Université Paris Cité

Choose Wisely: An Extensive Evaluation of Model Selection for Anomaly Detection in Time Series

Emmanouil Sylligardos, Paul Boniol, John Paparrizos, Panos Trahanias, Themis Palpanas

LIPADE-TR-Nº 11

March 18, 2023

Technical Report

LIPADE
Laboratoire d'Informatique PAris DEscartes

Choose Wisely: An Extensive Evaluation of Model Selection for Anomaly Detection in Time Series

Emmanouil Sylligardos*
ICS-FORTH
sylligardo@ics.forth.gr

Paul Boniol*
Université Paris Cité
boniol.paul@gmail.com

John Paparrizos
Ohio State University
paparrizos.1@osu.edu

Panos Trahanias
ICS-FORTH
trahania@ics.forth.gr

Themis Palpanas
Université Paris Cité; IUF
themis@mi.parisdescartes.fr

ABSTRACT

Anomaly detection is a fundamental task for time-series analytics with important implications for the downstream performance of many applications. Despite increasing academic interest and the large number of methods proposed in the literature, recent benchmark and evaluation studies demonstrated that no overall best anomaly detection methods exist when applied to very heterogeneous time series datasets. Therefore, the only scalable and viable solution to solve anomaly detection over very different time series collected from diverse domains is to propose a model selection method that will select, based on time series characteristics, the best anomaly detection method to run. Existing AutoML solutions are, unfortunately, not directly applicable to time series anomaly detection, and no evaluation of time series-based approaches for model selection exists. Towards that direction, this paper studies the performance of time series classification methods used as model selection for anomaly detection. Overall, we compare 17 different classifiers over 1800 time series, and we propose the first extensive experimental evaluation of time series classification as model selection for anomaly detection. Our results demonstrate that model selection methods outperform every single anomaly detection method while being in the same order of magnitude regarding execution time. This evaluation is the first step to demonstrate the accuracy and efficiency of time series classification algorithms for anomaly detection and represents a strong baseline that can then be used to guide the model selection step in general AutoML pipelines.

PVLDB Reference Format:
Emmanouil Sylligardos*, Paul Boniol*, John Paparrizos, Panos Trahanias, and Themis Palpanas. Choose Wisely: An Extensive Evaluation of Model Selection for Anomaly Detection in Time Series. PVLDB, XX(XX): XXXX - XXXX, 2023.
doi:XX.XXXXX/XXXXXXX.XXXXXXX
PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/boniolp/MSAD.

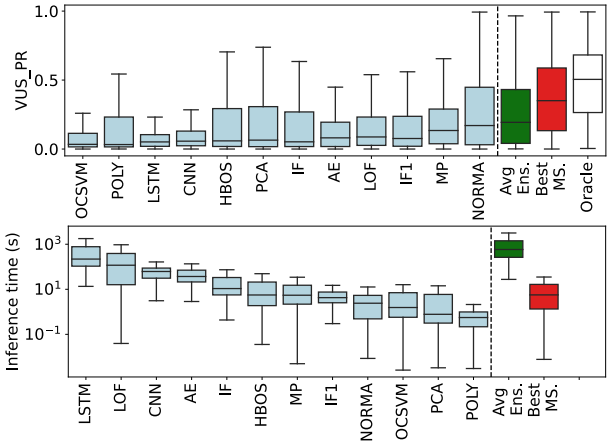

Figure 1: Summary of our evaluation on the TSB-UAD benchmark [61] of model selection methods (best in red) when compared to 12 anomaly detection methods and the averaging ensemble.

1 INTRODUCTION

Extensive collections of time-dependent measurements have become a reality in every scientific domain [9, 59]. The recording of these measurements results in an ordered sequence of real-valued data points commonly referred to as *time series*. Analysing time series data is becoming increasingly important in virtually every scientific and industrial domain, including astronomy [42], biology [10], energy sciences [7], environmental sciences [36], medicine [64], and social sciences [23]. Anomaly detection, in particular, has received ample academic and industrial attention [34, 58], finding applications across a wide range of domains and situations. These applications share the same goal [11, 72, 80]: analyzing time series to identify observations that do not correspond to expected behavior. In practice, anomalies can correspond to [2]: (i) noise or erroneous data (e.g., broken sensors); or (ii) actual data of interest (e.g., abnormal behavior of the observed system). In both cases, detecting such types is crucial for many applications [6, 40].

In recent years, many techniques have been proposed for time-series anomaly detection. Multiple surveys and experimental benchmarks have been written to summarize and analyze the state-of-the-art proposed methods [12, 44, 60, 61, 67]. Such surveys and benchmarks provide a holistic view of anomaly detection methods and how they perform. Unfortunately, these benchmark and evaluation studies demonstrated that no overall best anomaly detection methods exist when applied on very heterogeneous time series (i.e., coming from very different domains). In practice, we observe that some methods outperform others on specific time series with either specific characteristics (e.g., stationary or non-stationary time series) or anomalies (e.g., point-based or sequence-based anomalies).

*These authors contributed equally to this work.

Proceedings of the VLDB Endowment, Vol. XX, No. XX ISSN 2150-8097.
doi:XX.XXXXX/XXXXXXX.XXXXXXX

To overcome the above limitation, ensembling solutions have been proposed [4] that consist of running all existing anomaly detection methods and averaging all anomaly scores. Figure 1 shows that this solution (in green) is outperforming all individual existing techniques in the TSB-UAD benchmark [61]. Nevertheless, as shown in Figure 1, such solutions require running all methods resulting in an excessive execution time that is not feasible in practice.

Therefore, the only scalable and viable solution to solve anomaly detection over very different time series is to propose a model selection method that will select, based on time series characteristics, the best anomaly detection method to run. This topic has been tackled in several recent research works related to AutoML (Automatic Machine Learning) for the general case of anomaly detection [82, 83]. Nevertheless, existing AutoML solutions require (i) a universal objective function among models, which is not applicable to anomaly detection methods; (ii) a predefined set of features, which is difficult to obtain for time series due to varying lengths and the lack of standardized featurization solutions; (iii) running multiple anomaly detection methods several times, which is prohibitively expensive in practice; or (iv) labeled anomalies, which (in contrast to classification tasks) are difficult to obtain. Therefore, more work is needed in order to render AutoML solutions applicable to time-series anomaly detection.

Generally, the model selection problem for anomaly detection in time series can be formalized as a time series classification problem. In such a scenario, the objective is to train a time series classification model on time series for which we know in advance which anomaly detection method is the best. However, the lack of a benchmark with labeled time series has been a limiting factor for training robust model selection models (this only changed very recently [45, 61, 67]). Therefore, there exists no experimental evaluation that measures the efficiency of classification methods for the task of model selection for time series anomaly detection. Though, such an evaluation is very important for determining which time series classification methods are accurate as model selection methods, and which solutions should be considered in unsupervised settings (i.e., using model selection approaches on time series from domains that were not included in the training set). These results would significantly help the design and effectiveness of general AutoML pipelines for time series.

Thus, in this paper, we evaluate the performance of time series classification methods used as model selection for anomaly detection in time series. To do so, we propose a pipeline that enables any kind of time series classifier to be used for any univariate time series with different lengths. We then compare the execution time and accuracy for feature-based, traditional time series classifiers and deep learning classification algorithms. We also measure how these models perform when trained on time series of a given domain (e.g., electrocardiogram [55]) and tested on time series from a different domain (e.g., robotics sensors measurements [65]).

Overall, we compare 17 different classifiers over 1800 time series, and we propose the first extensive experimental evaluation of time series classification as model selection for anomaly detection. Our results demonstrate that model selection methods outperform every single anomaly detection method while being in the same order of magnitude regarding execution time. Figure 1 shows a summary of our experimental evaluation, where the best model selection method (in red in Figure 1) is up to 2.8× more accurate than the best anomaly detection method in the TSB-UAD benchmark and 1.9× more accurate than the ensembling solution mentioned above. This evaluation is the first step to demonstrate the accuracy and efficiency of time series classification algorithms for anomaly detection. It represents a strong baseline that can then be used to guide the choice of approaches for the model selection step in more general AutoML pipelines.

Our contributions can be summarized as follows.

- We describe and study the needs of evaluating time series classification methods for model selection (Section 3).
- We introduce our novel pipeline for model selection applied to anomaly detection in time series. As this pipeline is generic, we describe how it can be used with both feature-based classification methods, traditional time series classification methods, and deep learning-based methods (Section 4).
- We describe our experimental benchmark and provide details on both anomaly detection methods and time series classification methods considered in this paper (Section 5). We make all our material publicly available online [20] and provide an interactive WebApp [21] for exploring the experimental results.
- We present an extensive experimental evaluation, measuring the anomaly detection accuracy and execution time (both training and inference) of model selection algorithms (Section 5.1). We evaluate the influence of important parameters and the relationship between classification and anomaly detection accuracy (Sections 5.2, 5.3, and 5.4). Moreover, we measure the transferability of model selection algorithms to new types of time series by testing multiple combinations of train and test datasets that do not contain the same kinds of time series (Section 5.5).

Finally, we conclude with the implications of our work and discuss possible future directions that could help improve both the accuracy and the execution time of our proposed pipeline (Section 6).

2 BACKGROUND AND RELATED WORK

We first introduce notations useful for the rest of the paper (Section 2.1). Then, we review existing time-series anomaly detection methods (Section 2.2), and discuss their limitations when applied on large heterogeneous sets of time series (Section 2.3).

2.1 Time-Series and Anomaly Score Notations

Time Series: A time series $T \in \mathbb{R}^n$ is a sequence of real-valued numbers $T_i \in \mathbb{R}$ $[T_1, T_2, ..., T_n]$, where $n = |T|$ is the length of T, and T_i is the i^{th} point of T. We are typically interested in local regions of the time series, known as subsequences. A subsequence $T_{i,\ell} \in \mathbb{R}^\ell$ of a time series T is a continuous subset of the values of T of length ℓ starting at position i. Formally, $T_{i,\ell} = [T_i, T_{i+1}, ..., T_{i+\ell-1}]$. A dataset $\mathcal{D}$ as a set of time series. Note that the time series contained in $\mathcal{D}$ can be of diverse lengths. We define the size of $\mathcal{D}$ as $|\mathcal{D}|$.

Anomaly Score Sequence: For a time series $T \in \mathbb{R}^n$, an anomaly detection method (or detector) D returns an anomaly score sequence S_T. For point-based approaches (i.e., methods that return a score for each point of T), we have $S_T \in \mathbb{R}^n$. For subsequence-based approaches (i.e., methods that return a score for each subsequence of a given length ℓ), we have $S_T \in \mathbb{R}^{n-\ell}$. Overall, for subsequence-based approaches, we define $S_T = [S_{T1}, S_{T2}, ..., S_{Tn-\ell}]$ with $S_{Ti} \in [0, 1]$. In most of the applications, we require the anomaly score to have the same length as the time series. Therefore, for subsequence-based approaches, we define $S_T = [S_{T1}]_{i\in[0,\ell/2]} + [S_{T1}, S_{T2}, ..., S_{Tn-\ell}] + [S_{Tn-\ell}]_{i\in[0,\ell/2]}$ with $|S_T| = |T|$.

