ABSTRACT

The detection of anomalies in time series data is crucial in a wide range of applications, such as system monitoring, health care or cyber security. While the vast number of available methods makes selecting the right method for a certain application hard enough, different methods have different strengths, e.g. regarding the type of anomalies they are able to find. In this work, we compare six unsupervised anomaly detection methods with different complexities to answer the questions: Are the more complex methods usually performing better? And are there specific anomaly types that those method are tailored to? The comparison is done on the UCR anomaly archive, a recent benchmark dataset for anomaly detection. We compare the six methods by analyzing the experimental results on a dataset- and anomaly type level after tuning the necessary hyperparameter for each method. Additionally we examine the ability of individual methods to incorporate prior knowledge about the anomalies and analyse the differences of point-wise and sequence wise features. We show with broad experiments, that the classical machine learning methods show a superior performance compared to the deep learning methods across a wide range of anomaly types.

Keywords Anomaly Detection · Time Series · Machine Learning · Deep Learning · Benchmark
Three of them are classical machine learning methods\(^1\), while the remaining three are based on deep learning. The two central questions we are targeting with this comparison are:

1. "Is it worth to sacrifice the interpretability of the classical methods for a potentially superior performance of the deep learning methods?" and
2. "What different types of anomalies are the methods detecting?".

For this purpose we compare the classical methods Robust Random Cut Forest (RRCF) \([9]\), Maximally Divergent Intervals (MDI) \([10]\) and MERLIN \([11]\) to the deep learning methods Autoencoder (AE), Graph Augmented Normalizing Flows (GANF) \([12]\) and Transformer Networks for Anomaly Detection (TranAD) \([13]\). We compare the aforementioned methods on the UCR Anomaly Archive \([14]\), a novel Benchmark dataset for time series anomaly detection. This archive consists of 250 uni-variate time series from the four domains human medicine, industry, biology and meteorology. To ensure a fair comparison we carefully choose our experimental setup and performed intensive hyper-parameter tuning for those methods where applicable. To the best of our knowledge this is the first work doing an experimental comparison of classical and deep learning methods for anomaly detection in time series.

The remainder of this paper is structured as follows: After giving some fundamental introduction into time series data and different anomaly types in Sections 1.1 and 1.2 we present related work in Section 1.3. In Section 2.1 we give present the six anomaly detection methods before describing the UCR Anomaly Archive in Section 2.2 and the experiment setup in Section 2.3. The results of our experiments then presented in Section 3 and discussed in Section 4 before a conclusion is drawn 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 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 \(T\) with length \(n \in \mathbb{N}\) is defined as the set of pairs \(T = \{(t_i, p_i)| t_i \leq t_{i+1}, 0 \leq i \leq n\}\) with \(p_i \in \mathbb{R}^d\) being the data points having \(d\) behavioural attributes and \(t_i \in \mathbb{N}\) the timestamps a certain data point refers to. For \(d = 1\), \(T\) is called uni-variate, and for \(d > 1\) \(T\) is called multi-variate.

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 behaviour 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 analysed en bloc, we define a subsequence as a contiguous subset of the time series:

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

1.2 Anomalies

Although the understanding whether an observation is considered anomalous or not might differ between applications or even individual analysts, the literature distinguishes three main types: point anomalies, collective anomalies and contextual anomalies \([11, 15, 16, 17, 18]\). Point anomalies are single data points \((t_i, p_i) \in T\) deviating significantly from all other instances, for example a fraudulent transaction among legal finance transactions \([11]\). Collective anomalies refer to whole subsequences \(S_{a,b} \subseteq T\) being anomalous whereas the individual data points \((t_i, p_i) \in S_{a,b}\) would not be considered a point anomaly \([16]\). Supraventricular premature beats in an electrocardiogram (ECG) or are examples for collective anomalies. The third class are contextual anomalies, which only appear anomalous depending on specific context variables. While an outside air temperature measurement of \(28^\circ C\) during August is considered normal in Panama, it would be anomalous in Antarctica. We extend this classification in this work to 16 classes, by dividing especially the class of collective anomalies into different subclasses like "frequency change" or "time shift" which is described in Section 2.2.2.

