

## Article

# Is It Worth It? Comparing Six Deep and Classical Methods for Unsupervised Anomaly Detection in Time Series

Ferdinand Rewicki <sup>1,2,\*</sup> , Joachim Denzler <sup>2</sup>  and Julia Niebling <sup>1</sup> <sup>1</sup> Institute of Data Science, German Aerospace Center, 07745 Jena, Germany<sup>2</sup> Faculty of Mathematics and Computer Science, Friedrich Schiller University, 07743 Jena, Germany

\* Correspondence: ferdinand.rewicki@dlr.de

**Abstract:** Detecting anomalies in time series data is important in a variety of fields, including system monitoring, healthcare and cybersecurity. While the abundance of available methods makes it difficult to choose the most appropriate method for a given application, each method has its strengths in detecting certain types of anomalies. In this study, we compare six unsupervised anomaly detection methods of varying complexity to determine whether more complex methods generally perform better and if certain methods are better suited to certain types of anomalies. We evaluated the methods using the UCR anomaly archive, a recent benchmark dataset for anomaly detection. We analyzed the results on a dataset and anomaly-type level after adjusting the necessary hyperparameters for each method. Additionally, we assessed the ability of each method to incorporate prior knowledge about anomalies and examined the differences between point-wise and sequence-wise features. Our experiments show that classical machine learning methods generally outperform deep learning methods across a range of anomaly types.

**Keywords:** anomaly detection; time series; machine learning; deep learning; benchmark



**Citation:** Rewicki, F.; Denzler, J.; Niebling, J. Is It Worth It? Comparing Six Deep and Classical Methods for Unsupervised Anomaly Detection in Time Series. *Appl. Sci.* **2023**, *13*, 1778. <https://doi.org/10.3390/app13031778>

Academic Editor: Markus Goldstein

Received: 20 December 2022

Revised: 19 January 2023

Accepted: 23 January 2023

Published: 30 January 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The detection of anomalies, or observations that significantly deviate from what is considered normal [1], in time series data is essential in various fields, including healthcare [2], cybersecurity [3,4], industry [5] and robotics [6]. Anomaly detection is a notoriously challenging task, as the definition of what is considered anomalous can vary based on the context or application [7]. Moreover, the absence of labeled training data for non-academic problems often precludes the use of supervised machine-learning techniques. Anomaly detection in data streams, which requires rapid results while aiming to detect anomalies accurately and efficiently, is frequently necessary. It is important to minimize false-positive detections to prevent alarm fatigue, which can result in a serious problem being overlooked due to excessive false alarms [7]. It is also necessary to choose the appropriate method based on the application and, often, domain knowledge, as the existence of a universal anomaly detection method is a myth [8]. Choosing the appropriate method from the plethora of available options can be a challenge in itself, as different methods have different strengths in detecting certain types of anomalies. The numerous available methods can be categorized using various criteria, such as the underlying probabilistic, classification, or reconstruction-based model [1], the type of input data (univariate or multivariate), the need for labeled training data, or the ability to process data streams.

In this work, we compare six unsupervised anomaly detection methods with varying complexities. Three of these methods are classical machine-learning techniques (we refer to these methods as *classical methods*) while the remaining three are based on deep learning. Our central questions in this comparison are:

1. “Is it worthwhile to sacrifice the interpretability of classical methods for potentially superior performance of deep learning methods?”

## 2. “What different types of anomalies are the methods capable of detecting?”

To address these questions, we compare the classical methods of Robust Random Cut Forest (RRCF) [9], Maximally Divergent Intervals (MDI) [10] and MERLIN [11] to the deep learning methods of Autoencoder (AE), Graph Augmented Normalizing Flows (GANF) [12], and Transformer Networks for Anomaly Detection (TranAD) [13]. We evaluate these methods on the UCR Anomaly Archive [14], a new benchmark dataset for time series anomaly detection. This archive consists of 250 univariate time series from four domains: human medicine, industry, biology and meteorology. To ensure a fair comparison, we carefully design our experimental setup and perform intensive hyperparameter tuning for applicable methods. To the best of our knowledge, this is the first work to conduct an experimental comparison of classical and deep-learning methods for anomaly detection in time series. Our key contributions are:

- We conduct a comprehensive comparison of six state-of-the-art anomaly detection methods for time series data using the UCR Anomaly Archive benchmark dataset. Our comparison is carried out in a well-defined and fair benchmark environment.
- We enhance the UCR Anomaly Archive by annotating it with 16 distinct anomaly types, providing a more nuanced and informative benchmark.
- We address two crucial questions in the field of anomaly detection: (1) whether the superior performance of deep-learning methods justifies the loss of interpretability of traditional methods and (2) the similarities and differences between the analyzed methods in terms of detecting different anomaly types.
- We examine the impact of subsequence length on the performance of the MDI and MERLIN methods, and compare point-wise to subsequence-wise features for the RRCF method.

The remainder of this paper is organized as follows: after providing an introduction to time series data and different types of anomalies in Sections 1.1 and 1.2, respectively, we present related work in Section 1.3. In Section 2.1, we present the six anomaly detection methods, followed by a description of the UCR Anomaly Archive dataset in Section 2.2 and the experimental setup in Section 2.3. The results of our experiments are presented in Section 3 and discussed in Section 4. Finally, we summarize our findings and provide an outlook on future work in Section 5.

### 1.1. Time Series Data

Time series are sequential data that are naturally ordered by time. We distinguish regular and irregular time series depending on whether or not the observations are made at equidistant intervals. We define a time series as an ordered set of observations based on [11]:

**Definition 1.** The *time series*  $\mathcal{T}$  with length  $n \in \mathbb{N}$  is defined as the set of pairs  $\mathcal{T} = \{(t_i, p_i) | t_i \leq t_{i+1}, 0 \leq i \leq n\}$  with  $p_i \in \mathbb{R}^d$  being the data points with  $d$  behavioral attributes and  $t_i \in \mathbb{N}, i \leq n$  the timestamps to which a certain data point refers. For  $d = 1$ ,  $\mathcal{T}$  is called *univariate*, and for  $d > 1$ ,  $\mathcal{T}$  is called *multivariate*.

Time series can be described using different characteristics, such as *stationarity*, which refers to a constant mean, variance and auto-correlation structure, *seasonality* describing periodically reoccurring behavior, or *sampling rate* the frequency in which observations are made [7]. For an in-depth analysis of these characteristics, we refer to [7]. As time series are usually not analyzed en bloc, we define a subsequence as a contiguous subset of the time series:

**Definition 2.** The *subsequence*  $S_{a,b} \subseteq \mathcal{X}$  of the times series  $\mathcal{X}$ , with length  $L = b - a > 0$  is given by  $S_{a,b} := \{(t_i, p_i) | 0 \leq a \leq i \leq b \leq n\}$ . For simplicity, we will often omit the indices and refer to an arbitrary subsequence as  $S$ .

## 1.2. Anomalies

There are three main types of anomalies distinguished in the literature: point anomalies, collective anomalies and contextual anomalies [1,15–18]. Point anomalies are individual data points  $(t_i, p_i) \in \mathcal{T}$  that deviate significantly from all other instances, such as a fraudulent transaction among legal finance transactions [1]. Collective anomalies refer to whole subsequences  $\tilde{S}_{a,b} \subset \mathcal{T}$  being anomalous, while the individual data points  $(t_i, p_i) \in \tilde{S}_{a,b}$  would not be considered a point anomaly [16]. For example, supraventricular premature beats in an electrocardiogram (ECG) are examples of collective anomalies. Contextual anomalies only appear anomalous depending on specific context variables. For instance, while an outside air-temperature measurement of 28 °C during August is considered normal in Panama, it would be anomalous in Antarctica. In this work, we extend this classification to 16 classes by dividing the class of collective anomalies into different subclasses, such as “frequency change” or “time shift”, which are described in Section 2.2.2.

## 1.3. Related Work

While many survey and review papers on anomaly detection are available [15,17–22] there are only a few works that have compared different methods experimentally.

Freeman et al. [7] conduct an experimental comparison of twelve anomaly detection methods, including Seasonal AutoRegressive Integrated Moving Average with exogenous variables (SARIMAX), Generalized Linear Model, Facebook Prophet [23], Matrix Profile [24], or Donut [25]. The comparison is made using a dataset compiled mainly from the Numenta benchmark [26] with a focus on different time series characteristics, such as seasonality and trend. They use the Youden Index [27] to determine a threshold for classifying anomaly scores and assess the quality of the analyzed methods using AUC ROC, Windowed-F1 and the NAB Score, which is the metric used in the Numenta benchmark.

Graabæk et al. [28] compare 15 anomaly detection methods in the context of collaborative robots. The analyzed methods are categorized as *instance-based* methods, such as k-Nearest-Neighbors and Local Outlier Factor, and *explicit generalization models*, such as Principal Component Analysis, One-Class Support Vector Machine, or Autoencoder. They compare these methods on a dataset collected from different tasks performed by a robotic arm by using AUC ROC and Area under Precision-Recall Curve as quality measures.

Ruff et al. [1] provide a comprehensive review of classical and deep-learning methods for anomaly detection. They group the presented methods into the three main classes: *Density Estimation and Probabilistic Models*, *One Class Classifications* and *Reconstruction Models*, and present various classical and deep-learning methods from each category. They also give a unifying view of the anomaly detection problem by identifying specific anomaly detection modeling components to characterize the presented methods and exemplify the modeling and evaluation process on two real-world examples.

## 2. Materials and Methods

### 2.1. Analyzed Methods

To perform our comparison, we selected three deep-learning and three classical machine-learning methods for unsupervised anomaly detection. The selection of these methods was based on various factors, including simplicity, interpretability, applicability to data streams and the existence of useful features such as the dependency graph for GANF. In the following section, we will introduce the selected methods, starting with the classical ones. One way to categorize anomaly detection methods is based on their suitability for handling data streams, which we will refer to as “online” anomaly detection, as opposed to “offline” anomaly detection carried out on data batches. A summary of the properties of the compared methods can be found in Table 1.

**Table 1.** Overview of the properties of the anomaly detection methods considered in this comparison.

	Mechanism	Class	Online/Offline	Training	Multivariate	Anomaly Score
<b>RRCF</b>	Isolation Forest	Classical	Online	✗	✓	Collusive Displacement
<b>MDI</b>	Density Estimation	Classical	Offline	✗	✓	(KL/JS) Divergence
<b>MERLIN</b>	Discord Discovery	Classical	Offline	✗	✗	Discord Distance
<b>AE</b>	Reconstruction	Deep learning	Offline training Online inference	✓	✓	Reconstruction Loss
<b>GANF</b>	Density Estimation	Deep learning	Offline training Online inference	✓	✓	Density
<b>TranAD</b>	Reconstruction	Deep learning	Offline training Online inference	✓	✓	Reconstruction Loss

### 2.1.1. Robust Random Cut Forest (RRCF)

The Robust Random Cut Forest (RRCF) [9] is a modification of the well-known Isolation Forest [29] methods, which extends the approach to data streams. Both methods work by isolating individual points from the rest of the data by recursively partitioning the dataset. This process can be represented by a binary tree structure, where each cut is represented by a pair of branches from the same node. The average path length can then be used as an anomaly score, as shorter paths indicate that a point is more likely to be anomalous [29]. One key difference between RRCF and Isolation Forest is that RRCF selects the next dimension to cut, with a probability proportional to the range of values in that dimension, rather than selecting it uniformly at random. This modification is meant to avoid cutting irrelevant dimensions and reduce the number of false positives, as well as to maintain a good recall [9]. Due to its anomaly scoring function *Collusive Displacement* RRCF is also robust to the presence of duplicates or near-duplicates which could otherwise lead to outlier masking [30]. *Displacement* refers to the classification of points as outliers, if they significantly decrease the model complexity when removed from the tree. *Collusive Displacement* accounts for duplicates or near-duplicates by removing a subset of “colluders”  $C$  alongside the point of interest  $x$ . It is defined as the expected change in the depth of points in a tree when removing a set  $C \cup \{x\}$ . For an exact definition of the *Collusive Displacement* scoring function, please refer to [9]. As RRCF works by isolating single points, one would expect its strength to lie in finding point anomalies. For the detection of anomalous subsequences, an additional preprocessing step for constructing window-based features could be considered and is analyzed in Section 3.4. We selected RRCF mainly due to its simplicity and comprehensibility.