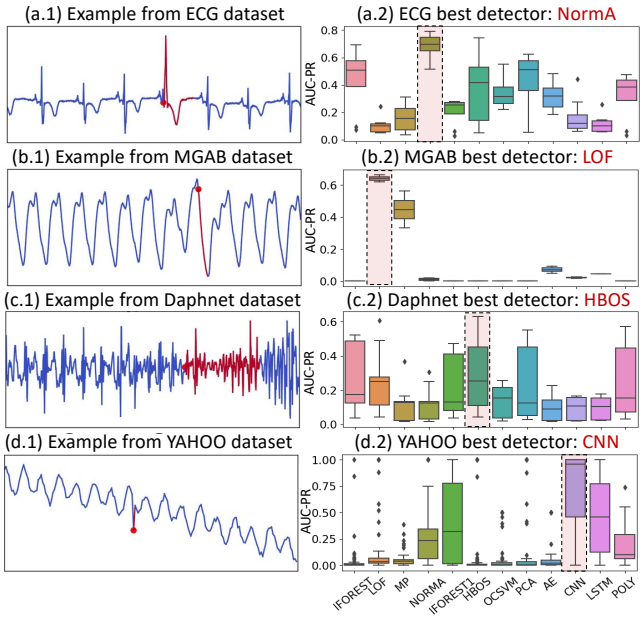

Figure 2: Accuracy of 12 anomaly detection methods on 4 datasets.

Anomaly detection Accuracy: For a time series $T \in \mathbb{R}^n$, an anomaly detection method (or detector) D that returns an anomaly score sequence S_T and labels $L \in [0,1]^n$ that indicated with 0 or 1 if the points in T are normal or abnormal respectively, we define $Acc : \mathbb{R}^n, \{0,1\}^n \rightarrow [0,1]$ as an accuracy function for which $Acc(D(T), L)$ indicates how D is accurate (i.e., detect the anomalies correctly) when applied on T and accordingly to L. The closer to one, the better the detector.

2.2 Anomaly Detection Methods for Time Series

Several different methods (for diverse types of time series, or applications) have been proposed in the literature.

One type of anomaly detection methods is *discord-based methods*. The latter focus on the analysis of subsequences for the purpose of detecting anomalies in time series, mainly by utilizing nearest neighbor distances among subsequences [24, 35, 46, 49, 51, 69, 80].

Instead of measuring nearest neighbor distances, *proximity-based methods* focus on estimating the density of particular types of subsequences in order to either extract a normal behavior or isolate anomalies. As a subsequence can be seen as a multi-dimensional point (with the number of dimensions corresponding to the subsequence length), general outlier detection methods can be applied for time series anomaly detection [22, 50, 53]. Among them, Isolation Forest [50] has been shown to work particularly well for time series anomaly detection task [17]. Moreover, recent proximity-based methods dedicated to identifying abnormal subsequences in time series have been proposed. Such method, as NormA that first cluster data to obtain the normal behavior [13–15, 18, 19], have been shown to achieve strong performances.

Furthermore, *forecasting-based methods*, such as recurrent neural network-based [54] or convolutional network-based [56], have been proposed for this task. Such methods use the past values as input, predict the following one, and use the forecasting error as an anomaly score. Such methods are usually trained on time series without anomalies, or make the assumption that the anomalies are significantly less frequent than the normal behaviors.

Finally, *reconstruction-based methods*, such as AutoEncoder-based approach [66], are trained to reconstruct the time series and use the reconstruction error as an anomaly score. As both forecasting and reconstruction-based categories detect anomalies using prediction errors (either forecasting or reconstruction error), we can group them into *prediction-based methods*.

2.3 Limitations of Anomaly Detection Methods

Recent benchmarks and experimental evaluation have been proposed in the literature [45, 60, 67]. Such benchmarks provide a large collection of time series from various domains and evaluate multiple methods belonging to the categories mentioned above. However, these experimental evaluations led to the same conclusion: no methods exist that outperforms all the other on all time series from various domains. Figure 2, which depicts the accuracy of 12 diverse anomaly detection methods[1] on four time series datasets, illustrates the conclusion above. In Figure 2(a.2), we observe that NormA is the most accurate model on the ECG dataset [55] (a time series example is depicted in Figure 2(a.1)). However, Local Outlier Factor (LOF) [22], and Matrix profile (MP) [80] are significantly outperforming NormA on the MGAB dataset [73] (see Figure 2(b.2)), whereas CNN [56] is outperforming NormA, LOF, and MP on the YAHOO dataset [47] (see Figure 2(d.2)). The following two reasons explain this large difference in performance among datasets.

2.3.1 **Heterogeneity in anomaly types**. First, There are three types of time-series anomalies: *point*, *contextual*, and *collective* anomalies. *Point* anomalies refer to data points that deviate remarkably from the rest of the data. Similarly, *Contextual* anomalies refer to data points within the expected range of the distribution (in contrast to point anomalies) but deviate from the expected data distribution, given a specific context (e.g., a window). For instance, Figure 2(d.1) illustrates a time series from the YAHOO dataset with a *Contextual* anomaly. The value of the anomalies is inside the range of normal values but is abnormal in the context of the distribution of values of the surrounding point. For this particular types of anomalies, *reconstruction* and *forcasting*-based methods are particularly accurate (as shown in Figure 2(d.2))

Collective anomalies refer to sequences of points that do not repeat a typical (previously observed) pattern. The first two categories, namely, point and contextual anomalies, are referred to as *point-based* anomalies, whereas *collective* anomalies are referred to as *subsequence* anomalies. For instance, Figure 2(a.1),(b.1), and (c.1) show three time series with sequence anomalies. However, even for time series belonging to the same anomaly type categories, we observe that the most accurate models are all different.

2.3.2 **Heterogeneity in time series structures**. This diversity in model accuracy can be explained by other factors related to the time series structures. Indeed, on top of these categories mentioned above, the combination of them also matters. First, we need to differentiate time series containing *single* anomalies from time series containing *multiple* anomalies. Last, the *multiple* time series category has to be divided into two subcategories, namely time series containing *multiple different* and *multiple similar* anomalies. For

[1]We use 12 methods that have been employed in previous studies [60, 61]. Note that other methods and variations exist that may lead to improved results.

instance methods based on neighbor distance computation such as LOF are very accurate in detecting *single* or *multiple different* anomalies, but less accurate for *multiple similar*. To illustrate this point, Figure 2(a.2) depicts the results of 12 anomaly detection methods on the ECG dataset (that contains a large number of *multiple similar* anomalies), for which LOF accuracy is low. On the contrary, Figure 2(b.2) depicts the results of the same 12 anomaly detection methods on the MGAB dataset (that few *multiple different* anomalies), for which LOF accuracy is high.

To all these differences between time series, resulting in large variability in the accuracy of anomaly detection methods, we can add differences in the time series statistical characteristics. The latter can be the differences between *stationary* (i.e., with a constant distribution of values over time) and *non-stationary* (i.e., with a changing distribution of values over time) time series, or *single normality* (i.e., time series containing only one normal behavior) and *multiple normalities* (i.e., time series containing multiple normal behaviors) time series.

3 MOTIVATION AND PROBLEM FORMULATION

In this section, we describe solutions that can be applied to solve the limitations mentioned above, and we motivate the benefits of these solutions. We finally formally define the problem.

3.1 Ensembling Detectors

The first solution is to ensemble all the anomaly scores produce by all the detectors. Multiple ensembling techniques have been proposed in the literature [4] from which two main methods arise: (i) *Averaging*: the average of the anomaly scores for each timestamp, (ii) *Maximizing*: the maximum anomaly score for each timestamp (iii) *Average of Maximum*: the average of the maximum for randomly selected sub-sample set of detectors. *Averaging* strategy is proven to be the more robust and low-risk strategy compared to the other two [4]. Formally, for a given time series T of length n and a set of detectors $\mathcal{B}$, *Averaging* strategy is defined as $Avg.Ens. = [Avg_0, Avg_1, ..., Avg_n]$ with Avg_i (for $i \in [0, n]$) equals to: $Avg_i = (1/|\mathcal{B}|) \sum_{D \in \mathcal{B}} D(T)_i$.

In the rest of the paper, we call the *Averaging* strategy *Averaging Ensemble (Avg Ens.)*. As depicted in Figure 1(a), which shows the accuracy of detectors (in light blue) and the Averaging ensemble, we observe that such a strategy already outperforms all existing approaches. Nonetheless, such a method requires running all detectors to produce one ensembled anomaly score, resulting in a costly execution time (see Figure 1(b)). In a scenario with very long time series and an increasing number of detectors to consider, such an approach is not sustainable and feasible in practice.

3.2 Model Selection

A solution to tackle the limitations mentioned above is to apply model selection based on the characteristics of the time series. The main idea is to train a model to automatically select the best detector (anomaly detection methods) for a given time series. In such a case, the user only has to run one model, drastically limiting the execution time required. This topic has been tackled in several recent papers related to AutoML (Automatic Machine Learning). Recent approaches, such as MetaOD [82, 83], explored meta-learning to identify the best outlier detection algorithm on tabular datasets. These research works rely on pre-computed accuracy performances of models on a subset of datasets to essentially learn a mapping from the characteristics of a dataset to detectors' accuracy performance. Methods have been proposed to select models in an unsupervised way [38], but require using multiple models trained, which (like the averaging ensembling approach) limit the applicability in terms of execution time.

3.3 Classification for Model Selection

In general, for the specific case of time series, most of the work described above and future AutoML methods will rely on time series classification methods for the model selection step. In such a case, the method aims to classify time series into classes corresponding to the available anomaly detection methods. One time series must be classified into the detector class that maximizes anomaly detection accuracy. However, no existing guidelines indicate which time series classification approach can be used as model selection. Therefore, there is a need to evaluate and measure what accuracy gains time series classification approach can bring to anomaly detection.

The first step is to evaluate the potential gain in accuracy that model selection could bring. To do this, recent time series Anomaly benchmarks [61, 67] can be used. We can evaluate the upper bound that model selection methods can reach on such benchmarks. Thus, we define a hypothetical model called *Oracle*, which, for a given time series, always selects the correct anomaly detector to use (i.e., the most accurate one). Formally, *Oracle* is defined as follows:

Definition 1. *Given a dataset $\mathcal{D}$ composed of time series $T \in \mathbb{R}^n$ (with the length of the time series n non-constant for all time series in $\mathcal{D}$) and a set of detectors $\mathcal{B} = \{D_0, D_i, ..., D_n\}$ with the number of detectors defined as $|\mathcal{B}| = m$, $Oracle(T) = \text{argmax}_{D \in \mathcal{B}} \left\{Acc\big(D(T), L\big)\right\}$*

In the rest of the paper, we call *Orcale*, the hypothetical model $Oracle(T)$ applied on all T in a given benchmark. For example, figure 1 shows in white the accuracy of *Orcale* applied on the TSB-UAD benchmark [61] and demonstrates that a perfect model selection method outperforms the best detector in TSB-UAD and the Averaging Ensemble by a factor of 2.5. This large improvement in accuracy and the fact that only one model has to be run confirms the interest in accuracy and execution time of model selection applied for time series anomaly detection. Thus, there is a need to evaluate the performances of existing time series classification methods when used as model selection algorithms and how close such methods can get to the *Orcale*.