1.3 Related Work

While many survey and review papers on anomaly detection are available \([15, 17, 18, 19, 20, 21, 22]\) there are only few works on comparing different methods experimentally.

\(^1\)We will refer to those methods as classical methods.
Table 1: Overview over the properties of the anomaly detection methods considered in this comparison.

| Mechanism   | Class            | Online/Offline | Training       | Multivariate | Anomaly Score                        |
|-------------|------------------|----------------|----------------|--------------|---------------------------------------|
| RRCF        | Isolation Forest | classical      | online         | X            | Collusive Displacement                 |
| MDI         | Density Estimation | classical    | offline        | X            | (KL/JS) Divergence                     |
| MERLIN      | Discord Discovery | classical     | offline        | X            | Discord Distance                       |
| AE          | Reconstruction   | deep-learning | offline training | ✓            | Reconstruction Loss                    |
| GANF        | Density Estimation | deep-learning | online inference | ✓            | Density                               |
| TranAD      | Reconstruction   | deep-learning | offline training | ✓            | Reconstruction Loss                    |

Freeman et al. [7] do an experimental comparison of twelve anomaly detection methods like Seasonal AutoRegressive Integrated Moving Average with exogenous variables (SARIMAX), Generalized Linear Model, Facebook Prophet [23], Matrix Profile [24] or Donut [25]. The comparison is done using a dataset compiled mainly from the Numenta benchmark [26] with a focus on time series different characteristics like seasonality and trend. They use the Youden Index [27] to determine a threshold for classifying anomaly scores and assess the quality of the analysed methods using AUC ROC, Windowed-F1 and 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 analysed methods are categorized as instance based like k-Nearest-Neighbours 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] give an 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 an unifying view of the anomaly detection problem by identifying specific anomaly detection modeling components to characterize the presented methods and exemplify the modelling and evaluation process on two real-world examples.

2 Materials and Methods

2.1 Analyzed Methods

For this comparison we selected three deep-learning and three classical machine learning methods for unsupervised anomaly detection. The selection-criteria for choosing the methods have been various, reaching from the methods simplicity and interpretability to the applicability to data streams and the existence of interesting side-features like the dependency graph for GANF. In the following section we will present the selected methods while starting with the classical ones. One criterion to categorize anomaly detection methods is their applicability to data streams. We will refer to anomaly detection on data streams as online anomaly detection whereas anomaly detection on data batches will be called offline. An overview over the properties of the compared methods can be found in Table 1.

2.1.1 Robust Random Cut Forest (RRCF)

The Robust Random Cur Forest (RRCF) [9] is a modification of the well-known Isolation Forest [29] methods, that extends the approach to data streams. Both methods detect anomalies by isolating single points from all other instances by recursively partitioning the set of data points. This recursive 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 less cuts are needed to isolate a point, the likelier it is that this point is anomalous [29]. The main modification in the construction of the forest is that in Isolation Forest, the next dimension to cut is chosen uniformly at random whereas in RRCF, the next dimension is chosen with a probability proportional to the range of values in the respective dimension. 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 else-wise 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 and is defined as the expected change in the depth of points in a tree when removing a set C ∪ {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 it's strength in finding point anomalies. For the detection of anomalous subsequences an additional preprocessing step for constructing window-based features could be considered and is analysed 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, spatio-temporal data. As we are focusing on purely temporal data in this work, we will give the definitions for temporal data only. For the original definitions including spatial attributes please refer to [10]. Given a multivariate time series 𝒰, MDI detects anomalous subsequences by comparing the probability density 𝑝_S of a subsequence 𝑆_{a,b} ⊆ 𝒰 to the density 𝑝_{Ω_S} of the remaining part of the times series Ω(𝒮) := 𝒰 \ S_{a,b} for all subsequences. The distributions are modelled using Kernel Density Estimation or Multivariate Gaussians. To measure the degree of deviation 𝒟(𝑝_S, 𝑝_{Ω_S}) between 𝑝_S and 𝑝_{Ω_S}, an unbiased version of the Kullback-Leibler divergence is used. The most anomalous subsequence ˇ𝒮 is found by solving the underlying optimization problem [10]:

\[ ˇ𝒮 := \arg \max_{\mathcal{S} \subseteq \mathcal{T}} \mathcal{D}(\mathcal{P}_\mathcal{S}, \mathcal{P}_{\Omega(\mathcal{S})}) \]