### 2.1.2. Maximally Divergent Intervals (MDI)

Maximally Divergent Intervals (MDI) [10] is a density-based method for offline anomaly detection in multivariate, spatiotemporal data. In this work, we focus on purely temporal data and provide the definitions for this case only. For the original definitions including spatial attributes, please refer to [10]. Given a multivariate time series  $\mathcal{T}$ , MDI detects anomalous subsequences by comparing the probability density  $p_S$  of a subsequence  $S_{a,b} \subseteq \mathcal{T}$  to the density  $p_\Omega$  of the remaining part of the time series  $\Omega(S) := \mathcal{T} \setminus S_{a,b}$  for all subsequences. The distributions are modeled using *Kernel Density Estimation* or *Multivariate Gaussians*. To measure the degree of deviation  $\mathcal{D}(p_S, p_\Omega)$  between  $p_S$  and  $p_\Omega$ , an unbiased version of the *Kullback–Leibler divergence* is used. The most anomalous subsequence  $\tilde{S}$  is found by solving the underlying optimization problem [10]:

$$\tilde{S} := \arg \max_{S \subseteq \mathcal{T}} \mathcal{D}(p_S, p_{\Omega(S)})$$

MDI locates this most anomalous subsequence  $\tilde{S}$  by scanning all subsequences  $S \subseteq \mathcal{T}$  with a length between  $L_{min}$  and  $L_{max}$  and estimates the divergence  $\mathcal{D}(p_S, p_{\Omega(S)})$ , which is then used as the anomaly score. The parameters  $L_{min}$  and  $L_{max}$  need to be defined in advance. The

top- $k$  anomalous subsequences are selected by ranking the subsequences by their anomaly score and selecting the top- $k$  subsequences. To accommodate the application to large-scale data, an interval proposal technique based on Hotelling's  $T^2$  method [31] is employed, which selects interesting subsequences based on point-wise anomaly scores instead of performing full scans over the entire time series. This preselection method is motivated by the fact that most subsequences are uninteresting for detecting anomalies as these are rare by definition [10]. We selected MDI mainly due to its easily interpretable approach.

### 2.1.3. MERLIN

MERLIN [11] is a method for offline anomaly detection based on discord discovery: given a subsequence  $S$  with length  $L$  starting at timestamp  $p$ , a matching subsequence  $M$  starting at timestamp  $q$  is called a non-self match to  $S$  if  $|p - q| \geq L$  [11]. The discord  $\tilde{S}$  of a time series  $\mathcal{T}$  is defined as the subsequence with the largest distance  $d(\tilde{S}, M_{\tilde{S}})$  from its nearest non-self match  $M_{\tilde{S}}$ , where  $d(\cdot, \cdot)$  is the z-normalized (zero mean and unit variance) Euclidean distance. MERLIN is based on the discord discovery algorithm from [32]. A key factor in the success and efficiency of the algorithm is the selection of the hyperparameter  $r$ . This parameter should be chosen to be slightly less than the discord distance,  $d(\tilde{S}, M_{\tilde{S}})$ . If  $r$  is too large, the algorithm will fail, while if it is too small, the runtime will be excessively long. To address this challenge, MERLIN provides a structured search procedure for determining an appropriate value for  $r$  by leveraging the observation that good values of  $r$  for subsequences of length  $L$  are likely to be similar to good values of  $r$  for subsequences of length  $L - 1$  [11]. The maximum value of  $r$  for subsequences of length  $L$  is given by  $r_{max}(L) = 2\sqrt{L}$  [33]. To find an appropriate value for  $r$ , the algorithm begins by setting  $r = r_{max}(L_{min})$ , where  $L_{min}$  is the smallest subsequence length being considered, and then halving  $r$  until the first discord is returned. For subsequences of other lengths  $L_{min}, \dots, L_{max}$ , the previously determined values of  $r$  can be used. We selected MERLIN as it is the method provided with the UCR anomaly archive dataset, which is the benchmark dataset for our study and will be introduced in Section 2.2.

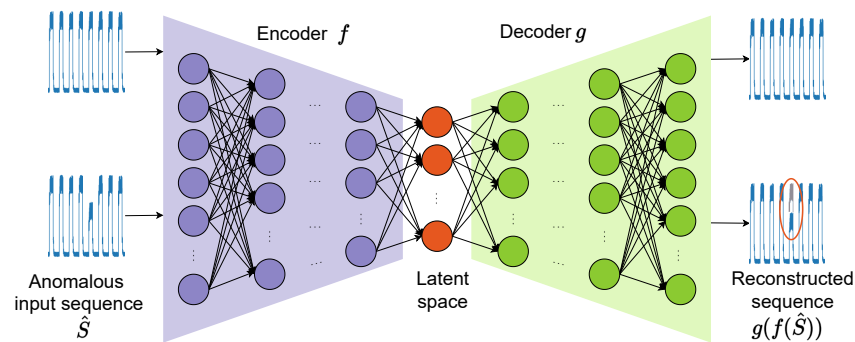
### 2.1.4. Autoencoder (AE)

Autoencoders, introduced in [34], are neural networks designed for dimensionality reduction that consists of an encoder network  $f : \mathbb{R}^n \rightarrow \mathbb{R}^l$  and a decoder network  $g : \mathbb{R}^l \rightarrow \mathbb{R}^n$ , where  $p, n \in \mathbb{N}$  and  $l < n$ . These networks are trained to reconstruct their input by learning a latent representation. The autoencoder problem can be formalized according to [35] as:

$$\arg \min_{f, g} \mathbb{E}[\Delta(x, g(f(x)))]$$

with  $x \in \mathcal{X}$  being the input data,  $\Delta$  the reconstruction loss, i.e., usually the  $L_2^2$  error function and  $\mathbb{E}[\cdot]$  the expectation over its argument [35,36]. In the context of unsupervised anomaly detection in time series data, the autoencoder learns a normal profile of the time series  $\mathcal{T}$  and detects anomalous input sequences  $\tilde{S}$  with a high reconstruction error. Figure 1 illustrates this approach.

In our experiments, we used a dense autoencoder with two hidden layers in the encoder and decoder, which have each a doubled number of neurons of the latent space and ReLU activations as non-linearities. We included the autoencoder as a basic deep-learning model in this study.



**Figure 1.** Visualization of a dense autoencoder model for time series anomaly detection, producing a high reconstruction error  $\Delta(S, g(f(\hat{S})))$  when presented with the anomalous subsequence  $\hat{S}$ . *Dense* refers to the encoder and decoder networks being fully connected.

### 2.1.5. Graph Augmented Normalizing Flows (GANF)

GANF [12] is an anomaly detection method for multivariate time series that uses normalizing flows for density estimation. Normalizing flows are generative models  $f: \mathbb{R}^d \rightarrow \mathbb{R}^d$  that utilize a series of invertible and differentiable transformations to normalize complex data distributions to “base” distributions, whose densities are typically easy to evaluate (e.g., isotropic Gaussians) [12]. In addition to modeling the density of the time series using a normalizing flow, GANF incorporates a Bayesian Network to model the causal relationships among multiple multivariate time series  $\mathcal{X} = (\mathcal{T}_1, \dots, \mathcal{T}_m)$ . Given a training set  $\mathcal{D} = \{\mathcal{X}_i\}_{i=1}^{|\mathcal{D}|}$  of multiple time series, GANF aims to learn the adjacency matrix  $A$  of the Bayesian Network and, simultaneously, the graph-augmented normalizing flow  $\mathcal{F}: (\mathcal{X}, A) \rightarrow \mathcal{Z}$ , where  $\mathcal{Z}$  is a random variable with a “simple” (base) distribution [12]. Once  $\mathcal{F}$  is learned, the estimated density  $p(\mathcal{X})$  can be evaluated to identify anomalies in low-density regions of the base distribution. The dependency encoder of the model consists of a recurrent neural network to summarize the time series up to a given time step  $t$  and a graph convolution layer to learn a dependency representation, which is then used to condition a normalizing flow  $f$ . For more information on the architectural details of GANF, please see [12]. As anomalies are rare by definition, it is typically assumed that their densities are low, and thus the estimated densities can be used as an anomaly score [12]. We included GANF as a deep-learning variant of a density-estimation-based anomaly detection method, given its ability to learn dependencies between multiple time series, although this feature is not used in the context of this comparison.

### 2.1.6. Transformer Network for Anomaly Detection (TranAD)

TranAD [13] is an anomaly detection method based on the Transformer model [37], which learns to reconstruct an input by applying several attention-based transformations. The model proposed by Tuli et al. [13] uses two-phase training. In the first phase, the model learns an approximate reconstruction of the whole time series  $\mathcal{T}$  to capture long-term trends and uses the deviation from the true time series as a *focus score*. In phase two, the focus score is used to find those subsequences where the deviation in phase one is high. Similar to other encoder–decoder models, the reconstruction loss is used as the anomaly score. We included TranAD, as it was one of the most recent publications by the time of its selection.

## 2.2. Benchmark Dataset: UCR Anomaly Archive

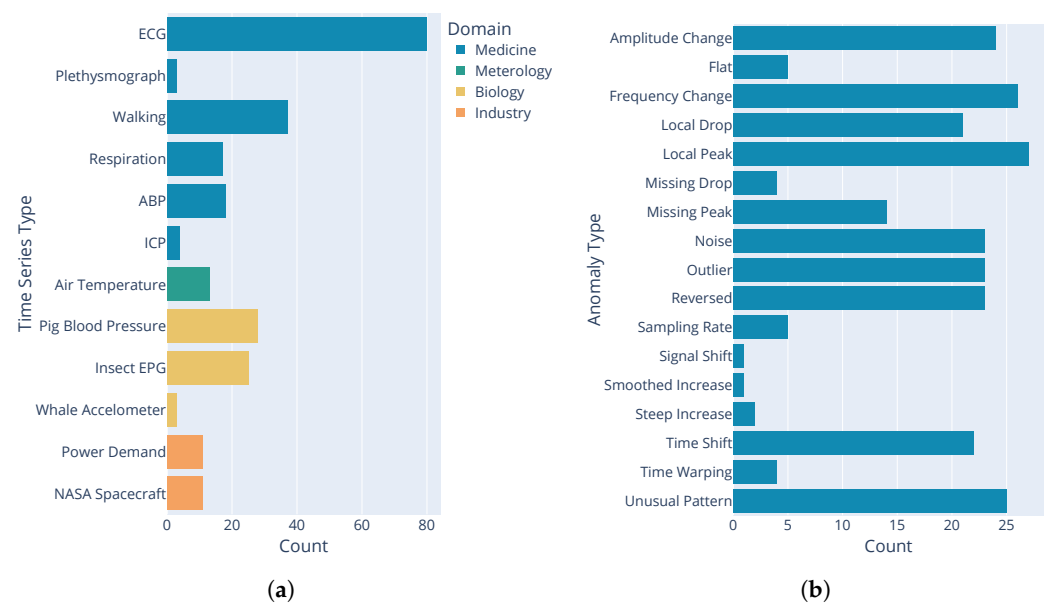
The dataset used in this study is the *UCR Anomaly Archive* [14,38], which consists of 250 univariate time series from various fields, including human medicine, biology, meteorology and industry. The time series in this dataset include both natural and artificial anomalies, with the majority being artificial. This allows for a more detailed analysis based on the type of anomaly injection. The UCR Anomaly Archive was first used in an anomaly detection contest preceding the ACM SIGKDD conference in 2021 and was published by

Wu and Keogh [14] as an alternative to commonly used benchmark datasets such as Yahoo S5 [39], Numenta [26] or NASA [40], which have been criticized for having trivial anomalies, unrealistic anomaly densities, mislabeled ground truth and being “run-to-failure biased”. This term refers to anomalies occurring at the end of a time series due to the recording being stopped after the anomaly (or *failure*) occurred.

