.26	0.25	0.97	0.31	0.37	0.32	0.10
	ARES	μ/σ	0.77	0.61	0.48	0.32	0.11	0.73	0.30	0.42	0.15	0.12	0.98	0.37	**0.48**	**0.37**	0.12
		KS	0.75	0.62	0.46	0.32	0.11	0.73	0.31	0.43	0.15	0.12	0.98	0.38	**0.48**	0.36	0.13
N-BEATS	*SW*	μ/σ	0.76	0.62	0.39	0.27	−7.21	0.97	**0.96**	0.58	0.34	−547.54	0.94	0.36	0.34	0.23	0.12
		KS	0.76	0.62	0.38	0.28	−8.32	0.98	0.95	0.58	0.34	−547.54	0.93	0.39	0.34	0.23	0.11
	URES	μ/σ	0.76	0.63	0.39	0.27	−0.62	0.98	0.94	0.57	0.34	−547.54	0.95	0.34	0.34	0.23	0.11
		KS	0.77	0.63	0.40	0.27	−0.42	0.98	0.95	0.58	0.34	−547.54	0.92	0.36	0.34	0.22	0.11
	ARES	μ/σ	**0.83**	0.39	0.39	0.27	0.08	0.96	0.95	**0.68**	**0.39**	**0.80**	0.87	**0.44**	0.41	0.19	0.18
		KS	0.81	0.54	0.40	0.26	0.09	0.96	0.93	**0.68**	**0.39**	**0.80**	0.91	0.43	0.40	0.20	**0.19**
PCB-iForest	*SW*	*KS*	0.82	0.44	0.39	0.22	0.03	0.93	0.61	0.49	0.22	0.26	0.94	0.32	0.35	0.22	0.07
	ARES	*KS*	0.79	0.47	0.38	0.22	0.03	0.94	0.61	0.49	0.21	0.26	0.91	0.35	0.34	0.22	0.07
Anomaly scores	*Raw*		0.73	0.57	0.27	0.36	−88.80	0.93	0.64	0.35	0.28	−346.40	0.98	0.24	0.20	0.24	−0.16
	Avg		0.73	0.68	0.37	0.37	−1.62	0.90	0.63	0.37	0.25	−345.72	0.98	0.24	0.21	0.22	0.03
	AL		0.78	0.57	0.41	0.29	0.09	0.91	0.54	0.53	0.23	0.33	0.95	0.36	0.43	0.24	0.08

The results displayed for the 26 algorithms reflect the results averaged across both anomaly scores (average / anomaly likelihood). Precision, recall, and their area-under-the-curve (AUC) are calculated based on overlapping intervals, whereas the NAB score is calculated point-wise. This is the reason for the disparity between very negative NAB scores and high precision / recall scores. The algorithm essentially predicts a long consecutive interval as anomalous. For the range-based scores it is counted as one false positive, while the NAB score counts every time step as $-\frac{1}{|\text{anomalies}|}$. For this purpose, the VUS metric combines both point-wise scores with overlapping intervals. In the last 3 rows, the impact of different anomaly scores is investigated by averaging over all algorithms that use them. The performance steadily increases from the raw nonconformity scores to an average of nonconformity scores to the anomaly likelihood, when considering the NAB score.

true anomaly sequences. This could be interpreted as the complex anomaly scores leading to a more focused prediction, while covering less points of the true anomaly sequences.

The role of fine-tuning after a detection of concept drift is analyzed qualitatively for one time series and one algorithm. After concept drift has occurred, two models are maintained: One model which is finetuned on the newest training set and the previous model, which is not finetuned. An artificial anomaly is inserted into the data stream shortly after in order to measure the difference in nonconformity score. The results are visualized in figure 1, where the error bars in the lower plot indicate the difference in nonconformity score of the peak of the anomaly to the previous average nonconformity score. This difference is clearly higher for the finetuned model than for the previous model, suggesting a better adaption to the current stream statistics. This experiment was performed on the dataset "S03R01E0" of the Daphnet collection. The algorithm consisted of a USAD model with a data representation length of 100, a sliding window and the μ/σ-Change strategy to detect concept drift.

VI. Conclusion

This work extended a framework for streaming anomaly detection to incorporate machine learning approaches and evaluated many different methods from literature in novel combinations. It was found that the two implemented concept drift detection strategies yielded almost identical results for the case of a training set relevant for training a machine learning model. This suggests the use of the computationally less complex μ/σ-Change strategy. Furthermore, it is apparent from the experimental results that range-based metrics and point-wise metrics can be very different for the same underlying anomaly scores. Lastly, many concepts from the literature could be confirmed to be beneficial, including the anomaly-aware reservoir and the anomaly likelihood.

Future research directions could include efficient hardware-accelerated implementations of individual components of a streaming anomaly detection algorithm or adapting further offline anomaly detection algorithms to the streaming scenario. Overall, streaming anomaly detection remains a challenging problem that is becoming increasingly important with the rise of autonomous systems and edge computing devices, and it is likely to stay relevant in the future.

REFERENCES

[1] E. Calikus, S. Nowaczyk, A. Sant'Anna, and O. Dikmen, "No free lunch but a cheaper supper: A general framework for streaming anomaly detection," *Expert Systems with Applications*, vol. 155, p. 113453, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420302773

[2] M. Bachlin, M. Plotnik, D. Roggen, I. Maidan, J. M. Hausdorff, N. Giladi, and G. Troster, "Wearable assistant for parkinson's disease patients with the freezing of gait symptom," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 2, pp. 436–446, 2010.