MDI locates this most anomalous subsequence ˇ𝒮 by scanning over all subsequences 𝒰 ⊆ 𝒰 with a length between 𝐿_{min} and 𝐿_{max} and estimates the divergence 𝒟(𝑝_S, 𝑝_{Ω_S}), which is then used as the anomaly score. The parameters 𝐿_{min} and 𝐿_{max} need 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 pre-selection method is motivated by the fact, that most subsequences are uninteresting for detecting anomalies as those 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 so-called discord discovery: Given a subsequence 𝒰 with length 𝐿 starting at timestamp 𝑝, a matching subsequence 𝑀 starting at timestamp 𝑞 is called a non-self match to 𝒰 if |𝑝 − 𝑞| ≥ 𝐿 [11]. The discord ˇ𝒮 of a time series 𝒰 is defined as the subsequence, which has the largest distance 𝑑(ˇ𝒮, 𝑀_2) from its nearest non-self match 𝑀_2 with 𝑑(·, ·) being the z-normalized (zero mean and unit variance) euclidean distance. MERLIN is based on the discord discovery algorithm from [32]. Crucial for this algorithm to finish successfully and in acceptable time is the choice of a hyper-parameter 𝑟, which should be chosen "a little less" than the discord distance 𝑑(ˇ𝒮, 𝑀_2). If 𝑟 is chosen too large, the algorithm fails. If it is chosen to small, the runtime explodes. As this hyper-parameter can not be guessed, MERLIN provides a structured search for determining a good value for 𝑟 by exploiting that a good value of 𝑟 for subsequences of length 𝐿 is likely to be very similar to a good value of 𝑟 for subsequences of length 𝐿 − 1 [11]. The largest possible value of 𝑟 for subsequences of length 𝐿 is 𝑟_{max}(𝐿) = 2√𝐿 [33]. Starting from this maximal value for the smallest subsequence length to be considered 𝑟 = 𝑟_{max}(𝐿_{min}), 𝑟 is halved until the first discord is returned. For every other subsequence length 𝐿_{min}, . . . , 𝐿_{max}, the information about previously found values for 𝑟 can be used. We selected MERLIN as it is the method released with the UCR anomaly archive dataset, which is the used benchmark dataset for our study and will be explained later in Section 2.2.

2.1.4 Autoencoder (AE)

Autoencoders, introduced in [34], are neural network models for dimensionality reduction consisting of an encoder network \( f : \mathbb{R}^n \rightarrow \mathbb{R}^l \) and a decoder network \( g : \mathbb{R}^l \rightarrow \mathbb{R}^n \) with \( p, n \in \mathbb{N}, l < n \) that are trained to reconstruct their input by learning a latent representation. The autoencoder problem can be formalised 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]. Applied to the task of unsupervised anomaly detection in time series, the autoencoder learns a normal profile of the time series \( \mathcal{T} \) and detects anomalous input sequences ˇ𝒮 by a high reconstruction error. Figure 1 provides a visualisation of this approach.
In our experiments we used a dense autoencoder with two hidden layers in the encoder and decoder which have each the doubled number of neurons of the latent space and ReLU activations as nonlinearities. We included the autoencoder as a basic deep-learning model into this study.