Each time series in the UCR Anomaly Archive contains a single, sometimes subtle anomaly after a certain time stamp, with the data before that time stamp being considered normal. As we evaluate the methods described in Section 2.1 in the unsupervised setting, we do not use this label. The time series in the UCR Anomaly Archive have lengths ranging from 6674 to 900,000 data points and anomalies with lengths between 1 and 1701 data points, with a maximum anomaly pollution of 4.9% per time series.

### 2.2.1. Included Time Series

The time series in the UCR Anomaly Archive can be classified into 12 types based on the domain they originate from: human medicine, meteorology, biology and industry. Figure 2a shows the distribution of time series types in the dataset.



**Figure 2.** (a) Histogram of time series types included in the UCR Anomaly Archive. The color indicates the domain the time series originates from. (b) Histogram of anomaly types present in the UCR Anomaly Archive.

The distribution is highly imbalanced, with approximately 64% of the time series coming from human medicine applications, 22% from biology, 9% from industry and 5% being air-temperature measurements. Within a single type of time series (e.g., ECG), the time series are not unique, but differ in terms of injected anomalies or modifications to the original time series, such as the addition of Gaussian noise or baseline wander. Baseline wander, a low-frequency artifact commonly found in ECG caused by factors such as breathing or subject movement, refers to slow changes in the signal baseline [41].

### 2.2.2. Anomaly Types

To evaluate the abilities of the six anomaly detection methods to detect different types of anomalies, we annotated each time series with the type of injected anomaly. We used the supplemental material provided with the UCR Anomaly Archive dataset [42] to obtain this information. The distribution of anomaly types is shown in Figure 2b. A list of explanations and examples for each anomaly type can be found in Appendix A.1.

### 2.3. Experimental Setup

For our experiments, we implemented the benchmark pipeline described in Section 2.3.1 in Python. The anomaly detection methods were either integrated from their publicly available GitHub repositories (RRCF: <https://github.com/kLabUM/rrcf> (accessed on 18 January 2023), MDI: <https://github.com/cvjena/libmaxdiv> (accessed on 18 January 2023), GANF: <https://github.com/EnyanDai/GANF> (accessed on 18 January 2023), TranAD: <https://github.com/imperial-qore/TranAD> (accessed on 18 January 2023)) or implemented by us if a Python version was not available (MERLIN: <https://gitlab.com/dlr-dw/py-merlin> (accessed on 18 January 2023)). The Autoencoder model was implemented using the PyTorch [43] library and is available in the repository for this work (<https://gitlab.com/dlr-dw/is-it-worth-it-benchmark> (accessed on 18 January 2023)). The relevant hyperparameters for each model were tuned through 20 rounds of Bayesian optimization on 25 randomly selected time series from the UCR Anomaly Archive, using the F1 score as the optimization target. The time series used for hyperparameter tuning were excluded from the actual experiments. A table containing all hyperparameters obtained from that search can be found in Appendix A.2. All experiments were run on an Intel Xeon Platinum 8260 CPU with 10GB of allocated memory. For TranAD, we increased the memory to 20GB for the timeseries “239\_UCR\_Anomaly\_taichidbS0715Master\_190037\_593450\_593514.txt”, “240\_UCR\_Anomaly\_taichidbS0715Master\_240030\_884100\_884200.txt” and “241\_UCR\_Anomaly\_taichidbS0715Master\_250000\_837400\_839100.txt”. We ran all experiments six times: the first time, we set the random number generators of Python, Numpy and PyTorch to a fixed value (we used 42 as the seed value across all experiments) and then performed five repetitions without setting a random seed to account for random sampling effects.

#### 2.3.1. Benchmark Pipeline

To maintain a controlled experimental environment and ensure fairness among all experiments, we implemented the pipeline shown in Figure 3. The time series data were normalized to the interval  $[0, 1]$  and a sliding window approach was applied, depending on the requirements of each method. While AE, GANF and TranAD require input data to be given as subsequences with fixed length  $L$ , MDI and MERLIN require the entire time series, along with a range of subsequence lengths  $L_{min}$  and  $L_{max}$ .



**Figure 3.** Benchmark pipeline used in the experiments.

In the case of the MDI and MERLIN methods, the range of subsequence lengths used was arbitrarily set to  $L_{min} = 75$  and  $L_{max} = 125$  time steps. For TranAD, the subsequence length  $L = 10$  and stride  $s = 1$  were used, according to the experiments in [13] for the time series taken from the UCR anomaly archive. For GANF, a subsequence length of  $L = 100$  was chosen, which is in the middle of the range  $L_{min}$  and  $L_{max}$ , and stride  $s = 10$  based on [12]. As for the AE method, the subsequence length  $L = 10$  and stride  $s = 10$  were determined empirically. RRCF does not require the data to be given as subsequences. A table summarizing the configurations used in our experiments can be found in Appendix A.2.

The normalized time series or subsequences were then used as input for the respective anomaly detection method, which calculates an anomaly score. MERLIN is an exception in this regard, as it returns only the anomalous subsequences. As the scores produced by the different methods are very heterogeneous, we employed a method called *Peak Over Threshold (POT)* [44] to determine a suitable threshold for classifying subsequences as normal or anomalous. This approach was also used in previous works such as [13,40,45].



### 2.3.2. Anomaly Score Classification

To ensure a fair comparison of the results produced using the six anomaly detection methods, we use the principle of Extreme Value Theory (EVT) to determine a threshold for classifying subsequences as anomalous or normal [44]. EVT is an approach used to find the law of extreme values, which are often located in the tails of a probability distribution, without making any assumptions about the data distribution [46]. The *Peaks-Over-Threshold (POT)* method [44], which is the second theorem in EVT, fits the tail of a probability distribution with a Generalized Pareto Distribution (GPD). In the context of anomaly score classification, the Peak-Over-Threshold (POT) method is utilized to learn an appropriate threshold for the anomaly scores  $\{\sigma_1, \sigma_2, \dots, \sigma_n\}$  [46]. Specifically, the Generalized Pareto Distribution (GPD) is adapted to focus on values at the low ends of the distribution. According to [46], a modified version of POT for anomaly score classification is defined as follows: Given a random variable  $S$  that models the anomaly scores  $\{\sigma_1, \sigma_2, \dots, \sigma_n\}$  and an initial threshold  $th^0$ , the cumulative distribution function of the GPD is adapted to:

$$\tilde{F}(s) = \mathbb{P}(th^0 - S > s | S < th^0) \sim (1 + \frac{\gamma s}{\beta})^{-\frac{1}{\gamma}}, \quad (1)$$

where  $\beta$  and  $\gamma$  are the scale and shape parameters of the GPD. The threshold  $th$  is then computed by:

$$th \simeq th^0 - \frac{\hat{\beta}}{\hat{\gamma}} \left( \left( \frac{qn}{n_{th^0}} \right)^{-\hat{\gamma}} - 1 \right), \quad (2)$$

where  $\hat{\beta}$  and  $\hat{\gamma}$  are the maximum likelihood estimates of the scale and shape parameters in Equation (1) estimated from  $\{\sigma_1, \sigma_2, \dots, \sigma_n\}$ ,  $q$  is the preferred probability to observe an anomaly score below the initial threshold  $S < th^0$  and  $n_{th^0}$  is the number of anomaly scores below the initial threshold  $|\{\sigma_i | \sigma_i < th^0\}|$ . The anomaly label  $y_i$  for a predicted subsequence  $\hat{S}_{i,i+w}$  of length  $w$  is obtained by:

$$y_i = \mathbb{1}(\sigma_i \geq th),$$

where  $\sigma_i$  is the anomaly score for the subsequence  $\hat{S}_{i,i+L}$ ,  $th$  is the threshold from Equation (2), and  $\mathbb{1}(\cdot)$  is the indicator function. For further information on POT, interested readers may refer to [44,46]. In the experiments, the Streaming POT variant from [44] is used, with POT being initialized on the first 10% of the anomaly scores and the parameter  $q$  set to 0.01 empirically.

We do not perform this step for the MERLIN method because it already returns binary labels per subsequence. Since it is typically acceptable for an algorithm to detect any point in an anomalous subsequence as long as the delay is not too long, we adopt the method proposed in [25] and subsequently used in [13,47] for adjusting the predicted anomalous labels to account for varying subsequence lengths. If a point in a true anomalous segment can be detected by the derived score and threshold, we count this segment as correctly detected from that point forward and treat all points within the segment as if they could be detected by the threshold.

### 2.3.3. Quality Measures

To evaluate the performance of the anomaly detection methods, we use the area under the receiver operating characteristic curve (AUC ROC), F1 Score and UCR score. The UCR score (the scoring function is not named in [42], so we call it the UCR score) is the recommended metric provided with the UCR Anomaly Archive. To calculate the AUC ROC and F1 Score, we scale the anomaly scores for the subsequences back to the length of the subsequence and calculate point-wise metrics.

### AUC ROC

The AUC ROC is a measure of the ability of a binary classifier to separate two classes and can be seen as a single-number summary of a ROC plot [48]. In a ROC plot, the true-positive rate is plotted against the false-positive rate at increasing threshold levels for thresholding the output of the classifier. The higher the AUC ROC, the more easily the classifier can separate the two classes. A perfect classifier achieves a score of one by ranking all examples of the positive class higher than all examples of the negative class. Therefore, we use the AUC ROC as a measure of the quality of the produced anomaly scores: a high score indicates good separability between normal and anomalous points or subsequences. We are aware that the AUC ROC is not a suitable measure for unbalanced problems such as anomaly detection, where the anomalous class is small by definition, but we report it due to its widespread use in the literature.

### F1 Score

The F1-Score is the harmonic mean of precision and recall and is defined as

$$F1 = 2 \cdot \frac{\textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

where TP, FP and FN are the true-positive, false-positive and false-negative detections, and precision and recall are defined as:

$$\textit{precision} = \frac{TP}{TP + FP}, \textit{recall} = \frac{TP}{TP + FN}.$$

Since the F1 Score is calculated based on the result of the binary classification, it provides evidence about the quality of the threshold used. If a method has a high AUC ROC but a low F1 Score, this would indicate a poor threshold.

### UCR Score

The UCR score is the recommended metric provided with the UCR Anomaly Archive and is a binary score indicating whether or not a method was able to find the single anomaly in a time series. It is defined as:

$$UCR_{\text{score}} := \mathbb{1}(\min(a - L_{\tilde{s}}, a - 100) < t^* < \max(b + L_{\tilde{s}}, b + 100)) \quad (3)$$

where  $a$  and  $b$  are the beginning and end of the true anomaly with length  $L_{\tilde{s}}$ ,  $t^*$  is the timestamp of the point with the highest anomaly score, and  $\mathbb{1}(\cdot)$  is the indicator function. For subsequences  $S_{i,j}$ , we use the middle point  $t^* = i + \lfloor \frac{j-i}{2} \rfloor$ . The tolerance of 100 time steps is added to account for very short anomalies [42].