3.4 Problem Formulation

Therefore, based on the limitations and the motivation listed above, we can formalize the problem of model selection as follows:

Problem 1. *Given a dataset $\mathcal{D}$ composed of time series $T \in \mathbb{R}^n$ (with the length of the time series n non-constant for all time series in $\mathcal{D}$) and a set of detectors $\mathcal{B} = \{D_0, D_i, ..., D_n\}$ with the number of detectors defined as $|\mathcal{B}| = m$, we want to build a model selection method $\mathcal{M}$ that takes a time series $T \in \mathcal{D}$ and returns a detector $D \in \mathcal{B}$ (formally $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{B}$) such that, for a given time series T and corresponding label L:*

$$\mathcal{M}(T) = Oracle(T) = \underset{D \in \mathcal{B}}{\text{argmax}} \left\{Acc\big(D(T), L\big)\right\}$$

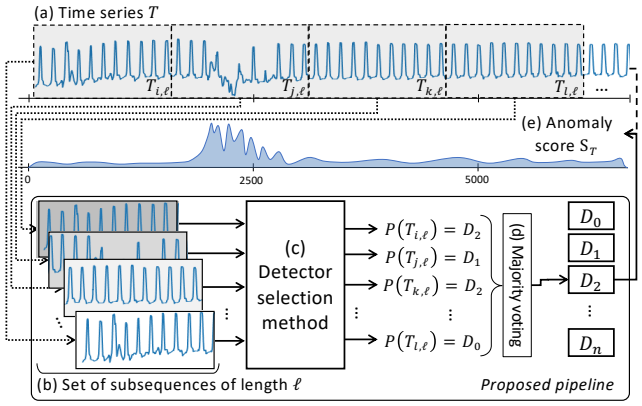

Figure 3: Proposed pipeline for the method selection

In practice, we do not have the label L. Therefore, the objective is to build a model $\mathcal{M}$ that estimate the equation above. Moreover, As the input of the model $\mathcal{M}$ is a time series (of variable length) and the output is a detector D among a finite state of possible detectors $\mathcal{B}$, the problem can be seen as a time series classification problem for which the classes are the detectors in $\mathcal{B}$. Therefore, the only requirement is to have computed in a benchmark all $Acc(D(T), L)$ for all $T \in \mathcal{D}$ and all $D \in \mathcal{B}$ and use it as a training set.

3.5 Objectives

In summary, the goal of this experimental evaluation is to answer the following questions:

- **Classification as Model selection**: How do existing time series classification methods compare to individual detectors and the *Oracle*?
- **Ensembling or selecting**: Is selecting detectors automatically more accurate than ensembling them?
- **Features or Raw values**: Should we use time series features or the raw time series values to predict which detector to use?
- **Out-Of-Distribution**: What happens when the model selection approach is trained on some datasets and tested on completely new datasets? Are all the answers from the previous questions holding?

We now describe our pipeline and experimental evaluation to answer the questions listed above.

4 PROPOSED PIPELINE

In the following section, we provide a comprehensive explanation of the proposed pipeline. This pipeline corresponds to a sequence of pre-processing and post-processing steps such that the input of the model selection algorithms are equal length. The proposed pipeline, illustrated in Figure 3, is composed of the following steps: (i) **Preprocessing step**: Extraction of the subsequences of same length (Figure 3(b)), (ii) **Prediction step**: Prediction of which detector to use for each subsequence (Figure 3(c)), and (iii) **Selection step**: Majority voting among all the different prediction to select one detector only (Figure 3(d)). In the following section, we describe in details the three steps mentioned above.

4.1 Preprocessing Step

Time series classification can be performed with three different strategies: (i) treating the entire time series as one sample, (ii) dividing the time series into overlapping subsequences, (iii) dividing the time series into shifting subsequences (i.e., non-overlapping subsequences). The first strategy is straightforward, as each time series is treated as a single observation. Nevertheless, not all classifiers can handle variable-length inputs, and training such models can be computationally intensive (i.e., batches of time series cannot be treated in parallel). The second strategy involves dividing the time series into overlapping subsequences (of a given window length ℓ). Despite posiible loss of information, it forces each input of the methods to be the same length (ℓ), allowing simpler and faster computation when performed in parallel. In the third strategy, we divide time series into non-overlapping subsequences (of a given length ℓ) removing redundant information in overlapping subsequences. The latter significantly reduces the number of inputs generated by the second strategy and significantly accelerates the training and inference time. For these reasons, as illustrated in Figure 3(a) and (b), we chose the third strategy.

Thus, the time series of length $|T|$ are divided into $\mathbb{T}_l$ non-overlapping subsequences of length ℓ. When the length of the time series is not divide evenly with the window length ℓ, the remainder is added as overlap between the first two windows. Formally, $\mathbb{T}_l$ is defined as follows:

$$\mathbb{T}_l = \begin{cases} \{T_{i*\ell,\ell} | \forall i \in [0, \lceil \frac{|T|}{\ell} \rceil]\} & , \text{if } \lceil \frac{|T|}{\ell} \rceil = \frac{|T|}{\ell} \\ \{T_{0,\ell}\} \bigcup \{T_{|T|-\lceil \frac{|T|}{\ell} \rceil + i*\ell,\ell} | \forall i \in [0, \lceil \frac{|T|}{\ell} \rceil]\} & , \text{if } \lceil \frac{|T|}{\ell} \rceil < \frac{|T|}{\ell} \end{cases}$$

We expect the length ℓ to impact the anomaly detection accuracy strongly. We thus test multiple length values and measure their influence (on accuracy and execution time) in Section 5.

At this point, we preprocessed the time series into subsequences of equal length. We now discuss the label (i.e., best detector to apply) attribution. For that matter, we use the TSB-UAD benchmark [61] that contains 12 different anomaly detection methods. We compute the 12 methods for each time series and attribute the most accurate (based on AUC-PR) detector as the label. Then, the produced subsequences share the same label as the time series they originate from. This labeled dataset can be used to train classification methods and divided into the train, test, and validation sets. It is important to note that although each time series produce multiple samples (i.e., subsequences), these samples should not be mixed between train, validation, and test set. Indeed too strong similarities between subsequences that belong to the same time series, if contained in both the train, validation and the test, can lead the classification model to overfit or create an illusion of accuracy. Therefore, we guarantee that the intersection between the train, validation, and test set, regarding which time series the corresponding subsequences originate from, is null.

4.2 Time Series Classification Approaches

In this section, we describe the time series classifier approaches that we use as model selection methods. As many approaches have been proposed in the literature, we restrict our experimental evaluation to two main categories: (i) feature-based and (ii) raw-based methods. In addition, the second category can be divided into two sub-categories: (i) convolutional-based and (ii) transformer-based.

Figure 4 illustrates a simplified taxonomy of the methods considered, and we describe them in the following section.

4.2.1 Feature-based classification. The main idea regarding feature-based classification is to use the dataset of time series (or subsequences of time series) to create a dataset whose samples are described by features common to all samples. Using the feature-based dataset, we then employ traditional machine learning classifiers to classify each time series. We use the TSFresh [26] (Time Series Feature extraction based on scalable hypothesis tests) to extract each subsequence's features. The latter is used for automated time series feature extraction and selection based on the FRESH algorithm [27]. More specifically, it automatically selects relevant features for a specific task. This is achieved using statistical tests, time series heuristics, and machine learning algorithms. The TSFresh package provides three options for automated feature extraction, that is (i) *comprehensive*, (ii) *efficient*, and (iii) *minimal*. The first two options provide 700 hundred features instead of the latter, which provides only 9. For scalability reasons (the dataset to transform can reach more than 6 million subsequences), we consider the *minimal* option in this paper. Once the feature-based dataset is created, we can consider the following classification approaches.

[SVC] A Support Vector Classifier (SVC) is a classifier that maps training instances in space in order to maximise the width of the gap between the classes. New instances are mapped into the same space and classified according to which side of the gap they fall.

[Bayes] The naive Bayes classifier uses Bayes' theorem to predict the class of a new instance based on prior probabilities and class-conditional probabilities. The prediction is made by computing the posterior probabilities for each class.

[MLP] A Multi Layer Perceptron (MLP) is a fully connected (connections between every neuron of two following layer) neural network.

[QDA] A Quadratic Discriminant Analysis (QDA) Classifier is a linear discriminant analysis algorithm. The prediction is made by computing the discriminant functions for each class.

[AdaBoost] AdaBoost is a boosting ensemble machine learning algorithm for solving classification problems. It creates a sequence of weak classifiers, where each classifier is trained on a weighted sample of the data. The prediction is made by combining the predictions of all classifiers, weighted by their accuracy respectively.

[Decision Tree] A Decision Tree Classifier is a tree-based method that represent a sequence of decisions based on the features of the data. To classify a new instance, the algorithm follows the decisions in the tree to reach a leaf node associated to a class.

[Random Forest] A Random Forest Classifier is an ensemble machine learning algorithm that combines multiple decision trees, where each tree is built using a random subset of the features and a random sample of the data. The final class prediction for a new instance results of the aggregation of the predictions of all trees.

[kNN] A KNN classifier is a method that classify instances based on its distance to other instances in a training set. The algorithm assigns the new instances to the class with the most number of closest neighbors among the K nearest data points.

4.2.2 Raw-based classification. Instead of using features to perform classification, the raw values of the time series can be used. Indeed, whereas features are efficient for homogenizing time series datasets (e.g., setting a constant number of features for variable length time series), it might hide important information in the shape of consecutive values. Thus, many approaches that use raw-values time series have been proposed.

[Rocket] Among the recent raw-values methods, miniRocket [29] is one of the state-of-the-art time series classification methods. The latter consists of a feature extraction step and a classification step. More specifically MiniRocket works by transforming input time series using a small, fixed set of convolutional kernels and using the transformed features to train a logistic regression classifier (using stochastic gradient descent). In the rest of the paper, we refer to miniRocket as Rocket.

4.2.3 Convolutional-based classification. Another specific type of approaches that takes as input raw-values of time series and have been shown to be accurate for time series classification [16] are convolutional-based approaches.

[ConvNet] A Convolutional Neural Network (CNN) [57] is a type of deep learning neural network widely used in image recognition that is specially designed to extract patterns through data with a grid-like structure, such as images, or time series. A CNN uses convolution, where a filter is applied on a sliding window over the time series. The ConvNet architecture proposed in [77] is composed of three stacked Convolutional blocks followed by Global Average Pooling (GAP), and a Softmax activation function. Each Convolutional block is composed of a convolutional layer (used with a kernel legnth of 3) followed by a batch normalization layer, followed by a ReLU activation function is applied.