2.1.5 Graph Augmented Normalizing Flows (GANF)

GANF [12] is an anomaly detection method for multivariate time series based on normalizing flows for density estimation. Normalizing Flows are generative models \( f : \mathbb{R}^d \rightarrow \mathbb{R}^d \) that employ a sequence of invertible and differentiable transformations to normalize complex data distributions to base distributions whose densities are easy to evaluate, usually isotropic Gaussians [12]. Additionally to modelling the density of the time series using a normalizing flow, GANF imposes a Bayesian Network to model the causal relationships among multiple multivariate time series \( X = (T_1, \ldots, T_m) \). Given a training set \( D = \{ X_i \}_{i=1}^{|D|} \) of multiple time series, GANF aims at learning the adjacency matrix \( A \) of the Bayesian Network and simultaneously the graph-augmented normalizing flow \( F : (\mathcal{X}, A) \rightarrow \mathcal{Z} \) with \( \mathcal{Z} \) being a random variable with a ”simple” (base) distribution [12]. Once \( F \) is learned, the estimated density \( p(X) \) can be evaluated to find anomalies in low density regions of the base distribution. The dependency encoder of the model consists of a recurrent neural network to summarise the time series up to a given time step \( t \) and a graph convolution layer to then learn a dependency representation. This representation is then used to condition a normalizing flow \( f \). For the architectural details of GANF please refer to [12]. As anomalies are rare by definition, the assumption is that their densities are low. Hence, 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 and it’s interesting feature to learn dependencies between multiple time series even though 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 a two phase training. In the first phase, the model learns an approximate reconstruction of the whole time series \( T \) 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 was 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 date of its selection.

2.2 Benchmark Dataset: UCR Anomaly Archive

The benchmark dataset for this study is the UCR Anomaly Archive [14,38], which is a collection of 250 uni-variate time series collected in human medicine, biology, meteorology and industry. The collected time series contain a few natural anomalies though the majority of the anomalies are artificial which allows a more detailed analysis based on the type of injection. The dataset was first used in an anomaly detection contest preceding the ACM SIGKDD conference 2021. Wu and Keogh [14] published the dataset as an alternative to existing and widely used benchmark datasets like Yahoo S5 [39], Numenta [26] or NASA [40]. According to [14], those datasets suffer from triviality, unrealistic anomaly densities, mislabeled ground truth and are run-to-failure biased. Run-to-failure bias refers to anomalies often occurring at the end of a time series as the time series recording was stopped after the anomaly (or failure) occurred. Each of the time series contains exactly one, occasionally subtle anomaly after a given time stamp. The data before that time...
Is It Worth It?

2.2.1 Included Time Series

The time series collected in the UCR Anomaly Archive can be categorized into 12 types originating from the four domains human medicine, meteorology, biology and industry. Figure 2a shows the histogram of time series type distribution.

The distribution across the domains is highly imbalanced with around 64% of the times series being collected in human medicine applications, 22% in biology, 9% in industry and 5% being air temperature measurements. The time series within a single type (e.g. ECG) are not completely unique, but differ in terms of injected anomalies or a modification of the original time series through added Gaussian noise and wandering baselines. Baseline Wander is a typical low-frequency artifact in ECG caused e.g. by breathing or subject movement and refers to slow changes of the signal baseline [41].

2.2.2 Anomaly Types

To analyse the six anomaly detection methods regarding their different abilities to detect certain types of anomalies, we annotated each time series with the type of the injected anomaly. We therefore relied on the supplemental material in [42] distributed with the dataset. The histogram of anomaly types is shown in Figure 2b. A list of explanations and examples for each anomaly type can be found in Appendix F.2.
2.3 Experimental Setup

For our experiments we implemented the benchmark pipeline described in Section 2.3.1 in Python. The anomaly detection methods have been either integrated from their publicly available GitHub repositories (RRCF\(^2\), MDI\(^3\), GANF\(^4\) or have been implemented by us as it was not available in Python (MERLIN\(^5\)). The Autoencoder model was implemented using the PyTorch\[^43^\] library and is available in the repository for this work\(^6\). The relevant hyperparameters for each model have been tuned by performing 20 rounds of Bayesian Optimization on 25 randomly selected time series from the UCR Anomaly Archive with the F1-Score being the optimization target. The time series used for hyper-parameter tuning have been excluded from the actual experiments. All experiments have been carried out on an Intel Xeon Platinum 8260 CPU with an allocated Memory of 10GB\(^7\). We ran all experiments six times: the first time we seeded the random number generators of Python, Numpy and Pytorch with a fixed value\(^8\) and then performed 5 repetitions without setting a random seed to account for random sampling effects.