Being a binary measure, the UCR score tells us whether or not the single anomaly in a certain time series was detected by having the highest anomaly score. A UCR score of zero does not convey any information about whether the anomaly was found, but a false-positive result has a higher anomaly score or was not detected at all. As this might not be necessary for a situation such as a challenge, where only positive results matter, it is essential to consider other metrics such as the F1 score alongside visual inspection to correctly interpret the results. If a method shows a UCR score of one but a low F1 score at the same time, this indicates the detection of the true anomaly with the highest anomaly score; otherwise, the UCR score would be zero. A low F1 score, however, can be either caused by the presence of false-positive or false-negative results. It may also be subject to the detection of a short anomaly within the 100 time steps tolerance interval considered in Equation (3). Therefore, evaluating the F1 score alone is not sufficient. On the other hand, if a method shows a UCR score of 0 but a high F1 score, this indicates that the anomaly was identified without many false positives or false negatives, but that the anomaly score for the true anomaly ranked lower than for false detections. The reasons for false positives or

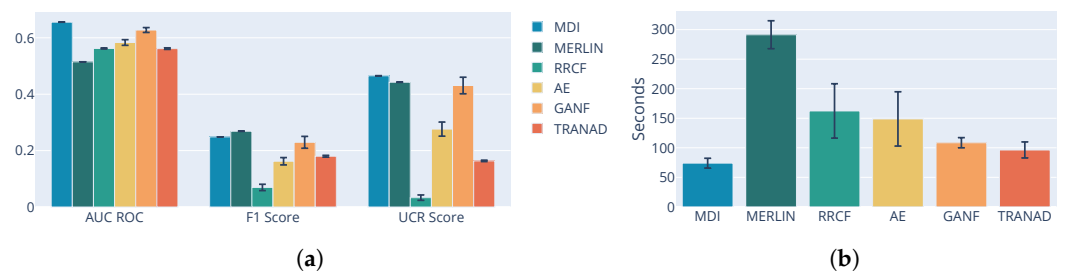
false negatives can be manifold and will be discussed in more detail at the beginning of Section 4. When interpreting aggregated UCR scores, an averaged UCR score of 0.5 means that the true anomaly was successfully identified as having the highest anomaly score in half of the analyzed time series.

### 3. Results

We analyze the six anomaly detection methods regarding their overall performance in Section 3.1 and their differences in detecting certain types of anomalies in Section 3.2. Beyond that, we analyze the influence of varying subsequence lengths on MDI and MERLIN, and thus, their ability to utilize additional information about the anomalies in Section 3.3 and compare the point-wise application of RRCF to the raw time series to that on subsequence-wise statistical vectors in Section 3.4.

#### 3.1. Performance Analysis by Method

We evaluate the performance of six anomaly-detection methods using three metrics: macro-averaged AUC ROC, F1 score and UCR score, as well as the average runtime for a single time series. The results are visualised in Figure 4. Of the methods compared, MDI achieves the highest AUC ROC and UCR scores, while MERLIN performs better in terms of F1 score. Among the deep-learning methods, GANF has the highest scores across all three metrics. F1- and UCR scores are 4% lower compared to the best-performing classical method, and the AUC ROC of 0.66 is the second best. AE scores are higher than TranAD for the AUC ROC and UCR Score, while TranAD shows a slightly higher F1 score. RRCF performs poorly, failing to detect a notable amount of anomalies in the test set and demonstrating the lowest F1 and UCR scores. The numerical results are shown in Table 2. The scores in Table 2 are generally low across all methods, likely due to the test set time series producing low or zero scores. We discuss the implications of various combinations of high and low scores, as well as their potential causes, in Section 4. MDI and MERLIN, being deterministic methods, are not subject to sampling effects and therefore have a standard deviation of 0.0 among the six repetitions of the experiment.



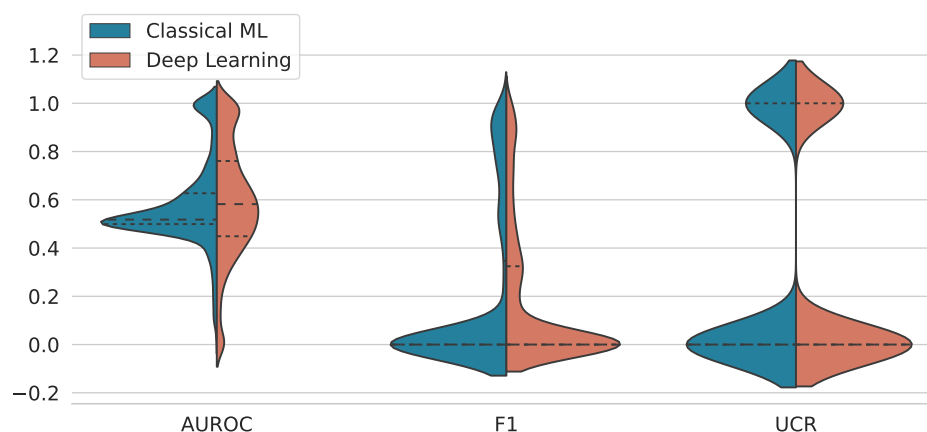
**Figure 4.** (a) Macro-averaged performance metrics for each method. The error bars indicate the standard deviation caused by random effects over six repetitions of the experiment. (b) Average runtime per time series. For the deep-learning methods, training time is included. The error bars indicate the standard deviation over six repetitions of the experiment.

**Table 2.** Performance comparison of six anomaly detection methods on macro-averaged AUC ROC, F1 score, UCR score and runtime for a single time series. Results are grouped by model class (classical ML and deep learning) and presented as mean  $\pm$  standard deviation over six repetitions. The value in each second column denotes the mean aggregated by method class.

Class	Method	AUC ROC		F1 Score		UCR Score		Runtime (s)
Classical ML	MDI	<b>0.66 <math>\pm</math> 0.0</b>		0.25 $\pm$ 0.0		<b>0.47 <math>\pm</math> 0.0</b>		<b>74</b>
	MERLIN	0.51 $\pm$ 0.0	0.58 $\pm$ 0.0006	<b>0.27 <math>\pm</math> 0.0</b>	<b>0.20 <math>\pm</math> 0.004</b>	0.44 $\pm$ 0.0	<b>0.31 <math>\pm</math> 0.0033</b>	291
	RRCF	0.56 $\pm$ 0.0019		0.07 $\pm$ 0.011		0.03 $\pm$ 0.0094		162
Deep Learning	AE	0.58 $\pm$ 0.01		0.16 $\pm$ 0.013		0.28 $\pm$ 0.025		149
	TranAD	0.56 $\pm$ 0.003	<b>0.59 <math>\pm</math> 0.002</b>	0.18 $\pm$ 0.003	0.19 $\pm$ 0.009	0.16 $\pm$ 0.003	0.29 $\pm$ 0.007	109
	GANF	0.63 $\pm$ 0.009		0.23 $\pm$ 0.021		0.43 $\pm$ 0.03		96

In terms of average runtime for a single time series from the UCR anomaly Archive, MDI performs best, with a runtime of 74 s. TranAD is about 22 s slower on average (96 s) and GANF has an average runtime of 109 s. AE has a runtime that is twice as long as that of MDI, while RRCF (162 s) has a slightly longer runtime, but both fall below 200 s. MERLIN has the worst runtime of 291 s, which is almost four times that of MDI. It is worth noting that these runtimes may be influenced by the specific implementations used. MDI is implemented in C++ with a Python interface, while the other methods are purely implemented in Python. For the deep learning methods AE, GANF and TranAD, the training time is included in the reported runtime.

To target the main question addressed in this paper, we aggregated the results by model class and visualized them using violin plots in Figure 5. The violin plots show the kernel density estimates for the two classes: “Classical Machine Learning Methods” (containing MDI, MERLIN and RRCF) and “Deep Learning Methods” (containing AE, GANF and TranAD). All density curves have two peaks: one around 0.5 for AUC ROC and 0.0 for F1 and UCR scores, and a smaller one around 0.9 (F1 score) and 1 (AUC ROC and UCR score). The peaks around 0.5 and 0 represent those results where the methods failed to detect anomalies, while the peaks around 0.9 and 1 mark successful anomaly detection. For the F1- and UCR scores, the area under the peaks at 0.9 and 1 is larger for the classical ML methods than for the deep-learning methods, indicating more successful anomaly detection for the “Classical ML” class. Conversely, the area under the peaks at 0.5 and 0 is larger for the deep-learning methods.



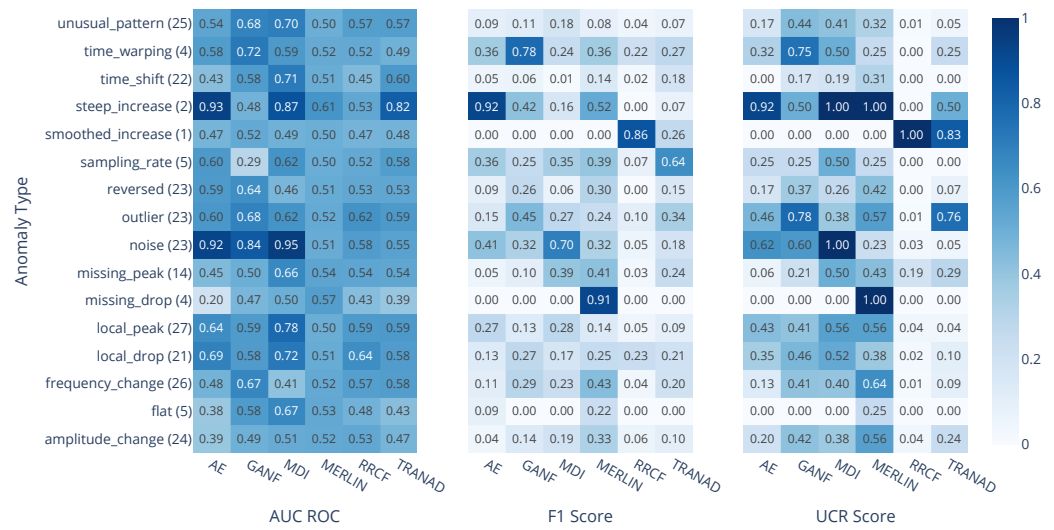
**Figure 5.** Distribution of results for the AUC ROC, F1 Score and UCR score aggregated by model class. The dashed lines encode the quartiles of the distributions. The area under the peaks around 1 is larger for the Classical ML methods for F1- and UCR score.

### 3.2. Performance Analysis by Anomaly Type

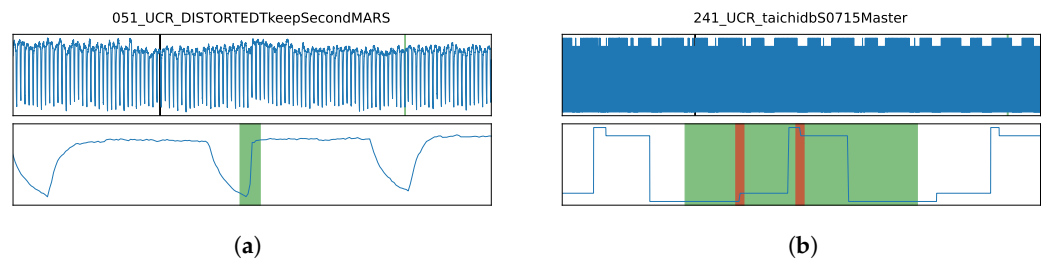
The second area of interest in this study is the differences between the analyzed methods to detect certain types of anomalies. Therefore, we aggregated our results based on the 16 anomaly classes described in Section 2.2.2. The results are shown in Figure 6.

The anomaly that was detected by all methods except RRCF is the “steep increase” anomaly shown in Figure 7a. This anomaly can be found in two time series of the UCR Anomaly Archive which represent the same data but were distorted in one case, which is shown in Figure 7a. According to the UCR score, MDI and MERLIN detect this anomaly in every repetition of the experiment with the highest UCR score. As both methods are deterministic, it is expected that the results do not differ between multiple runs. AE finds this anomaly in 11/12 cases. GANF and TranAD detect the “steep\_increase” anomaly only in the undistorted version of the time series. RRCF and TranAD detect the “smoothed increase” anomaly shown in Figure 7b where a normally steep increase was smoothed by increasing the number of different values in one cycle. For RRCF, this is also the only type this method can find. While RRCF has a UCR score below 0.05 for 14/16 anomaly types, it

detects the “smoothed increase” anomaly with a UCR score of 1.0 and an F1 score of 0.86. TranAD finds this anomaly in 5/6 cases.