[ResNet] The Residual Network (ResNet) architecture [41] was introduced to address the gradient vanishing problem encountered in large CNNs [70]. A ResNet is composed of several blocks connected together with residual connections (i.e., identity mapping). For time series classification, a ResNet architecture has been proposed in [77], and has demonstrated a strong classification accuracy [31]. It is the same architecture as the previously described ConvNet model, with adding residual connection between each Convolutional block.

[InceptionTime] The model consists of a network using residual connections and convolutional layers with kernel of variable lengths [32]. Such network uses 3 Inception block that replace the traditional residual blocks that we can find in a ResNet architecture. Each Inception block consist of a concatenation of convolutional layers using different size of filters. For each blcok, the time series is fed to three different 1D convolutional layers with different kernel sizes (10, 20, and 40) and one Max-Pooling layer with kernel size 3. The last step consists of concatenating the previous four layers along the channel dimension and applying a ReLu activation function to the output, followed by batch normalization. All the convolutional layers used in the module come with 32 filters and a stride parameter of 1.

4.2.4 Transformer-based classification. The second category, initially introduced for computer vision tasks [30, 74], is Transformer-based approaches. Such methods can easily be adapted for time series classification tasks, and we propose in this paper SIT (Signal Transformer), an extension of a recent computer vision transformer approach [30]. SIT first starts by projecting the input to the latent space with an embedding step. After the embedding step, the input is mapped to a D dimensional space (we use $D = 256$ in the rest of the paper) that serves as input to an encoder. For SIT, we use an encoder originally proposed for computer vision tasks [74] that consists of multiple blocks. Each block has an alternating multi-headed self-attention block and a feed-forward layer, both preceded

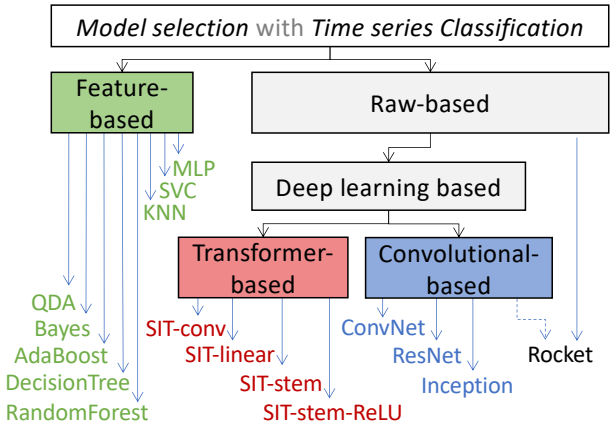

Figure 4: Taxonomy of time series classification approaches used as model selection methods. We use the same color-code for each class in all figures in the paper.

by a normalization step and a residual connection. We now describe the different embedding steps in detail in the following paragraphs. In the experimental evaluation, we consider the SIT architecture with the four embedding as four different methods.

[SIT-conv] This embedding uses a single convolutional layer to map the time series into the latent space. The convolutional layer has a kernel and stride of the same length (we use a length of 16 in the rest of the paper), essentially making non-overlapping steps over the time series. Finally, the convolutional layer has D filters to match the input dimension of SIT encoder.

[SIT-linear] The linear embedding [30] splits the input time-series into non-overlapping subsequences of length l_sit (we use $l_sit = 16$ in the rest of the paper). Then, each patch is linearly projected into D dimensions to match the input dimension SIT encoder.

[SIT-stem] The stem embedding [78] consists of 3 convolutional layers with a kernel length of 3, a stride length of 2, and a number of filters equal to 3, 5, and 7, respectively. These three convolutional layers are then followed by a last convolutional layer with D dimensions and a kernel and stride length equal to 1. This embedding was initially proposed to overcome unstable behavior while training because of their early visual processing step.

[SIT-stem-ReLU] Similarly to the previous embedding, the stem-ReLU embedding [76] consists of 4 convolutional layers with kernel lengths of 7, 3, 3, 8, stride lengths of 2, 1, 1, 8, and padding of 3, 1, 1, 0. Moreover, the number of filters for each convolutional layer is set to 3, except the last one that has D filters to match the dimension SIT encoder.

4.3 Selecting the Detector

We train the time series classification methods mentioned in the previous section to predict the best detector for each subsequence (as shown in Figure3(c)). However, there is no guarantee that the classification model selects for all subsequences the same model. Therefore, we choose the best detector for one time series by doing a majority voting step between the predictions for every subsequence, such that the most voted detector is selected as the detector of the time series. Formally, given a classification model $\mathcal{M}_{cl}$ applied on a given time series T subsequences $\mathbb{T}_\ell$, we define $\mathcal{M}_{cl}(\mathbb{T}_\ell)$ the set of model selected for each subsequence in $\mathbb{T}_\ell$. Therefore, we define the majority voting function as follows:

$$f_{MV}(T, \mathcal{M}_{cl}) = \underset{D \in \mathcal{M}_{cl}(\mathbb{T}_\ell)}{\text{argmax}} \sum_{T_{i,\ell} \in \mathbb{T}_\ell} \mathbb{1}_{[\mathcal{M}_{cl}(T_{i,\ell})=D]}$$

Majority voting serves the pipeline with two significant factors, (i) it does not depend on the design of the detector and makes the pipeline easily usable for multiple different types of anomaly detection methods, and (ii) majority voting averages the predictions and reduces the impact of misclassified subsequences. To conclude, in our pipeline, the model selection method introduced in Problem 1 is the output of $f_{MV}(T, \mathcal{M}_{cl})$.

5 EXPERIMENTAL EVALUATION

We now describe in detail our experimental analysis. For additional information, we make all our material publicly available online [20] and provide an interactive WebApp [21] for navigating and exploring the experimental results.

Technical setup: We implemented the deep learning-based model selection methods in Python 3.5 using the PyTorch library [62]. For the feature-based approach, we used the tsfresh [26] and scikit-learn [63] libraries. We then used sktime [52] for the rocket algorithm implementation. For the anomaly detection methods, we used the implementation provided in the TSB-UAD benchmark [61]. The evaluation was conducted on a server with Intel Core i7-8750H CPU 2.20GHz x 12, with 31.3GB RAM, and Quadro P1000/PCIe/SSE2 GPU with 4.2GB RAM, and on Jean Zay cluster with Nvidia Tesla V100 SXM2 GPU with 32 GB RAM.

Datasets: For our evaluation purposes, we use the public datasets identified in the TSB-UAD benchmark [61]. The latter corresponds to 16 datasets (described in Table 1) proposed in the literature containing 1900 time series with labeled anomalies. Specifically, each point in every time series is labeled as normal or abnormal.

Anomaly Detection Methods: For the experimental evaluation, We select 12 different anomaly detection methods, summarized in Table 1. Out of these, 8 are fully unsupervised (i.e., they require no prior information on the anomalies to be detected): IForest, IForest1, LOF, MP, NormA, PCA, HBOS, and POLY. The remaining 4 methods are semi-supervised (i.e., they require some information related to the normal behavior), namely, OCSVM, AE, LSTM-AD, and CNN. For all these anomaly detection baselines, we set the parameter as described in the TSB-UAD benchmark [61].

Method Selection baselines: We then consider the method selection baseline described in Section 4 and summarized in Table 1. We first consider *feature-based* methods, that extract features using tsfresh [26] library to select the correct anomaly detection method. We then consider rocket, the state of the art *time series classifier*. We also include two types of deep learning classifier; (i) *Convolutional-based neural networks* and (ii) *Tranformer-based neural networks*. Table 1 summarizes the different model selection methods (i.e., classifiers). In total, we consider 16 methods, trained with windows lengths ℓ equals to 16, 32, 64, 128, 256, 512, 768 and 1024. In total, we trained 128 models. In the following section we refer to a model M trained using a window length ℓ as M-ℓ.

Parameter settings: We use the same 70/30 split of the benchmark for all the classification models. Therefore, we can compare models trained on the same training set and evaluated on the same set of time series. Then, For the feature-based methods, We set the hyper-parameters of the models based on the default parameters of scikit-learn. Moreover, for rocket, we use 10000 kernels to extract

Datasets	Description
Dodgers [43]	unusual traffic after a Dodgers game **(1 time series)**
ECG [55]	standard electrocardiogram dataset **(52 time series)**
IOPS [1]	performance indicators of a machine **(58 time series)**
KDD21 [45]	composite dataset released in a recent SIGKDD 2021 **(250 time series)**
MGAB [73]	Mackey-Glass time series with non-trivial anomalies **(10 time series)**
NAB [5]	Web-related real-world and artificial time series **(58 time series)**
SensorScope [79]	environmental data **(23 time series)**
YAHOO [47]	real and synthetic time series based on Yahoo production systems **(367 time series)**
Daphnet [8]	acceleration sensors on Parkinson's disease patients **(45 time series)**
GHL [33]	Gasoil Heating Loop telemetry **(126 time series)**
Genesis [75]	portable pick-and-place demonstrator **(6 time series)**
MITDB [55]	ambulatory ECG recordings **(32 time series)**
OPPORTUNITY [65]	motion sensors for human activity recognition **(465 time series)**
Occupancy [25]	temperature, humidity, light, and CO2 of a room **(10 time series)**
SMD [71]	Server Machine telemetry **(281 time series)**
SVDB [39]	ECG recordings **(115 time series)**
Anomaly Detection	**Description**
IForest [50]	It constructs binary trees based on random space splitting. The nodes (i.e., subsequences) with shorter path lengths to the root are more likely to be anomalies.
IForest1 [50]	same as IForest, but each point (individually) are used as input.
LOF [22]	It computes the ratio of the neighboring density to the local density.
MP [81]	It detects as anomaly the subsequence with the most significant nearest neighbor distance.
NormA [15]	It identifies the normal patterns based on clustering and calculates each point's effective distance to the normal patterns weighted using statistical criteria of these patterns.
PCA [3]	It projects data to a lower-dimensional hyperplane, and data points with a significant distance from this plane can be identified as outliers.
AE [66]	projects data to the lower-dimensional latent space and reconstructs the data, and outliers are expected to have more evident reconstruction deviation.
LSTM-AD [54]	use an LSTM network that from the current subsequence tries to predict the following value. The error prediction is then used to identify anomalies.
POLY [48]	fits a polynomial model that tries to predict the values of the data series from the previous subsequences. The outliers are detected by measuring the prediction error.
CNN [56]	build, using a convolutional-based deep neural network, a correlation between current and previous subsequences, and the outliers are detected by the deviation between the prediction and the actual value.
OCSVM [68]	is a support vector method that fits the normal training dataset and finds the normal data's boundary.
HBOS [37]	constructs a histogram for the data and the inverse of the height of the bin is used as the outlier score of the data point.
Model Selection	**Description**
Feature-based classifier	
SVC	maps training examples to points in space so as to maximize the gap between the two categories.
Bayes	uses Bayes' theorem to predict the class of a new data point using the posterior probabilities for each class
MLP	consists of multiple layers of interconnected neurons
QDA	is a discriminant analysis algorithm for classification problems
AdaBoost	is a meta-algorithm using boosting technique with weak classifiers
Decision Tree	is a tree-based approach that split data point into separate leaves based on features
Random Forest	is an ensemble Decision Trees fed with random sample (with replacement) of the training set and random set of features.
kNN	assigns the most common class among its k nearest neighbors.
Time series classifier	
Rocket	transforms input time series using a small set of convolutional kernels, and uses the transformed features to train a linear classifier
Convolutional-based neural networks	
ConvNet	uses convolutional layers to automatically and adaptively learn spatial hierarchies of features from input data.
ResNet	It is a ConvNet with residual connections between convolutional block
Inception Time	is a combination of ResNets with kernels of multiple sizes
Transformer-based neural networks	
SIT-conv	is a transformer architecture with a convolutional layer as input
SIT-linear	is a transformer architecture for which the time series are divided into non-overlapping patches and linearly projected into the embedding space
SIT-stem	is a transformer architecture with convolutional layers with increasing dimensionality as input
SIT-stem-ReLU	is similar to SIT-stem but with Scaled ReLU.