[3] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, "Exathlon: a benchmark for explainable anomaly detection over time series," *Proc. VLDB Endow.*, vol. 14, no. 11, p. 2613–2626, jul 2021. [Online]. Available: https://doi.org/10.14778/3476249.3476307

[4] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2828–2837. [Online]. Available: https://doi.org/10.1145/3292500.3330672

[5] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms – the numenta anomaly benchmark," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. Miami, FL, USA: IEEE, Dec. 2015, p. 38–44. [Online]. Available: http://ieeexplore.ieee.org/document/7424283/

[6] J. Paparrizos, P. Boniol, T. Palpanas, R. S. Tsay, A. Elmore, and M. J. Franklin, "Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection," *Proc. VLDB Endow.*, vol. 15, no. 11, p. 2774–2787, jul 2022. [Online]. Available: https://doi.org/10.14778/3551793.3551830

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, jul 2009.

[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. Pisa, Italy: IEEE, Dec. 2008, p. 413–422. [Online]. Available: http://ieeexplore.ieee.org/document/4781136/

[9] S. Ahmad and S. Purdy, "Real-time anomaly detection for streaming analytics," no. arXiv:1607.02480, Jul. 2016, arXiv:1607.02480 [cs]. [Online]. Available: http://arxiv.org/abs/1607.02480

[10] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, and S. Frankenstein, "Time series anomaly discovery with grammar-based compression," in *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015*. OpenProceedings.org, 2015, pp. 481–492. [Online]. Available: https://doi.org/10.5441/002/edbt.2015.42

[11] M. Heigl, K. A. Anand, A. Urmann, D. Fiala, M. Schramm, and R. Hable, "On the improvement of the isolation forest algorithm for outlier detection with streaming data," *Electronics*, vol. 10, no. 13, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/13/1534

[12] C. Liu, S. C. Hoi, P. Zhao, and J. Sun, "Online arima algorithms for time series prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[13] M. Munir, M. A. Chattha, A. Dengel, and S. Ahmed, "A comparative analysis of traditional and deep learning-based anomaly detection methods for streaming data," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. Boca Raton, FL, USA: IEEE, Dec. 2019, p. 561–566. [Online]. Available: https://ieeexplore.ieee.org/document/8999106/

[14] P. Mulinka and P. Casas, "Stream-based machine learning for network security and anomaly detection," in *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, ser. Big-DAMA '18. New York, NY, USA: Association for Computing Machinery, Aug. 2018, p. 1–7. [Online]. Available: https://dl.acm.org/doi/10.1145/3229607.3229612

[15] M. Munir, S. A. Siddiqui, M. A. Chattha, A. Dengel, and S. Ahmed, "Fusead: Unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models," *Sensors*, vol. 19, no. 11, p. 2451, May 2019.

[16] M. Y. Iqbal Basheer, A. Mohd Ali, N. H. Abdul Hamid, M. A. Mohd Ariffin, R. Osman, S. Nordin, and X. Gu, "Autonomous anomaly detection for streaming data," *Knowledge-Based Systems*, vol. 284, p. 111235, Jan. 2024.

[17] P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin, "Sand: streaming subsequence anomaly detection," *Proceedings of the VLDB Endowment*, vol. 14, no. 10, p. 1717–1729, Jun. 2021.

[18] A. D. R. L. Ribeiro, R. Y. C. Santos, and A. C. A. Nascimento, "Anomaly detection technique for intrusion detection in sdn environment using continuous data stream machine learning algorithms," in *2021 IEEE International Systems Conference (SysCon)*. Vancouver, BC, Canada: IEEE, Apr. 2021, p. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/9447092/

[19] S. Harush, Y. Meidan, and A. Shabtai, "Deepstream: Autoencoder-based stream temporal clustering and anomaly detection," *Computers Security*, vol. 106, p. 102276, Jul. 2021.

[20] K. Wu, K. Zhang, W. Fan, A. Edwards, and P. S. Yu, "Rs-forest: A rapid density estimator for streaming anomaly detection," in *2014 IEEE International Conference on Data Mining*. Shenzhen, China: IEEE, Dec. 2014, p. 600–609. [Online]. Available: http://ieeexplore.ieee.org/document/7023377/

[21] J. Ko and M. Comuzzi, "Keeping our rivers clean: Information-theoretic online anomaly detection for streaming business process events," *Information Systems*, vol. 104, p. 101894, Feb. 2022.

[22] N. Belacel, R. Richard, and Z. M. Xu, "An lstm encoder-decoder approach for unsupervised online anomaly detection in machine learning packages for streaming data," in *2022 IEEE International Conference on Big Data (Big Data)*. Osaka, Japan: IEEE, Dec. 2022, p. 3348–3357. [Online]. Available: https://ieeexplore.ieee.org/document/10020872/

[23] Z. Wang, Y. Zhou, and G. Li, "Anomaly detection by using streaming k-means and batch k-means," in *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. Xiamen, China: IEEE, May 2020, p. 11–17. [Online]. Available: https://ieeexplore.ieee.org/document/9101212/

[24] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, vol. 416, p. 340–351, Nov. 2020. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2019.11.111

[25] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Berlin, Heidelberg: Springer, 2005. [Online]. Available: http://link.springer.com/10.1007/978-3-540-27752-1

[26] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[27] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, p. 1479–1489, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2019.2947676

[28] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3395–3404. [Online]. Available: https://doi.org/10.1145/3394486.3403392

[29] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," Mar. 2022. [Online]. Available: https://openreview.net/forum?id=r1ecqn4YwB

[30] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," *CoRR*, vol. abs/1802.04431, 2018. [Online]. Available: http://arxiv.org/abs/1802.04431