**Figure 6.** The heatmaps show the macro-averaged AUC ROC, F1 and UCR scores for the 16 annotated anomaly types over six repetitions of the experiment. Next to the anomaly type, the number of times series containing that type is shown in parenthesis. The standard deviations resulting from random effects can be found in Appendix A.3.



**Figure 7.** Time Series containing “step increase” (a) and “smoothed increase” (b) anomalies. (a) Overview (top) and detail (bottom) plot of the distorted time series containing the “step increase” anomaly. The ground truth anomaly is highlighted in green. (b) Overview (top) and detail (bottom) plot of the time series containing the “smoothed increase” anomaly. The ground truth anomaly is highlighted in green, while the sections where the smoothing is visible are highlighted in red.

For the remaining anomaly types, the results are more diverse. GANF, MERLIN and TranAD find the majority of the 23 “outlier” anomalies, with GANF and TranAD performing better than MERLIN on this type. The 23 “noise” anomalies, however, are detected by AE, GANF and MDI, with MDI finding every single one with the highest anomaly score.

From a method point-of-view, MDI achieves UCR scores above or equal to 0.5 for the classes “time warping”, “step increase”, “sampling rate”, “noise”, “missing peak”, “local peak” and “local drop”. For the “noise”-type anomalies, the F1 score is above 0.5 as well. The results for AE tend to look similar to those of MDI, but the scores for AE are mostly a few points lower; therefore, AE has a UCR score above 0.5 only for “noise” and “step increase”, with the latter having an F1 Score of 0.92. MERLIN shows a UCR score above or equal to 0.5 for “step increase”, “outlier”, “missing drop”, “local peak”, “frequency change” and “amplitude change” anomalies, making “step increase” and “missing peak” the only classes in which both methods have a UCR score above or equal 0.5. In terms of F1 Score, MERLIN scores above 0.5 for the classes “noise” and “step increase”. GANF is the best-performing method regarding the anomaly types “time warping” and “outlier”

with UCR scores of 0.75 and 0.78 and F1 scores of 0.78 and 0.45, respectively. Additionally, GANF detects at least half of the anomalies with the type “steep increase” and “noise”. TranAD achieves a UCR score above or equal to 0.5 for the classes “outlier”, “smoothed increase” and “steep increase”, but for the latter, the corresponding low F1 Score indicates a high number of false positives. For the “sampling\_rate” anomalies, the opposite is true, as the F1 Score is 0.61 here, but the UCR score is 0.0.

We will discuss these differences in Section 4 in more detail. TMDI and MERLIN together detect the anomalies of more than two-thirds of the annotated anomaly types. For the classes “flat”, “reversed”, “time shift” and “unusual pattern”, no method achieved a UCR score above or equal to 0.5.

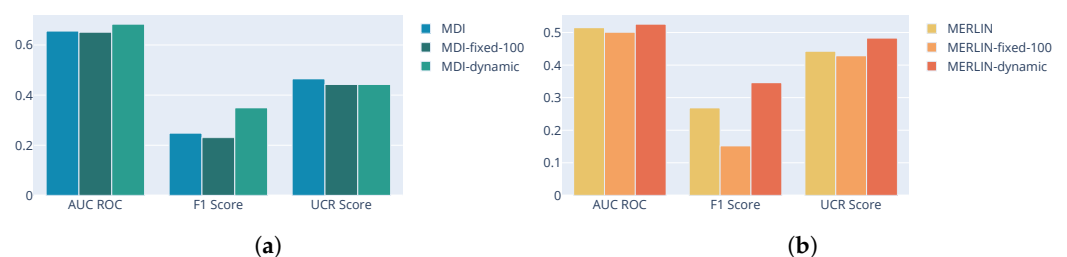
### 3.3. The Influence of Subsequence Length on MDI and MERLIN

The goal of this experiment was to examine the influence of the subsequence length on the results for MDI and MERLIN and to evaluate their ability to utilize additional information about the problem domain given with the range of subsequence lengths. To make a fair comparison, we fixed the subsequence length range for MDI and MERLIN to  $L_{min} = 75$  and  $L_{max} = 125$  time steps in the results presented in Sections 3.1 and 3.2, regardless of the specific characteristics of the individual time series, such as cycle length or expected length of the anomaly.

We therefore compare the baseline results from Section 3.1 with two strategies for setting the subsequence range. For the “dynamic” strategy, we provided additional information by setting the range of subsequence lengths based on the length  $L_{\tilde{s}_{a,b}} = b - a$  of the true anomaly  $\tilde{s}_{a,b}$  to  $L_{\tilde{s}_{a,b}} \pm 25\%$ . For the “fixed” strategy, we chose a fixed length of 100 timesteps, thereby reducing the given information compared to the baseline.

The results for MDI, shown in Figure 8a, demonstrate that fixing the subsequence length to 100 and reducing the given information leads to a decrease in the AUC ROC and F1 score. In contrast, choosing the range for the subsequence length dynamically leads to an increase in the AUC ROC and F1 score. The results for the UCR score do not reflect this trend; the highest UCR score is still achieved with the baseline configuration, but the difference between the fixed and dynamically chosen subsequence length is relatively small.

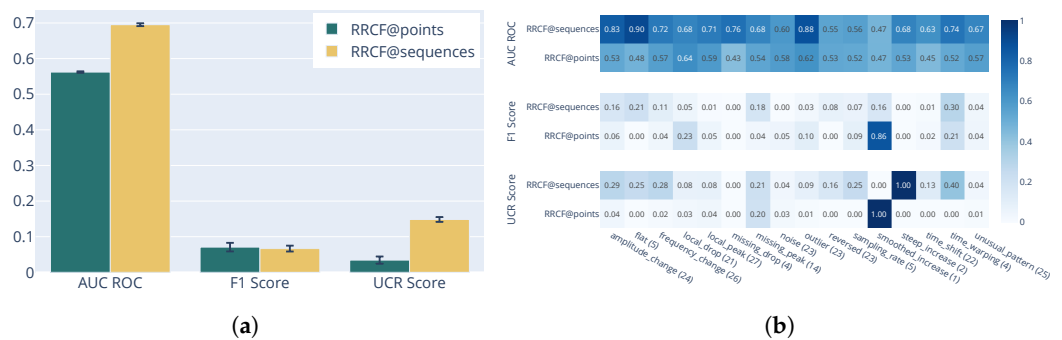
The results for MERLIN, displayed in Figure 8b, show similar behavior, but the differences between the strategies are more pronounced. For MERLIN, the positive effect of additional information is present across all three metrics.



**Figure 8.** Results for different subsequence lengths for MDI (a) and MERLIN (b). (a) The baseline results “MDI” are those from the main experiments using a subsequence length range of  $L_{min} = 75$  and  $L_{max} = 125$ . The results labeled “MDI\_fixed\_100” were obtained using a fixed subsequence length of 100 time steps, while the results labeled “MDI\_dynamic” were obtained by individually setting  $L_{min}$  and  $L_{max}$  to 75% and 125% of the true anomaly length, respectively, for each time series. (b) The baseline results “MERLIN” are those from the main experiments using a subsequence length range of  $L_{min} = 75$  and  $L_{max} = 125$ . The results labeled “MERLIN\_fixed\_100” were obtained using a fixed subsequence length of 100 time steps, while the results labeled “MERLIN\_dynamic” were obtained by individually setting  $L_{min}$  and  $L_{max}$  to 75% and 125% of the true anomaly length, respectively, for each time series.

### 3.4. RRCF on Sliding Window Statistics

RRCF applied to point-wise features is tailored towards finding point anomalies due to its principle of isolating single points. In this experiment, we compare the baseline RRCF we used in the former experiments (RRCF@points) to an alternative (RRCF@sequences) where we preprocess the time series by computing a vector consisting of the minimum, maximum coefficient of variation and the first four moments (mean, variance, skewness and kurtosis) of a sliding window. We choose a subsequence length of 100 and a stride of 50. We also tuned the hyper-parameters  $n\_trees$  and  $tree\_size$  as described in Section 2.3. The results are shown in Figure 9. Using subsequence-wise features for RRCF increased the AUC ROC from 0.56 to 0.7, making this the best AUC ROC result among the analyzed methods. Additionally, the UCR score increased for RRCF@sequences by a factor of 5 from 0.03 to 0.15. The F1 score did not change substantially. While RRCF applied to point-wise features was the only method detecting the “smoothed increase” anomaly, this anomaly is not detected any more. Instead, RRCF applied to subsequence-based features detected the ‘step increase’ anomalies, like the other five methods. For all other anomaly types except “midding drop”, the UCR score increased for RRCF@sequences. The highest increase is made for the “time warping” anomaly from 0.0 to 0.4.



**Figure 9.** Comparing AUC ROC, F1- and UCR score of the baseline RRCF using the raw time series as input (RRCF@points) to RRCF applied on sliding window statistics (RRCF@sequences). (a) Macro-averaged performance metrics for each method. The error bars indicate the standard deviation caused by random effects over six repetitions of the experiment. (b) Macro-averaged AUC ROC, F1 and UCR scores for the 16 annotated anomaly types over six repetitions of the experiment. Next to the anomaly type, the number of timeseries containing that type is shown in parenthesis.

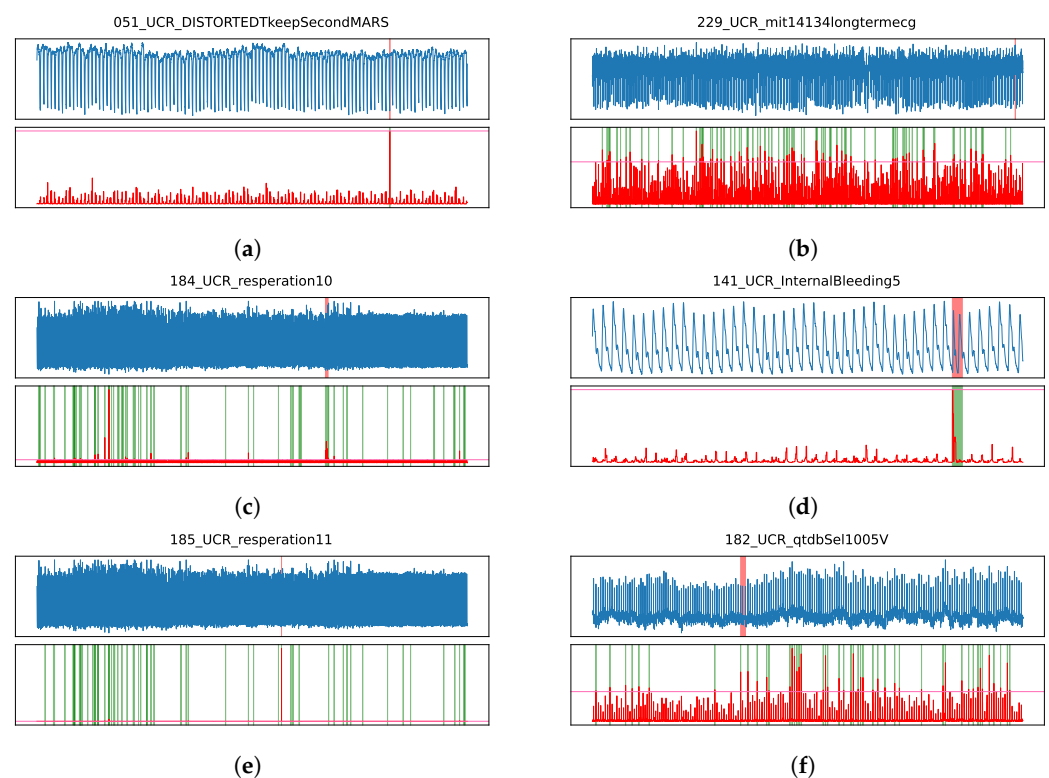
## 4. Discussion

Before discussing the results obtained for individual methods, it is necessary to explain how to interpret the various metrics and their combinations. The low macro-averaged scores across all methods shown in Table 2 can have different causes. To understand these, we will build upon the discussion of the importance of jointly analyzing different metrics, given in Section 2.3.3, and focus on the various reasons for false-positive or false-negative results in the following. For instance, a low F1 score may be due to an insufficient anomaly score, which prevents the detection of the true anomaly, or it may be due to a poor choice of threshold, leading to an increase in false positives.