Table 1: Summary of datasets, methods, and measures.

the features and the logistic regression with stochastic gradient descent (computed in batches) for the classification step. Finally, for Convolutional and Transformer-based methods, we use a learning rate of 10^{-5}, with a batch size of 256 and an early stopping strategy with a maximum of 50 epochs without improvement. Moreover, We use the weighted cross-entropy loss and set the maximum number of epochs to 10,000 (with a training time limit of 20 hours).

Evaluation measures: We finally use four evaluation measures For model selection accuracy, we use the classification accuracy (i.e., the number of models correctly selected divided the total number of time series). For anomaly detection accuracy, we use both AUC-PR [28] and VUS-PR [60] (with a buffer length equals to 10 points). For execution time, we measure the training time (i.e., the time required to train a model selection algorithm), the prediction time (i.e., the time that a a model needs to predict which model to use), and the inference time (i.e., the time require to predict which detector to use, and to run it).

5.1 Overall Evaluation

We first conduct an extensive evaluation of accuracy (classification and anomaly detection) and execution time for all model selection methods over the entire benchmark. thus, we split the benchmark into one training and testing set. The first contains 1404 time series and the second 496. Both sets contains time series from all datasets. Therefore, the models have examples of all available domains. In Section 5.5, we evaluate the performance of model selection when applied to unseen (i.e., not used in the training set) datasets.

5.1.1 **Accuracy Evaluation**. We first analyze the accuracy of all model selection methods (using all window lengths) and compare them to the Oracle , the Averaging ensembling method , and anomaly detection methods in the TSB-UAD benchmark.

Figure 5(a) depict the overall VUS-PR over the entire TSB-UAD benchmark (i.e., each box-plot corresponds to 497 accuracy values for the 497 time series into the test set). The Convoutional-based approaches are in blue, the Transformer-based approaches are in red, the Feature-based approaches are in green, Rocket models are in dark grey, and the anomaly detection methods of the TSB-UAD benchmark are in light blue. The oracle is the top box plot (in white), and the Averaging ensembling is the dark red box plot. The box-plot are sorted based on the median value. In total, we compare 142 models on 497 time series.

First, we observe that almost all model selection methods outperform the existing anomaly detection methods. We also see that most model selection methods outperform the Averaging ensembling approach. Thus, we can conclude that model selection using time series classifiers significantly improves the state of the art for anomaly detection in time series.

More interestingly, we observe a partition in the ranking of the methods. First, Convolutional and Transformer-based approaches produce equivalent accuracy values and represent the top-48 methods. However, whereas all the convolutional-based methods are in the top-48, a few of the Transformer-based approaches are further away in the ranking. Moreover, the first non-deep learning method is *rocket*-128 (ranked 49), followed closely by *knn* models. We also observe that the *rocket* approaches are very spread across the ranking (*rocket*-128 is ranked 50, and *rocket*-16 is ranked 124). This implies that the choice of window length strongly impacts accuracy. Overall, we note that the best selection model is 2.8 times more accurate than the best anomaly detection method in TSB-UAD.

Then, we also note that all the model selection methods are significantly less accurate than the Oracle. For example, in Figure 5(a), there is a gap of 0.2 VUS-PR between the Oracle and the best model selection method. Such a gap is significant and indicates a large margin of improvement for future work. We also note that all model selection approaches produce accuracy values between 0 and 1 (as shown by each box-plot in Figure 5(a)). This means that no model selection method is guaranteed to perform above a given accuracy

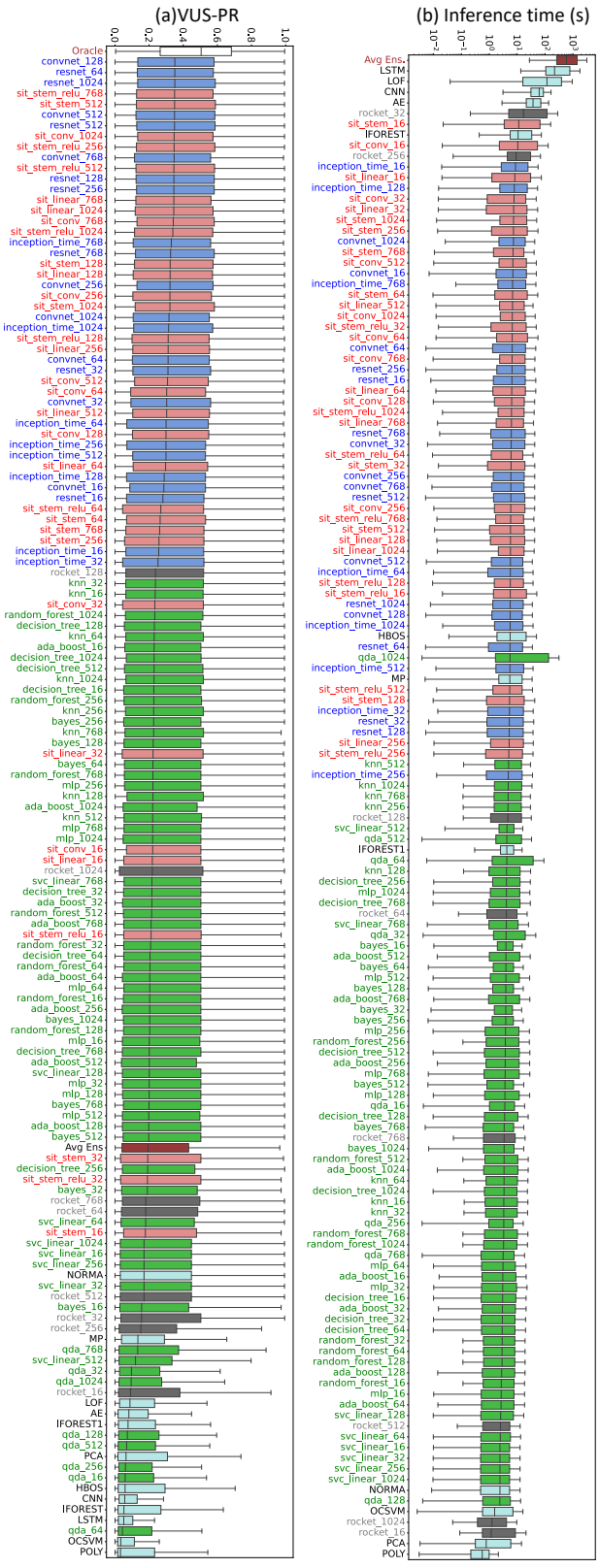

Figure 5: VUS-PR and inference execution time (in seconds) for all method selection methods and configurations over a test set of 497 time series from the TSB-UAD benchmark.

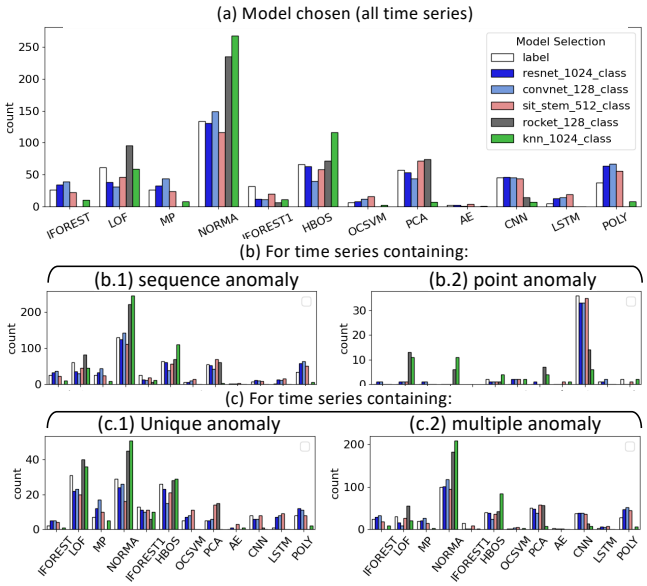

Figure 6: Distribution of the selected models for five models (the best for each category) compared to the distribution of the labels (in white). Difference of distributions between time series containing (b) sequence and point anomalies, and (c) unique or multiple anomalies.

value. Rendering model selection more stable and robust is essential for several use cases, and is an interesting open problem.

5.1.2 **Model selected distribution**. We then inspect in detail the prediction and the detector chosen by the model selection approaches. In this section, we consider only *resnet*-1024, *convnet*-128, sit_stem_512, $rocket_128$, and *knn*-1024. These approaches are the best models (using either AUC-PR or VUS-PR) based on the analysis conducted in Section 5.1 (you may find additional information on AUC-PR evaluation in our website [21]).

Figure 6(a) depicts the distribution of the chosen detectors by the 5 model selection approaches mentioned above for the entire TSB-UAD benchmark. The white bar corresponds to the true labels (i.e., the best detectors). We observe from Figure 6(a) that *rocket*-128 and *knn*-1024 are significantly overestimating the detector NormA (as well as LOF for *rocket*-128 and HBOS for *knn*-1024), whereas *resnet*-1024, *convnet*-128, and *sit-stem*-512 are matching the correct distribution of detectors (we observe a slight underestimation of LOF, IFOREST1 and an overestimation for POLY).

Moreover, we measure the prediction distribution differences for time series containing sequence anomalies (Figure 6(b.1)) and point anomalies (Figure 6(b.2)), and for time series containing only one anomaly (Figure 6(c.1)) and multiple anomalies (Figure 6(c.1)). We first observe that predictions of model selection methods are significantly different for time series with sequence and point anomalies. More specifically, *resnet*-1024, *convnet*-128, and *sit-stem*-512 are correctly selecting the method CNN, whereas *rocket*-128 and *knn*-1024 are overestimating LOF and NormA for time series containing point anomalies. However, for sequence anomaly, as it represents most of the TSB-UAD benchmark, the prediction distribution is similar to the one over the entire benchmark. Moreover, the correct predictions of *resnet*-1024, *convnet*-128, and *sit-stem*-512 for time series containing point anomalies are interesting, as this information is not provided in the training step. Therefore, These models

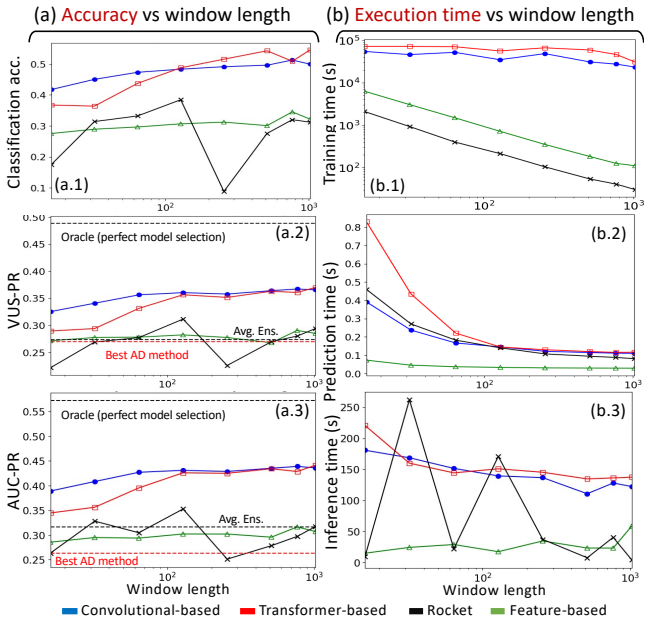

Figure 7: (a) Accuracy ((a.1) classification accuracy, (a.2) VUS-PR and (a.3) AUC-PR) and (b) execution time ((b.1) training time, (b.2) prediction time and (b.3) inference time) versus window length ℓ.

found discriminant features in the time series that indicate whether it might contain a point or a sequence anomaly.

We, finally, measure the differences between the prediction distribution of model selection methods between time series containing unique and multiple anomalies. The true labels (white bars in Figure 6(c.1) and (c.2)) indicates that, for unique anomalies, the best detectors are LOF, NormA, and IFOREST1 and for multiple anomalies, the best detector is NormA. We observe that all model selection approaches correctly predict more LOF, NormA, and IFOREST1 for time series containing unique anomaly. The latter indicates that model selection methods can extract discriminant features that indicate if one time series is more likely to have multiple anomalies.

5.1.3 **Execution Time Evaluation**. We now discuss the execution time of model selection methods. In this section, we focus only on the inference time (i.e., the number of seconds required by a method to predict the detector to use and to run it). Figure 5(b) depicts the inference time (in log scale) for each method and detector in the TSB-UAD benchmark. We first observe that the Averaging ensembling required to run all detectors is significantly slower than the rest. Then, all model selection methods are of the same order of magnitude as the detectors. We also observe that all the deep learning methods are slower than the feature-based approaches. This is surprising because the inference time mainly depends on the chosen detector. Overall, we conclude that method selection is the only viable solution that outperforms the existing anomaly detection methods and can be executed in the same order of magnitude of time.

5.2 Influence of the Window Length

In this section, we analyze the influence of the window length on classification accuracy (Figure 7(a.1)), anomaly detection accuracy (Figure 7(a.2) and (a.3)) and execution time (Figure 7(b)). We perform

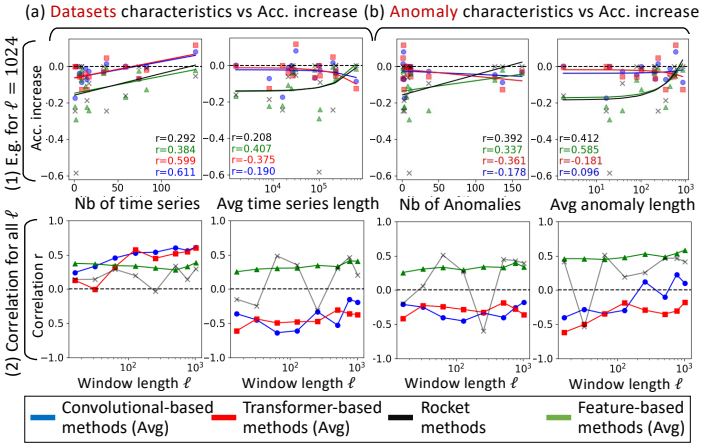

Figure 8: Average accuracy increase (the difference between VUS-PR of a family of methods in average with the most accurate anomaly detection methods in TSB) for each dataset. Correlation between accuracy increase and the window length used to train the model selection methods.

the analysis per group of methods (i.e., average performances for Convolutional, Transformer, rocket, and Feature-based methods).

We first observe in Figure 7(a) that Convolutional-based and Transformer-based methods outperform the best anomaly detection methods (red dotted line in Figure 7(a.2) and (a.3)), the Averaging ensembling approach (black dotted line in Figure 7(a.2) and (a.3)), Rocket and Feature-based methods whatever the length used with regards to the classification accuracy, VUS-PR, and AUC-PR. We also observe that Transformer-based approaches are less accurate for shorter lengths (less than 100 points), whereas, Convolutional-based approaches are more stable to the length. Overall, we observe that Transformer and Convolutional-based approach performances converge to the same anomaly detection accuracy (both for VUS-PR and AUC-PR) when the window length increases.

Furthermore, we observe that both rocket and Feature-based approaches are significantly faster to be trained than Convolutional and Transformer-based approaches (Figure 7(b.1)). We make the same observation for prediction time ((Figure 7(b.2))). For Inference time, we observe that rocket execution time is very unstable when compared to the other approaches. The latter means that the choice of length strongly impacts the model selection performed by rocket, leading to very diverse selection and execution times.

In the general case, we can make the following two statements: (i) The bigger the window length, the more accurate and faster the model selection process is. (ii) Feature-based approaches are significantly faster but less accurate than Convolutional-based and Transformer based approaches, whatever the length used.

5.3 Influence of Datasets and Anomaly Types

In this section, we evaluate the influence of datasets and anomaly characteristics on model selection accuracy. We perform the analysis per group of methods (i.e., average performances for Convolutional, Transformer, rocket, and Feature-based methods).

For this experiment, we evaluate the dataset and anomaly characteristics (i.e., the number of time series, the average length of the time series, the average number of anomalies and the average anomaly length). Figure 8 depicts these characteristics (x-axis)

versus the average increase of accuracy (VUS-PR of the model selection method subtracted by VUS-PR of the best anomaly detection method for each dataset) for each model selection method using a given window length. For instance, If a point (one model selection method on one dataset) is positive (above the black dotted line), thus the corresponding model is more accurate on the corresponding dataset than the best anomaly detection method selected on this same dataset. Figure 8(1) shows a specific example for window length $\ell = 1024$. Figure 8(2) shows the correlation values (trend of the different lines in Figure 8(1)) versus the window length.

We generally observe low correlations between dataset and anomaly characteristics (i.e., $-0.6 < r < 0.6$). With such correlation values, we cannot conclude any factual statement on the impact of these characteristics and the model selection methods' performances. However, we can make the following observations.

First, Figure 1(a.1) shows that all model selection methods (with a window length $\ell = 1024$) are more likely to be accurate on large datasets. Figure 1(a.2) shows that the number of time series is impacting more substantially Convolutional and Transformer-based approaches with large window lengths. For the average time series length, only Feature-based approaches are positively impacted. On the contrary, Convolutional and Transformer-based approaches are less accurate when the average time series length is increasing. These observations imply that Convolutional and Transformer-based are more affected by the number of examples in the dataset rather than the length of each instance. In contrast, Feature-based approaches benefit from both more and large instances.

Then, Figure 1(b.2) shows that Feature-based approach accuracy is increasing with the anomaly characteristics, whereas these characteristics either do not or negatively impact Convolutional and Transformer-based methods. More specifically, we observe that Feature-based approaches (regardless of the window length) are more accurate with time series containing large anomalies, and Convolutional based approaches are less accurate (irrespective of the window length) when the number of anomalies increases.

In general, we note that rocket correlation with the dataset and the anomaly characteristics is unstable. The latter is explained by the fact that the model prediction of rocket is very sensitive to the window length (as described in Section 5.2). Thus, it is impossible to conclude any link between rocket performances, datasets, and anomaly characteristics.

5.4 Detection vs Classification Accuracy

In this section, we analyze the relationship between the model selection methods' classification accuracy and the resulting anomaly detection accuracy. In this experiment, we consider VUS-PR as anomaly detection measures. For this experiment, we extend the definition of *Oracle* (introduced in Section 3) as follows:

Definition 2. *For a given dataset $\mathcal{D}$, we define $Oracle_{k,j}$ as a hypothetical model selection method that has a classification accuracy of $k \in [0,1]$ and selects the j^{th} best detector (among m detectors) in cases of misclassification. Thus, $Oracle_{1,1}$ always selects the best detector, and $Oracle_{0,m}$ always selects the worst detector. Finally, we define $Oracle_{k,R}$ as the model selection method that has a classification accuracy of $k \in [0,1]$ and randomly selects a detector in misclassification cases.*

Figure 9 depicts the latter comparison for four specific datasets (Figure 9(a)) and for all datasets (Figure 9(b)). We first observe a

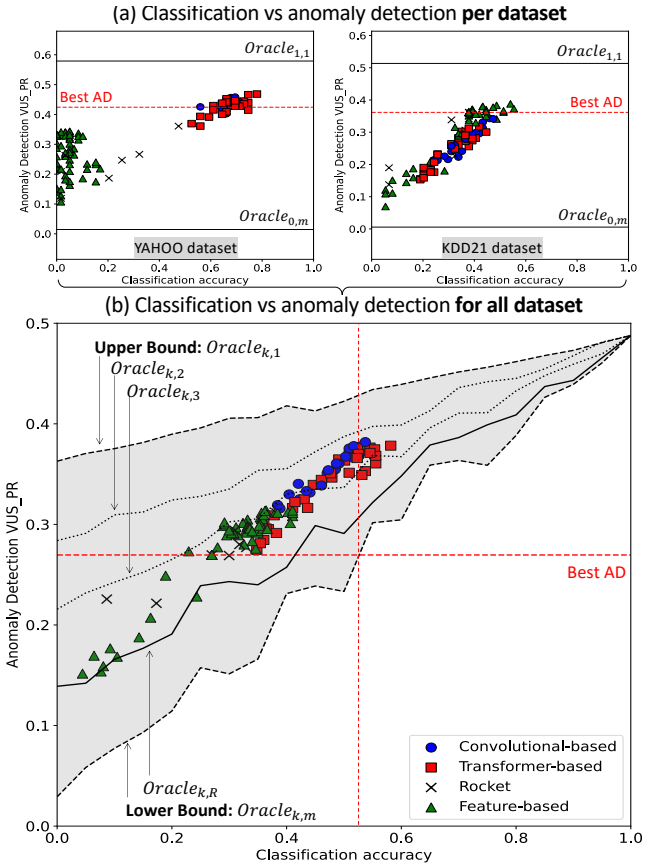

Figure 9: Classification accuracy versus anomaly detection accuracy (VUS-PR) for (a) four datasets and for (b) all datasets.

strong correlation between classification accuracy and anomaly detection accuracy for each specific dataset and, on average, for all datasets. However, methods belonging to different families (e.g., Feature-based or Transformer-based) are not performing the same. For instance, Figure 9(a) shows that Feature-based approaches are not accurate for the YAHOO dataset but are the best models for the KDD21 dataset. Overall, we observe that Convolutional-based and Transformer-based are both more accurate in classification and anomaly detection (Figure 9(b)).