Figure 10 illustrates this using different results for the autoencoder model. A high F1 score and a UCR score of one at the same time indicate the successful detection of the true anomaly without any—or with very few—false-positive results, depending on the value of the F1 score, as shown in Figure 10a,d. On the other hand, a low F1 score and a UCR score of 0, as shown in Figure 10b,f, indicate that the anomaly was not detected due to an insufficient anomaly score. In this case, a high AUC ROC value may indicate that the anomaly could have been detected with a low anomaly score, but the threshold was set too high, resulting in the subsequence not being classified as anomalous. Figure 10b,f also demonstrate that AUC ROC is generally not a suitable measure to assess the quality of

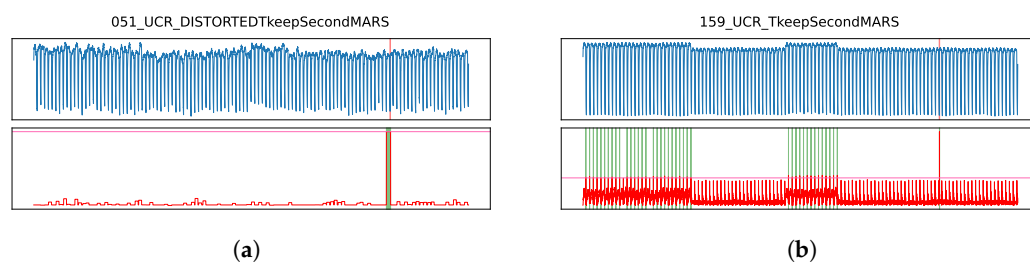
results in highly unbalanced problems in a meaningful way. In both cases, the anomaly score is not suitable for detecting the true anomaly.

A high F1 score but a UCR score of 0, as shown in Figure 10c, indicates that the true anomaly was detected, but a false-positive result has a higher anomaly score. A UCR score of one but a low F1 score signifies the correct detection of the true anomaly with the highest anomaly score, but false-positive or false-negative results lead to a low F1 score. Figures 11 and 12 illustrate the different reasons for this situation, which can be caused by a poor threshold value, as shown in Figure 11b; or the detected anomaly's subsequence length being much longer, as in Figure 11a, or shorter than the ground truth label, as in Figure 12b. This leads to increased false-positive or false-negative results, respectively. A fourth case with this result occurs from the detection of a short anomaly within the 100 time steps tolerance, which is considered in the definition of the UCR score in Equation (3). This is shown in Figure 12a.

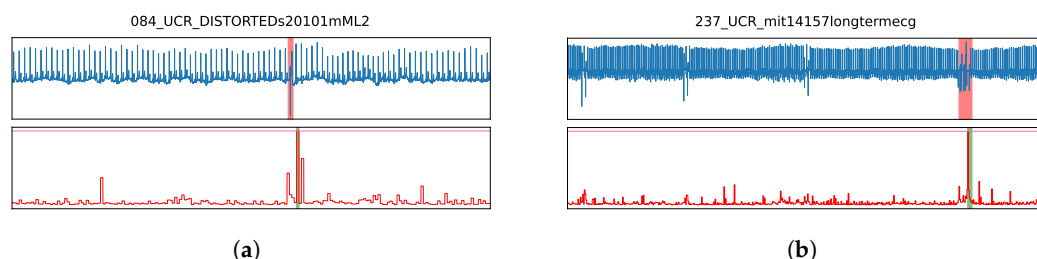


**Figure 10.** Interpretation of results shown on examples of the autoencoder model. The top figure represents the time series with the true anomaly marked in red, while the bottom figure shows the anomaly score, with the pink horizontal line being the determined threshold and the predicted anomalies highlighted in green. (a) High AUC ROC, F1- and UCR scores indicate correct detections without or with very few false positives. (b) High AUC ROC but low F1- and UCR score indicate an insufficient anomaly score and/or an insufficient threshold. (c) A high F1- but a low UCR score indicates the detection of the true anomaly but a false positive has a higher anomaly score. (d) Low AUC ROC but high F1- and UCR score indicate a correct detection of the labeled anomaly without or with few false positives but the detected subsequence is shorter than the true anomaly. (e) High UCR- but low F1 score indicates the correct detection of the true anomaly by the highest anomaly score but a bad threshold value leading to an increased number of false positives. (f) Low AUC ROC, F1- and UCR scores indicate an insufficient anomaly score not detecting the true anomaly.





**Figure 11.** Two different reasons for low F1 scores—(a) shows a result from MDI, (b) shows a result from TranAD. (a) Low F1 score caused by false positive results due to the length of the correct detected anomaly. (b) Low F1 score caused by false positive results due to a poor threshold.



**Figure 12.** Two different reasons for low F1 scores. Both results are from RRCF. (a) The anomaly is detected slightly after the ground truth label but within the 100 time steps tolerance for short anomalies in the UCR score. (b) The detected anomaly is shorter than the ground truth label causing false negatives in the computation of the F1 score.

The best results in terms of F1 score and UCR score are obtained by MDI and MERLIN, with MERLIN having a slightly higher F1 score and MDI scoring slightly higher in terms of UCR score. The differences between these two methods are around 0.02. This difference in F1 score is likely due to the different methods used to choose the threshold. Both methods return a score only for the detected anomalous sequences, but MDI either requires the number of anomalies to be returned or may return a score for up to every subsequence. In order to not give MDI an advantage over its competitors, the latter option was chosen and a threshold was determined using the POT method instead of the minimum anomaly score, as was the case for MERLIN. This makes MDI more prone to detecting false positives in the described setup compared to MERLIN, which only returns the subsequences that have been detected as anomalous. RRCF performs poorly in terms of F1 score and UCR score, which may be due to its mechanism for isolating single points and its focus on point anomalies.

Among the deep-learning methods, GANF demonstrates the best performance. GANF achieves the highest scores for all three metrics and detects the largest variety of anomaly types. The results for AE and TranAD are mixed. AE has a higher AUC ROC and UCR score, but it only detects two different anomaly types with a UCR score above 0.5. TranAD, in contrast, has a slightly higher F1 score when compared to AE and detects three different anomaly types.

Despite the low results for RRCF, the classical machine-learning methods show superior performance compared to the deep-learning methods when aggregating the results by method class, as shown in Figure 5. This difference is particularly notable in the F1 score. One assumption might be that these methods perform better in unsupervised settings that do not require a training phase. However, this is contradicted by the RRCF results.

In terms of runtime, MERLIN has the longest average processing time of 291 s per time series. This runtime is mainly determined by the discord discovery algorithm that is called for every subsequence length  $L_i \in \{L_{min}, \dots, L_{max}\}$ . The complexity of this algorithm is quadratic with respect to the size of the set of potential discords, which is determined in the candidate selection phase. However, for small candidate sets produced by a “good” choice for the parameter  $r$  [11], the complexity becomes effectively linear. As MERLIN starts with the highest possible value for  $r$  and decreases it, it is unlikely to encounter a case in which a small  $r$  value causes the candidate subset to become too large. On the other hand,

MDI uses a subsequence proposal technique based on Hotelling's  $T^2$  method [31], which selects interesting subsequences based on point anomaly scores rather than performing full scans of the data. In addition to these differences in candidate subsequence selection, the specific implementations of the algorithms also have a significant impact on their runtime. While we implemented MERLIN purely in Python, MDI is implemented in C++ with a Python interface.

All methods except RRFCF were able to detect the "steep increase" anomaly shown in Figure 7a, with varying UCR scores ranging from 0.5 for GANF and TranAD to 1.0 for MDI and MERLIN. The low F1-Score for MDI indicates a high number of false-positive results, which in this case depends on the length of the detected subsequence, as shown in Figure 11a. In contrast, the low F1 score for TranAD is caused by a poor threshold, leading to an increased number of false-positive results, as shown in Figure 11b.

RRFCF is unique in its ability to detect the "smoothed increase" anomaly shown in Figure 7b, but not the "steep increase" anomaly detected by the other five methods. This behavior can be explained by the working principle of RRFCF to isolate single points. The values in the smoothed subsequence occur only once in the time series and can therefore be isolated from all other values. That RRFCF does not find the outlier anomalies seems contradicting, but this is due to the time series containing other extreme values, e.g., with an inverted sign, covering the true anomaly in the anomaly score.

The comparison of different strategies for choosing the range of subsequence lengths for MDI and MERLIN presented in Section 3.3 reveals that both methods can utilize additional information about the anomalies, with a stronger effect for MERLIN. Providing additional information in terms of the subsequence length of the true anomaly increased the F1 score for MDI but decreased the UCR score by 0.02, which indicates that MDI utilized the additional information to reduce false-positive results. For MERLIN, the F1 score and UCR score increased, indicating that the information on the true anomaly length helped MERLIN to identify anomalies it missed before.

In the final experiment, we used subsequence-based statistics instead of point-wise features for RRFCF, which increased the UCR score and the AUC ROC. However, the macro-averaged F1 score slightly decreased due to its inability to detect the "smoothed increase" anomaly. Instead, RRFCF@sequences was able to detect the "steep increase" anomaly like the other five methods, indicating that this anomaly can only be detected on the subsequence level. The low F1 scores for RRFCF@sequences on those time series with a UCR score of one are mostly caused by the anomaly being detected slightly before or after the ground truth label but within the 100 time steps tolerance for short anomalies, or by the true anomaly being much shorter or longer than the subsequence length used for RRFCF@sequences. Figure 12 illustrates these two cases.

We conclude this section by summarizing the strengths and weaknesses of the methods analyzed in this study. MDI and MERLIN have the notable advantage of not requiring any hyperparameter tuning. The only parameters that need to be set are the minimal and maximal subsequence lengths, which practitioners select based on the specific application or domain. Despite the arbitrary choice of  $L_{min} = 75$  and  $L_{max} = 125$  time steps, MDI and MERLIN still outperform all other methods in this study. Additionally, these methods detect a wide range of anomaly types. However, a disadvantage of MDI and MERLIN is that they are not immediately applicable in an online setting. Although discord discovery can be performed online using a different algorithm such as DAMP [49], MERLIN cannot be directly applied to data streams. Similarly, while it may be possible to adapt MDI to consider only subsequences up to a given timestamp when estimating the density of  $\Omega(S)$ , the current version of MDI does not support this.

The isolation forest approach used in RRFCF is intuitive and can be applied to data streams; these factors are advantages of RRFCF. However, RRFCF shows poor results in this study and may be more suitable for applications in which outliers have distinct values from normal data. All three classical methods have the advantage of being easily interpretable.

GANF is the best-performing deep learning-based method in this study, which suggests that density-estimation-based methods are effective when it comes to detecting anomalous sequences. Additionally, GANF is capable of being applied online once trained and has the potential to learn the dependency graph of multiple time series, which, although not analyzed in this study, could be beneficial in specific applications. A major disadvantage of GANF is that it requires values to be selected for numerous hyperparameters. In our experiments, we used Bayesian Optimization to determine suitable values for the three most important hyperparameters (latent space dimension, learning rate and number of blocks), as identified by [12], using 10% of the time series in the UCR Anomaly Archive. We used the default values from [12] for the remaining eight hyperparameters, as they had not been tuned in that study either. Using default values from [12] for the three tuned hyperparameters leads to a decrease in the F1 score of 2–5% and a drop in the UCR score of up to 19%. However, averaging the two sets of hyperparameter values used in [12] increases the UCR score by approximately 6%. These better hyperparameters were not identified during the hyperparameter search, which highlights the general disadvantage of methods with a high number of hyperparameters.