We also depict in Figure 9(b) the lines corresponding to $Oracle_{k,1}$, $Oracle_{k,2}$, $Oracle_{k,3}$, $Oracle_{k,R}$, and $Oracle_{k,m}$. For a given classification accuracy, k, $Oracle_{k,1}$, and $Oracle_{k,m}$ correspond to the upper and lower bounds. The latter means any model selection that cannot be outside these bounds (i.e., outside of the grey zone in Figure 9(b)). Therefore, for the TSB benchmark, any model selection method that has a classification accuracy above 0.53 (intersection between the two dotted red lines) is guaranteed to be better than the currently existing best anomaly detection method in TSB-UAD (i.e., best AD in Figure 9(b)). In our experiments, this is the case only for a few Convolutional- and Transformer-based methods.

Moreover, we compare the positions of the model selection methods with regards to $Oracle_{k,2}$, $Oracle_{k,3}$, and $Oracle_{k,R}$. We observe in Figure 9(b) that almost all methods are above $Oracle_{k,R}$. The latter means that when the wrong detector is selected, the model selection methods do not select detectors randomly. Moreover, we

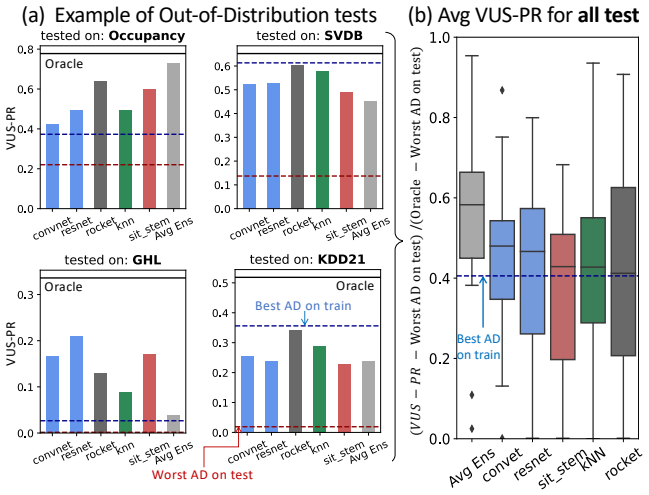

Figure 10: Out-of-distribution experiment when model selection algorithms are trained on all but one dataset. (a) results for each dataset (when not included in the training set) and (b) average results.

observe that most models follow the $Oracle_{k,3}$ line. The latter indicates that the models averagely select the third-best in case of misclassification. Finally, the observations discussed above demonstrate three important statements: (i) Classification accuracy can be used as a proxy for anomaly detection accuracy, and without computing the anomaly detection accuracy, we can provide an anomaly detection accuracy lower and upper bounds. (ii) the gap between the best model selection and the top right corner of the grey zone shows that there is a significant margin of improvement for future work. (iii) The vertical gap between the models and the upper bound ($Oracle_{k,1}$) shows that there is an important margin of improvement in the prediction rank: e.g., a model with the same classification accuracy can gain up to 0.1 VUS-PR if it selects models in better order.

5.5 Out-of-Distribution Experiments

At this point, we tested the performances of the model selection methods when trained on a subset of the benchmark with examples from all 16 datasets available. In some cases though, we may want to analyze time series that are not similar to any of those in the benchmark. Therefore, in this section, we measure the ability of the model selection methods to be used in an unsupervised manner (i.e., used for datasets that are not similar to the one used in the training set). We run the following experiment. We train the model selection methods on 15 datasets (70% percent of the time series for training and the other 30% for validation), and we test on the remaining one. We try all 16 possible test partitions, and (for brevity) report 4 of these tests in Figure 10(a). We only show the results for the best-performing model selection methods listed in Section 5.1.2.

Figure 10(b) depicts the normalized VUS-PR for all 16 tests: VUS-PR of 1 corresponds to the VUS-PR of the *Oracle* on each test, while 0 corresponds to the worst anomaly detection methods on each test. This figure shows that, in the unsupervised case, the Averaging ensembling is outperforming all model selection methods, as well as the best anomaly detection method selected on the train set (dotted blue line in Figure 10(b)). The latter means that, for unknown datasets, it is safer to run all existing anomaly detection methods and average their scores. Knowing that such ensembling methods are not scalable (as shown in Figure 5), Figure 10(b) shows that ConvNet or ResNet are still a better choice than choosing the best anomaly detection method selected on train data (i.e., known data). However, knn, rocket, and sit-stem are only slightly more accurate than the best anomaly detection method.

Figure 10(a) depicts the average accuracy per test (i.e., dataset not included in the training set and used for the test). We observe very different results. First, for Electrocardiograms (SVDB), none of the model selection methods and the Averaging ensembling outperforms the best anomaly detection method (selected on the training set). But, most model selection methods or Averaging ensembling outperform the best anomaly detection method for sensor data of different kinds (GHL and Occupancy) on the train. The latter can be explained by the fact that ECGs contain less heterogeneous behaviors (i.e., repetitive normal behavior and similar anomalies) than measurements from sensors data, and it is more likely to have in the benchmark one method that would perform very well on all time series.

These observations lead to the following remarks: (i) There is a significant margin of improvement when using the existing time series classifier as model selection method in the unsupervised case. (ii) When a new dataset arrives, it is safer in the general case to use an ensembling method such as the simple average of all anomaly scores. (ii) For heterogeneous datasets (sensor measurement without any known and repetitive normal or abnormal patterns), classifiers as model selection (mainly convolutional-based classifiers) can be used even though similar time series are not contained in the training set.

6 CONCLUSIONS

Time series anomaly detection is a challenging problem and an important area of research with applications in many scientific, societal, and industrial domains. Despite the multitude of solutions proposed in the literature, we observed that there exists no method that outperforms the other when measured on large heterogeneous benchmarks. Based on our experimental evaluation, we answer the questions of Section 3.5 as follows:

(1) **Classification as Model selection**: We observe that time series classification methods accurately select anomaly detection models. Overall, Transformer and Convolutional-based model selection methods outperform each individual detector. Nevertheless, there is a large gap between the best method and the *Oracle*, motivating future work toward that direction.
(2) **Ensembling or selecting**: We observe that model selection is significantly more accurate than the Ensembling method.
(3) **Features or Raw values**: We observe that raw-based methods are more accurate on average than feature-based approaches.
(4) **Out-Of-Distribution**: (1) and (3) hold. However, for (2), We observe that Ensembling is more accurate than model selection when applied to time series very different from those in the training benchmark.

The observations provided above point to promising directions for future work in AutoML frameworks that rely on model selection. As mentioned in Section 5.4, improving the rank prediction could significantly improve the anomaly detection accuracy. Moreover, model selection could be trained to choose the best compromise between accuracy and execution time, improving the overall inference time of model selection.

REFERENCES

[1] [n.d.]. http://iops.ai/dataset_detail/?id=10.

[2] Charu C Aggarwal. 2017. An introduction to outlier analysis. In *Outlier analysis*. Springer, 1–34.

[3] Charu C. Aggarwal. 2017. *Outlier Analysis* (2 ed.). Springer International Publishing. https://doi.org/10.1007/978-3-319-47578-3

[4] Charu C. Aggarwal and Saket Sathe. 2015. Theoretical Foundations and Algorithms for Outlier Ensembles. *SIGKDD Explor. Newsl.* 17, 1 (sep 2015), 24–47. https://doi.org/10.1145/2830544.2830549

[5] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. 2017. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* 262 (2017), 134–147. https://doi.org/10.1016/j.neucom.2017.04.070

[6] Jérôme Antoni and Pietro Borghesani. 2019. A statistical methodology for the design of condition indicators. *Mechanical Systems and Signal Processing* 114 (2019), 290–327. https://doi.org/10.1016/j.ymssp.2018.05.012

[7] Martin Bach-Andersen, Bo Rømer-Odgaard, and Ole Winther. 2017. Flexible non-linear predictive models for large-scale wind turbine diagnostics. *Wind Energy* 20, 5 (2017), 753–764.

[8] Marc Bachlin, Meir Plotnik, Daniel Roggen, Inbal Maidan, Jeffrey M. Hausdorff, Nir Giladi, and Gerhard Troster. 2010. Wearable Assistant for Parkinson's Disease Patients With the Freezing of Gait Symptom. *IEEE Transactions on Information Technology in Biomedicine* 14, 2 (2010), 436–446. https://doi.org/10.1109/TITB.2009.2036165

[9] Anthony Bagnall, Richard L. Cole, Themis Palpanas, and Kostas Zoumpatianos. 2019. Data Series Management (Dagstuhl Seminar 19282). *Dagstuhl Reports* 9, 7 (2019), 24–39. https://doi.org/10.4230/DagRep.9.7.24

[10] Ziv Bar-Joseph, Georg K Gerber, David K Gifford, Tommi S Jaakkola, and Itamar Simon. 2003. Continuous representations of time-series gene expression data. *Journal of Computational Biology* 10, 3-4 (2003), 341–356.

[11] V. Barnet and T. Lewis. 1994. *Outliers in Statistical Data*. John Wiley and Sons, Inc.

[12] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. 2021. A Review on outlier/Anomaly Detection in Time Series Data. *ACM Computing Surveys (CSUR)* 54, 3 (2021), 1–33.

[13] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. 2020. Automated Anomaly Detection in Large Sequences. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 1834–1837. https://doi.org/10.1109/ICDE48307.2020.00182

[14] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. 2020. SAD: An Unsupervised System for Subsequence Anomaly Detection. In *ICDE*. https://doi.org/10.1109/ICDE48307.2020.00168

[15] Paul Boniol, Michele Linardi, Federico Roncallo, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. 2021. Unsupervised and scalable subsequence anomaly detection in large data series. *The VLDB Journal* (March 2021). https://doi.org/10.1007/s00778-021-00655-8

[16] Paul Boniol, Mohammed Meftah, Emmanuel Remy, and Themis Palpanas. 2022. dCAM: Dimension-wise Class Activation Map for Explaining Multivariate Data Series Classification. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1175–1189. https://doi.org/10.1145/3514221.3526183

[17] Paul Boniol and Themis Palpanas. 2020. Series2Graph: Graph-Based Subsequence Anomaly Detection for Time Series. *Proc. VLDB Endow.* 13, 12 (July 2020), 1821–1834. https://doi.org/10.14778/3407790.3407792

[18] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. 2021. SAND in action: subsequence anomaly detection for streams. *Proceedings of the VLDB Endowment* 14, 12 (2021).

[19] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. 2021. SAND: streaming subsequence anomaly detection.