The results for AE and TranAD are inconclusive, but are generally worse when compared to GANF. However, they are not as poor as the results for RRFCF. Additionally, these two methods also have the disadvantage of having various hyperparameters that need to be set. When using the default parameters from [13] for TranAD, the results for F1- and UCR score decrease by about 4–2%, depending on which set is used. The values mentioned in the paper differ from those used in the repository. For AE, we do not have a set of default parameters, but we observed comparable or slightly worse results when choosing an arbitrary set of parameters. Like GANF, both methods have the advantage of being able to be applied to data streams after training.

## 5. Conclusions

In this study, we compared six anomaly detection methods, three of which were classical machine-learning methods and three of which were based on deep learning. We conducted extensive experiments on the UCR Anomaly Archive benchmark dataset, which we annotated with the types of anomalies present. We compared the methods on both a dataset level and an anomaly-type level to address two main questions: Does the potential superior performance of deep-learning methods justify the sacrifice of the intrinsic interpretability of classical methods? What are the similarities and differences between the analyzed methods when detecting different anomaly types? Our experiments showed that the classical machine-learning methods MDI and MERLIN outperform the deep-learning methods. The third classical method, RRFCF, was unable to detect a substantial number of anomalies, but it improved when using sequence-based statistical features instead of raw data points. Among the deep learning methods, the Autoencoder model detected the most anomalies and was also the simplest model in this group.

While we present our experimental results in this work, a deeper theoretical analysis of the reasons and mechanisms behind these results is left for future research. Regarding the second question about the similarities and differences in detecting certain anomaly types, we found that all subsequence-based methods detect the “steep increase” anomaly but not the “smoothed increase”, while the opposite is true for the method that uses point-wise features. However, these classes are too small to produce a significant result. Although MDI and MERLIN had the best results in this comparison, they detected a diverse range of anomaly types. Together, they detected most of the anomalies, i.e., they detected 11 out of 16 anomaly types. However, the anomaly types “unusual pattern”, “time shift”, “reversed” and “flat” could not be reliably detected by any of the analyzed models. A more theoretical analysis of these results will be conducted in a subsequent study.

**Author Contributions:** Conceptualization, F.R., J.D. and J.N.; Data curation, F.R.; Formal analysis, F.R.; Investigation, F.R.; Methodology, F.R.; Project administration, J.N.; Software, F.R.; Supervision, J.D. and J.N.; Validation, F.R.; Visualization, F.R.; Writing—original draft, F.R.; Writing—review and editing, F.R., J.D. and J.N. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The UCR Anomaly Archive dataset [38] that has been used in this work is available at [https://www.cs.ucr.edu/~eamonn/time\\_series\\_data\\_2018/](https://www.cs.ucr.edu/~eamonn/time_series_data_2018/) (accessed on 18 January 2023). A python repository which contains our benchmark source code and the anomaly type annotations to reproduce the results can be found at: <https://gitlab.com/dlr-dw/is-it-worth-it-benchmark> (accessed on 18 January 2023).

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

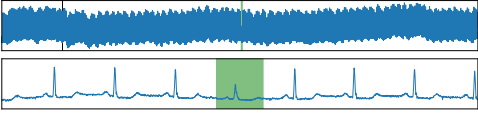
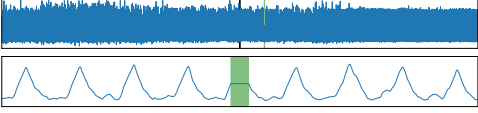
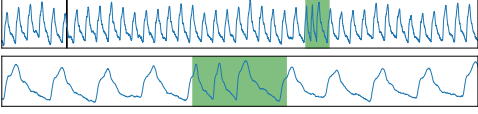
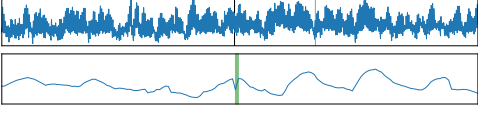
The following abbreviations are used in this manuscript:

ABP	Arterial Blood Pressure
AE	Autoencoder
ECG	Electrocardiogram
EPG	Electrical Penetration Graph
EVT	Extreme Value Theory
GANF	Graph Augmented Normalizing Flows
GPD	Generalized Pareto Distribution
ICP	Intracranial Pressure
MDI	Maximally Divergent Intervals
POT	Peaks Over Threshold
RRCF	Robust Random Cut Forest
TranAD	Transformer Network for Anomaly Detection

## Appendix A

### Appendix A.1

**Table A1.** Description of the annotated anomaly types in the UCR Anomaly Archive dataset.

Anomaly Type	Description	Example
Amplitude Change	Amplitude of the signal increased or decreased within a section.	
Flat	Flat section was added.	
Frequency Change	The cycle length was modified within a section.	
Local Drop	A drop was added, which is shallower than the minimal value of the time series.	

**Table A1.** *Cont.*

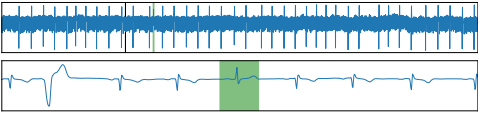
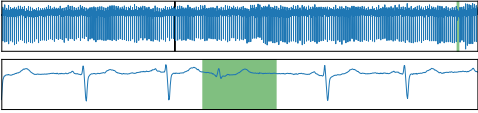
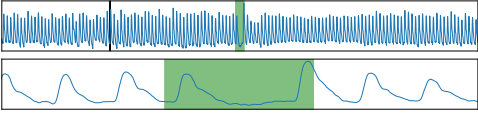
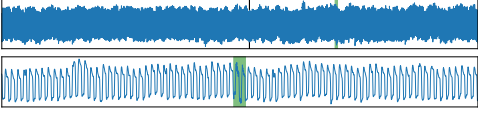
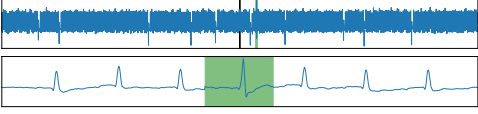
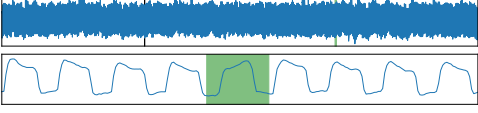
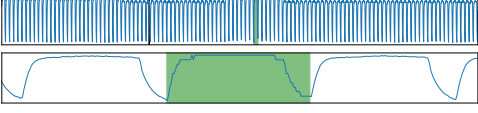
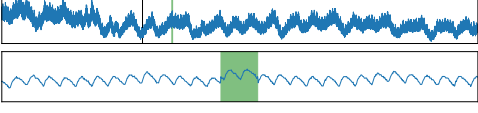
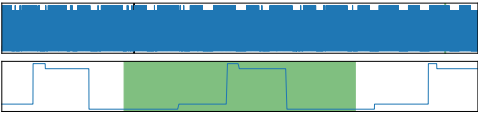
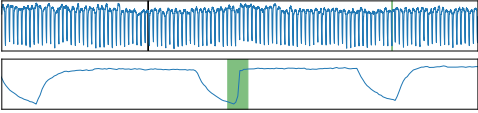
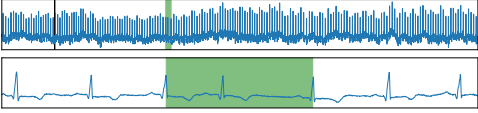
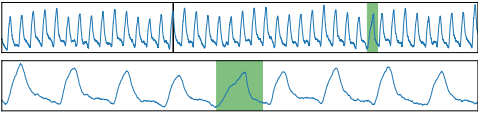
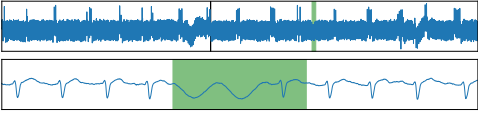
Anomaly Type	Description	Example
Local Peak	A peak was added, which is lower than the maximal value of the time series.	
Missing Drop	A drop was removed.	
Missing Peak	A peak was removed.	
Noise	Noise was added to a section.	
Outlier	A global outlier.	
Reversed	Cycle(s) got reversed.	
Sampling Rate	The sampling rate of the signal was increased or decreased in a section.	
Signal Shift	A section was shifted up or down.	
Smoothed Increase	A otherwise steep increase was smoothed, increasing the number of individual values in this section.	
Steep Increase	A otherwise smooth increase was made steep, reducing the number of individual values within this section.	
Time Shift	Increasing the pause between two peaks.	

Table A1. Cont.

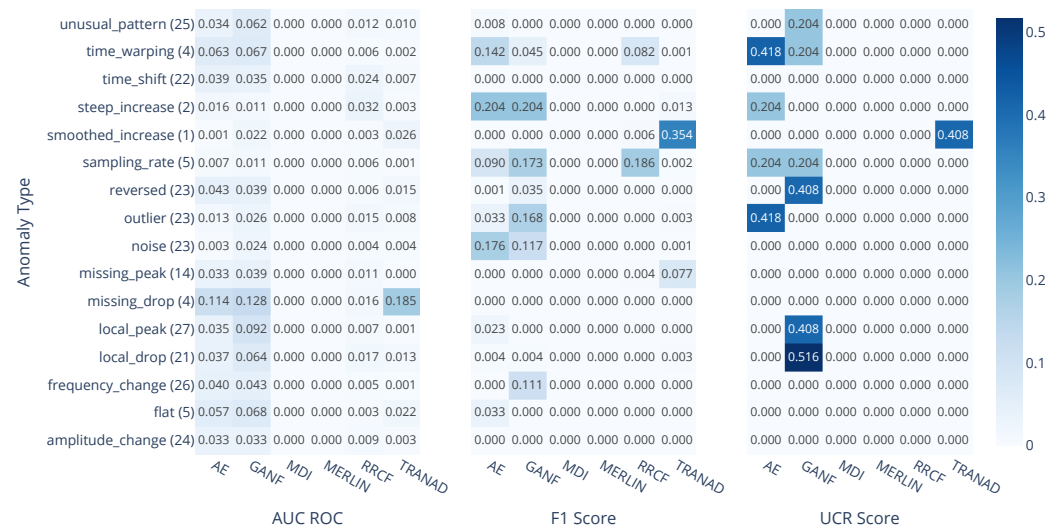
Anomaly Type	Description	Example
Time Warping	Moving the cycle peak without changing the cycle length.	
Unusual Pattern	Replacement of one or more cycle(s) with a different pattern.	

## Appendix A.2

Table A2. Tuned and not tuned parameters used in our experiments. All other parameters within the methods have been kept to their default values.

	Parameter	Value	Tuned?
AE	subsequence length	10	no
	stride	10	no
	epochs	20	no
	batch size	32	no
	latent space dimension	16	yes
	learning rate	0.005	yes
	weight decay	$10^{-5}$	no
GANF	subsequence length	100	no
	stride	10	no
	epochs	20 + 30	no
	batch size	32	no
	latent space dimension	16	yes
	learning rate	0.003	yes
	n_blocks	4	yes
	weight decay	$10^{-5}$	no
	h_tol	$10^{-4}$	no
	rho_init	1.0	no
	rho_max	$10^{16}$	no
	lambda1	0.0	no
	alpha_init	0.0	no
MDI	$L_{min}$	75	no
	$L_{max}$	125	no
MERLIN	$L_{min}$	75	no
	$L_{max}$	125	no
RRCF	n_trees	51	yes
	tree_size	1001	yes
RRCF@sequences	subsequence length	100	no
	stride	50	no
	n_trees	68	yes
	tree_size	150	yes
TranAD	subsequence length	10	no
	stride	1	no
	epochs	1	no
	batch size	128	no
	learning rate	0.02	yes
	weight decay	$10^{-5}$	no
	step size	3	yes
	gamma	0.75	yes

## Appendix A.3



**Figure A1.** The heatmaps show the standard deviation for AUC ROC, F1 Score and UCR Score over six repetitions of the experiment. Next to the anomaly type, the number of times series containing it is shown in parenthesis. The respective macro-averaged mean values can be found in Figure 6.