[20] P. Boniol and E. Sylligardos. 2023. Our open-source code for this paper. https://github.com/boniolp/MSAD.

[21] P. Boniol and E. Sylligardos. 2023. Our website. https://adecimots.streamlit.app/.

[22] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*. ACM, New York, NY, USA, 93–104. https://doi.org/10.1145/342009.335388

[23] Peter J Brockwell and Richard A Davis. 2016. *Introduction to time series and forecasting*. springer.

[24] Yingyi Bu, Oscar Tat-Wing Leung, Ada Wai-Chee Fu, Eamonn J. Keogh, Jian Pei, and Sam Meshkin. 2007. WAT: Finding Top-K Discords in Time Series Database. In *SDM*.

[25] Luis M. Candanedo and Véronique Feldheim. 2016. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buildings* 112 (2016), 28–39. https://doi.org/10.1016/j.enbuild.2015.11.071

[26] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing* 307 (2018), 72–77. https://doi.org/10.1016/j.neucom.2018.03.067

[27] Maximilian Christ, Andreas W Kempa-Liehr, and Michael Feindt. 2016. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint arXiv:1610.07717* (2016).

[28] Jesse Davis and Mark Goadrich. 2006. The Relationship between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*. Association for Computing Machinery, New York, NY, USA, 233–240. https://doi.org/10.1145/1143844.1143874

[29] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. 2021. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 248–257.

[30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[31] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33, 4 (2019), 917–963.

[32] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[33] Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. 2016. Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. arXiv:1612.06676 [cs.LG]

[34] Anthony J Fox. 1972. Outliers in time series. *Journal of the Royal Statistical Society: Series B (Methodological)* 34, 3 (1972), 350–363.

[35] Ada Wai-chee Fu, Oscar Tat-Wing Leung, Eamonn Keogh, and Jessica Lin. 2006. Finding Time Series Discords Based on Haar Transform. In *Proceedings of the Second International Conference on Advanced Data Mining and Applications* (Xi'an, China) *(ADMA'06)*. Springer-Verlag, Berlin, Heidelberg, 31–41. https://doi.org/10.1007/11811305_3

[36] Steve Goddard, Sherri K Harms, Stephen E Reichenbach, Tsegaye Tadesse, and William J Waltman. 2003. Geospatial decision support for drought risk management. *Commun. ACM* 46, 1 (2003), 35–37.

[37] Markus Goldstein and Andreas Dengel. 2012. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track* 9 (2012).

[38] Mononito Goswami, Cristian Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. 2022. Unsupervised Model Selection for Time-series Anomaly Detection. https://doi.org/10.48550/ARXIV.2210.01078

[39] Scott David Greenwald. 1990. *Improved detection and classification of arrhythmias in noise-corrupted electrocardiograms using contextual information*. Thesis. Massachusetts Institute of Technology. https://dspace.mit.edu/handle/1721.1/29206 Accepted: 2005-10-07T20:45:22Z.

[40] Medina Hadjem, Farid Naït-Abdesselam, and Ashfaq Khokhar. 2016. ST-segment and T-wave anomalies prediction in an ECG data using RUSBoost. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*. 1–6. https://doi.org/10.1109/HealthCom.2016.7749493

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. https://doi.org/10.48550/ARXIV.1512.03385

[42] Pablo Huijse, Pablo A Estevez, Pavlos Protopapas, Jose C Principe, and Pablo Zegers. 2014. Computational intelligence challenges and applications on large-scale astronomical time series databases. *IEEE Computational Intelligence Magazine* 9, 3 (2014), 27–39.

[43] Alexander Ihler, Jon Hutchins, and Padhraic Smyth. 2006. Adaptive Event Detection with Time-Varying Poisson Processes. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Philadelphia, PA, USA) *(KDD '06)*. Association for Computing Machinery, New York, NY, USA, 207–216. https://doi.org/10.1145/1150402.1150428

[44] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. 2021. Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series. *Proc. VLDB Endow.* 14, 11 (oct 2021), 2613–2626.

[45] E. Keogh, T. Dutta Roy, U. Naik, and A Agrawal. 2021. Multi-dataset Time-Series Anomaly Detection Competition 2021. https://compete.hexagon-ml.com/practice/competition/39/.

[46] Eamonn J. Keogh, Stefano Lonardi, Chotirat Ratanamahatana, Li Wei, Sanghee Lee, and John C. Handley. 2006. Compression-based data mining of sequential data. *Data Mining and Knowledge Discovery* 14 (2006), 99–129.

[47] N. Laptev, S. Amizadeh, and Y. Billawala. 2015. *S5 - A Labeled Anomaly Detection Dataset, version 1.0(16M)*. https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70

[48] Zhi Li, Hong Ma, and Yongbing Mei. 2007. A Unifying Method for Outlier and Change Detection from Data Streams Based on Local Polynomial Fitting. In *Advances in Knowledge Discovery and Data Mining (Lecture Notes in Computer Science)*, Zhi-Hua Zhou, Hang Li, and Qiang Yang (Eds.). Springer, Berlin, Heidelberg, 150–161. https://doi.org/10.1007/978-3-540-71701-0_17

[49] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn J. Keogh. 2020. Matrix profile goes MAD: variable-length motif and discord discovery in data series. *Data Min. Knowl. Discov.* 34, 4 (2020), 1022–1071. https://doi.org/10.1007/s10618-

020-00685-w

[50] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*. 413–422. https://doi.org/10.1109/ICDM.2008.17 ISSN: 2374-8486.

[51] Yubao Liu, Xiuwei Chen, and Fei Wang. 2009. Efficient Detection of Discords for Time Series Stream. *Advances in Data and Web Management* (2009), 629–634. http://www.springerlink.com/index/n546h380446p95r7.pdf

[52] Markus Löning, A. Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J. Király. 2019. sktime: A Unified Interface for Machine Learning with Time Series. *ArXiv* abs/1909.07872 (2019).

[53] Haoran Ma, Benyamin Ghojogh, Maria N. Samad, Dongyu Zheng, and Mark Crowley. 2020. Isolation Mondrian Forest for Batch and Online Anomaly Detection. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 3051–3058. https://doi.org/10.1109/SMC42975.2020.9283073

[54] Pankaj Malhotra, L. Vig, Gautam M. Shroff, and Puneet Agarwal. 2015. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *ESANN*.

[55] G.B. Moody and R.G. Mark. 2001. The impact of the MIT-BIH Arrhythmia Database. *IEEE Engineering in Medicine and Biology Magazine* 20, 3 (2001), 45–50. https://doi.org/10.1109/51.932724

[56] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed. 2019. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access* 7 (2019), 1991–2005. https://doi.org/10.1109/ACCESS.2018.2886457

[57] Keiron O'Shea and Ryan Nash. 2015. An Introduction to Convolutional Neural Networks. *CoRR* abs/1511.08458 (2015). arXiv:1511.08458 http://arxiv.org/abs/1511.08458

[58] ES Page. 1957. On problems in which a change in a parameter occurs at an unknown point. *Biometrika* 44, 1/2 (1957), 248–252.

[59] Themis Palpanas and Volker Beckmann. 2019. Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA). *SIGMOD Rec.* 48, 3 (Dec. 2019), 36–40. https://doi.org/10.1145/3377391.3377400

[60] John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S. Tsay, Aaron Elmore, and Michael J. Franklin. 2022. Volume under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection. *Proc. VLDB Endow.* 15, 11 (2022).

[61] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin. 2022. TSB-UAD: An End-to-End Benchmark Suite for Univariate Time-Series Anomaly Detection. *Proc. VLDB Endow.* 15, 8 (2022).

[62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, Vol. 32.

[63] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[64] Joshua S Richman and J Randall Moorman. 2000. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology* 278, 6 (2000), H2039–H2049.

[65] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and José del R. Millàn. 2010. Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh International Conference on Networked Sensing Systems (INSS)*. 233–240. https://doi.org/10.1109/INSS.2010.5573462

[66] Mayu Sakurada and Takehisa Yairi. 2014. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis* (Gold Coast, Australia QLD, Australia) *(MLSDA'14)*. Association for Computing Machinery, New York, NY, USA, 4–11. https://doi.org/10.1145/2689746.2689747

[67] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly Detection in Time Series: A Comprehensive Evaluation. *Proc. VLDB Endow.* 15, 9 (jul 2022), 1779–1797.

[68] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 1999. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99)*. MIT Press, Cambridge, MA, USA, 582–588.

[69] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, and Susan Frankenstein. 2015. Time series anomaly discovery with grammar-based compression. In *EDBT*.

[70] K Simonyan and A Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations (ICLR 2015)*, 1–14.

[71] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 2828–2837. https://doi.org/10.1145/3292500.3330672

[72] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. 2006. Online Outlier Detection in Sensor Data Using Non-Parametric Models. In *Proceedings of the 32nd International Conference on Very Large Data Bases* (Seoul, Korea) *(VLDB '06)*. VLDB Endowment, 187–198.

[73] Markus Thill, Wolfgang Konen, and Thomas Bäck. 2020. *MarkusThill/MGAB: The Mackey-Glass Anomaly Benchmark, https://doi.org/10.5281/zenodo.3762385*. https://doi.org/10.5281/zenodo.3762385

[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[75] Alexander von Birgelen and Oliver Niggemann. 2018. *Anomaly Detection and Localization for Cyber-Physical Production Systems with Self-Organizing Maps*. Springer Berlin Heidelberg, Berlin, Heidelberg, 55–71. https://doi.org/10.1007/978-3-662-57805-6_4

[76] Pichao Wang, Xue Wang, Hao Luo, Jingkai Zhou, Zhipeng Zhou, Fan Wang, Hao Li, and Rong Jin. 2022. Scaled relu matters for training vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 2495–2503.

[77] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time series classification from scratch with deep neural networks: A strong baseline. *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), 1578–1585.

[78] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. 2021. Early convolutions help transformers see better. *Advances in Neural Information Processing Systems* 34 (2021), 30392–30400.

[79] Yuan Yao, Abhishek Sharma, Leana Golubchik, and Ramesh Govindan. 2010. Online anomaly detection for sensor systems: A simple and efficient approach. *Performance Evaluation* 67, 11 (2010), 1059–1075. https://doi.org/10.1016/j.peva.2010.08.018 Performance 2010.

[80] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 1317–1322. https://doi.org/10.1109/ICDM.2016.0179

[81] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2018. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Mining and Knowledge Discovery* 32, 1 (Jan. 2018), 83–123. https://doi.org/10.1007/s10618-017-0519-9

[82] Yuanxiang Ying, Juanyong Duan, Chunlei Wang, Yujing Wang, Congrui Huang, and Bixiong Xu. 2020. Automated Model Selection for Time-Series Anomaly Detection. https://doi.org/10.48550/ARXIV.2009.04395

[83] Yue Zhao, Ryan Rossi, and Leman Akoglu. 2021. Automatic Unsupervised Outlier Model Selection. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 4489–4502. https://proceedings.neurips.cc/paper/2021/file/23c894276a2c5a16470e6a31f4618d73-Paper.pdf