## References

- Ruff, L.; Kauffmann, J.R.; Vandermeulen, R.A.; Montavon, G.; Samek, W.; Kloft, M.; Dietterich, T.G.; Müller, K.R. A Unifying Review of Deep and Shallow Anomaly Detection. *Proc. IEEE* **2021**, *109*, 756–795.
- Šabić, E.; Keeley, D.; Henderson, B.; Nannemann, S. Healthcare and anomaly detection: Using machine learning to predict anomalies in heart rate data. *AI Soc.* **2021**, *36*, 149–158. [\[CrossRef\]](#)
- Li, D.; Chen, D.; Jin, B.; Shi, L.; Goh, J.; Ng, S.K. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. In *Artificial Neural Networks and Machine Learning—ICANN 2019: Text and Time Series*; Tetko, I.V., Kůrková, V., Karpov, P., Theis, F., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 703–716.
- Buczak, A.L.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. [\[CrossRef\]](#)
- Sarda, K.; Acernese, A.; Nolè, V.; Manfredi, L.; Greco, L.; Glielmo, L.; Vecchio, C.D. A Multi-Step Anomaly Detection Strategy Based on Robust Distances for the Steel Industry. *IEEE Access* **2021**, *9*, 53827–53837. [\[CrossRef\]](#)
- Park, D.; Hoshi, Y.; Kemp, C.C. A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1544–1551. [\[CrossRef\]](#)
- Freeman, C.; Merriman, J.; Beaver, I.; Mueen, A. Experimental Comparison and Survey of Twelve Time Series Anomaly Detection Algorithms. *J. Artif. Intell. Res.* **2022**, *72*, 849–899. [\[CrossRef\]](#)
- Laptev, N.; Amizadeh, S.; Flint, I. Generic and Scalable Framework for Automated Time-Series Anomaly Detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, Sydney, Australia, 10–13 August 2015*; Association for Computing Machinery: New York, NY, USA, 2015; pp. 1939–1947. [\[CrossRef\]](#)
- Guha, S.; Mishra, N.; Roy, G.; Schrijvers, O. Robust Random Cut Forest Based Anomaly Detection on Streams. In *Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016*; Balcan, M.F., Weinberger, K.Q., Eds.; PMLR: New York, New York, USA, 2016; Volume 48, pp. 2712–2721.
- Barz, B.; Rodner, E.; Garcia, Y.G.; Denzler, J. Detecting Regions of Maximal Divergence for Spatio-Temporal Anomaly Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 1088–1101. [\[CrossRef\]](#)
- Nakamura, T.; Imamura, M.; Mercer, R.; Keogh, E. MERLIN: Parameter-Free Discovery of Arbitrary Length Anomalies in Massive Time Series Archives. In *Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020*; IEEE: Sorrento, Italy, 2020; pp. 1190–1195. [\[CrossRef\]](#)
- Dai, E.; Chen, J. Graph-Augmented Normalizing Flows for Anomaly Detection of Multiple Time Series. In *Proceedings of the 10th International Conference on Learning Representations (ICLR), 25–29 April 2022, 2022*. Available online: [https://openreview.net/forum?id=45L\\_dgP48Vd](https://openreview.net/forum?id=45L_dgP48Vd) (accessed on 18 January 2023).
- Tuli, S.; Casale, G.; Jennings, N.R. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proc. VLDB Endow.* **2022**, *15*, 1201–1214. [\[CrossRef\]](#)
- Wu, R.; Keogh, E. Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress. *IEEE Trans. Knowl. Data Eng.* **2021**, *Early Access*. [\[CrossRef\]](#)

15. Gupta, M.; Gao, J.; Aggarwal, C.C.; Han, J. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.* **2013**, *26*, 2250–2267. [[CrossRef](#)]
16. Goldstein, M.; Uchida, S. Behavior analysis using unsupervised anomaly detection. In Proceedings of the 10th Joint Workshop on Machine Perception and Robotics (MPR 2014), Online, 16–17 October 2014.
17. Blázquez-García, A.; Conde, A.; Mori, U.; Lozano, J.A. A review on outlier/anomaly detection in time series data. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–33. [[CrossRef](#)]
18. Braei, M.; Wagner, S. Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. *arXiv* **2020**, arXiv:2004.00433.
19. Chalapathy, R.; Chawla, S. Deep Learning for Anomaly Detection: A Survey. *arXiv* **2019**, arXiv:1901.03407.
20. Pang, G.; Shen, C.; Cao, L.; van den Hengel, A. Deep Learning for Anomaly Detection: A Review. *ACM Comput. Surv.* **2020**, *54*, 1–38.
21. Salehi, M.; Mirzaei, H.; Hendrycks, D.; Li, Y.; Rohban, M.H.; Sabokrou, M. A Unified Survey on Anomaly, Novelty, Open-Set, and Out-of-Distribution Detection: Solutions and Future Challenges. *Trans. Mach. Learn. Res.* **2022**. Available online: <https://openreview.net/forum?id=aRtjVZvbpK> (accessed on 18 January 2023).
22. Bulusu, S.; Kailkhura, B.; Li, B.; Varshney, P.K.; Song, D. Anomalous Example Detection in Deep Learning: A Survey. *IEEE Access* **2020**, *8*, 132330–132347. [[CrossRef](#)]
23. Taylor, S.J.; Letham, B. Forecasting at scale. *Am. Stat.* **2018**, *72*, 37–45. [[CrossRef](#)]
24. Yeh, C.C.M.; Zhu, Y.; Ulanova, L.; Begum, N.; Ding, Y.; Dau, H.A.; Zimmerman, Z.; Silva, D.F.; Mueen, A.; Keogh, E. Time series joins, motifs, discords and shapelets: A unifying view that exploits the matrix profile. *Data Min. Knowl. Discov.* **2018**, *32*, 83–123. [[CrossRef](#)]
25. Xu, H.; Chen, W.; Zhao, N.; Li, Z.; Bu, J.; Li, Z.; Liu, Y.; Zhao, Y.; Pei, D.; Feng, Y.; et al. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In Proceedings of the 2018 World Wide Web Conference, WWW '18, Lyon, France, 23–27 April 2018; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2018; pp. 187–196. [[CrossRef](#)]
26. Lavin, A.; Ahmad, S. Evaluating real-time anomaly detection algorithms—the Numenta anomaly benchmark. In Proceedings of the 2015 IEEE 14th international conference on machine learning and applications (ICMLA), Miami, FL, USA, 9–11 December 2015; pp. 38–44.
27. Fluss, R.; Faraggi, D.; Reiser, B. Estimation of the Youden Index and its associated cutoff point. *Biom. J.* **2005**, *47*, 458–472. [[CrossRef](#)]
28. Graabæk, S.G.; Ancker, E.V.; Christensen, A.L.; Fugl, A.R. An Experimental Comparison of Anomaly Detection Methods for Collaborative Robot Manipulators. *IEEE Access* **2022**, Preprint. . [[CrossRef](#)]
29. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation-Based Anomaly Detection. *ACM Trans. Knowl. Discov. Data* **2012**, *6*, 3:1–3:39. [[CrossRef](#)]
30. Wang, Y.; Wang, Z.; Xie, Z.; Zhao, N.; Chen, J.; Zhang, W.; Sui, K.; Pei, D. Practical and White-Box Anomaly Detection through Unsupervised and Active Learning. In Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 3–6 August 2020; IEEE: Honolulu, HI, USA, 2020; pp. 1–9. [[CrossRef](#)]
31. MacGregor, J. Statistical Process Control of Multivariate Processes. *IFAC Proc. Vol.* **1994**, *27*, 427–437. [[CrossRef](#)]
32. Yankov, D.; Keogh, E.; Rebbapragada, U. Disk Aware Discord Discovery: Finding Unusual Time Series in Terabyte Sized Datasets. In Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM 2007), Omaha, NE, USA, 28–31 October 2007; IEEE: Omaha, NE, USA, 2007; pp. 381–390. [[CrossRef](#)]
33. De Paepe, D.; Avendano, D.N.; Van Hoecke, S. Implications of Z-Normalization in the Matrix Profile. In Proceedings of the Pattern Recognition Applications and Methods: 8th International Conference, ICPRAM 2019, Prague, Czech Republic, 19–21 February 2019; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany, 2019; pp. 95–118. .5. [[CrossRef](#)]
34. Rumelhart, D.E.; McClelland, J.L. *Learning Internal Representations by Error Propagation*; MIT Press: Cambridge, MA, USA, 1987; pp. 318–362.
35. Baldi, P. Autoencoders, Unsupervised Learning, and Deep Architectures. In *ICML Workshop on Unsupervised and Transfer Learning*; Guyon, I., Dror, G., Lemaire, V., Taylor, G., Silver, D., Eds.; Proceedings of Machine Learning Research; PMLR: Bellevue, WA, USA, 2012; Volume 27, pp. 37–49.
36. Bank, D.; Koenigstein, N.; Giryas, R. Autoencoders. *arXiv* **2020**, arXiv:2003.05991.
37. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
38. Wu, R.; Keogh, E. UCR Anomaly Archive. 2021. Available online: [https://www.cs.ucr.edu/~Eamonn/time\\_series\\_data\\_2018/UCR\\_TimeSeriesAnomalyDatasets2021.zip](https://www.cs.ucr.edu/~Eamonn/time_series_data_2018/UCR_TimeSeriesAnomalyDatasets2021.zip) (accessed on 30 January 2023).
39. Laptev, N.; Amizadeh, S.; Billawala, Y. S5—A Labeled Anomaly Detection Dataset, Version 1.0 (16M). 2015. Available online: <https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70> (accessed on 18 January 2023).
40. Hundman, K.; Constantinou, V.; Laporte, C.; Colwell, I.; Soderstrom, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 387–395.



41. Lenis, G.; Pilia, N.; Loewe, A.; Schulze, W.H.; Dössel, O. Comparison of baseline wander removal techniques considering the preservation of ST changes in the ischemic ECG: A simulation study. *Comput. Math. Methods Med.* **2017**, *2017*, 9295029. [[CrossRef](#)]
42. Wu, R.; Keogh, E. UCR\_AnomalyDataSets.pptx, Supplemental Material to the UCR Anomaly Archive. 2021. Available online: [https://www.cs.ucr.edu/~Eeamonn/time\\_series\\_data\\_2018/UCR\\_TimeSeriesAnomalyDatasets2021.zip](https://www.cs.ucr.edu/~Eeamonn/time_series_data_2018/UCR_TimeSeriesAnomalyDatasets2021.zip) (accessed on 18 January 2023).
43. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
44. Siffer, A.; Fouque, P.A.; Termier, A.; Largouet, C. Anomaly detection in streams with extreme value theory. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 1067–1075.
45. Boniol, P.; Palpanas, T.; Meftah, M.; Remy, E. Graphan: Graph-based subsequence anomaly detection. *Proc. VLDB Endow.* **2020**, *13*, 2941–2944. [[CrossRef](#)]
46. Su, Y.; Zhao, Y.; Niu, C.; Liu, R.; Sun, W.; Pei, D. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Anchorage, AK, USA, 4–8 August 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 2828–2837. [[CrossRef](#)]
47. Zhao, H.; Wang, Y.; Duan, J.; Huang, C.; Cao, D.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; Zhang, Q. Multivariate Time-Series Anomaly Detection via Graph Attention Network. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; IEEE Computer Society: Los Alamitos, CA, USA, 2020; pp. 841–850. [[CrossRef](#)]
48. Bradley, A.P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognit.* **1997**, *30*, 1145–1159. [[CrossRef](#)]
49. Lu, Y.; Wu, R.; Mueen, A.; Zuluaga, M.A.; Keogh, E. Matrix Profile XXIV: Scaling Time Series Anomaly Detection to Trillions of Datapoints and Ultra-fast Arriving Data Streams. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Washington, DC, USA, 14–18 August 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 1173–1182. [[CrossRef](#)]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.