2.3.1 Benchmark Pipeline

To ensure a controlled environment and fairness across the experiments we implemented the pipeline shown in Figure 3. The time series are normalized to the interval \([0, 1]\) and a sliding window is applied depending to the needs of the respective method. While AE, GANF and TranAD require that the input data given as fixed length subsequences, MDI and MERLIN require the whole time series as input together with a range of subsequence lengths \(L_{\text{min}}\) and \(L_{\text{max}}\). For MDI and MERLIN we fixed that range to \(L_{\text{min}} = 75\) and \(L_{\text{max}} = 125\) time steps. The detailed window sizes and strides per methods can be found in Appendix F.3

![Figure 3: Benchmark pipeline used in the experiments.](https://github.com/kLabUM/rrcf)

The normalized time series or subsequences are then used as input to the respective anomaly detection method which calculates an anomaly score. MERLIN is one exception here as this method returns only the subsequences detected as anomalous. As the produced scores are very heterogeneous across the methods we use a method called Peak Over Threshold (POT)\[^44\] to determine a suitable threshold to classify subsequences as normal or anomalous. This approach was also taken in prior works such as \[^40, 45, 13\].

2.3.2 Anomaly Score Classification

To ensure a fair comparison of results produced by the six anomaly detection methods we determine a threshold for classifying the subsequences as anomalous or normal automatically employing the principle of Extreme Value Theory (EVT)\[^44\]. EVT aims at finding the law of extreme values which are usually located in the tails of a probability distribution without making any assumptions on the data distribution\[^46\]. Peaks-Over-Threshold (POT)\[^44\], the second theorem in EVT, is a statistical method that fits the tail portion of a probability distribution with a Generalized Pareto Distribution. In the context of anomaly score classification, POT is used to learn a suitable threshold of the anomaly scores\[^46\]. The anomaly label \(y_i\) for a predicted subsequence \(\hat{S}_{i,i+w}\) with length \(w\) is then given by:

\[
y_i = \mathbb{1}(\sigma_i \geq \text{POT}(\sigma_i)),
\]

with \(\sigma_i\) being the anomaly score for the subsequence \(\hat{S}_{i,i+w}\), \(\text{POT}\) being the Peaks-Over-Threshold algorithm from\[^46\] that determines the threshold based on the anomaly score and \(\mathbb{1}(\cdot)\) being the indicator function. For details on POT we refer to \[^46\] and \[^44\]. We initialize the POT classifier on the first 10% anomaly scores and use a static risk of 0.01. For

---

\(^2\)https://github.com/kLabUM/rrcf  
\(^3\)https://github.com/cvjena/libmaxdiv  
\(^4\)https://github.com/EnyanDai/GANF  
\(^5\)https://gitlab.com/dlr-dw/py-merlin  
\(^6\)https://gitlab.com/dlr-dw/is-it-worth-it-benchmark

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”.  
\(^8\)We used 42 as the seed value across all experiments.
MERLIN we omit this step as the method already returns binary labels per subsequence. As it is usually acceptable that an algorithm detects 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 e.g. in [47, 13] for adjusting the predicted anomalous labels e.g. 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 from this point forward as correctly detected and treat all points within this 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, which is the measure recommended for the UCR Anomaly Archive. To calculate 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 for 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 better the classifier can separate both classes from another. A perfect classifier achieves a score of 1 by ranking all examples of the positive class higher than all examples of the negative class. Hence, we use AUC ROC as a measure for the quality of the produced anomaly scores, where a high score indicates a good separability of normal and anomalous points or subsequences. We are aware that AUC ROC is actually no suitable measure for unbalanced problems like the detection of anomalies, where the “anomalous” class is small by definition, but report it due to it’s widely use throughout the literature.

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

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

with TP, FP and FN being the True Positive, False Positive and False Negative detections and precision and recall being defined as:

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

As the F1-Score is calculated on the result of the binary classification it adds evidence about the quality of the used threshold. If a method receives a high AUC ROC but a low F1 Score this would indicate a poor threshold.

UCR score The UCR score is the recommended metric distributed with the UCR Anomaly Archive and is a binary score telling whether or not a method was able to find that single anomaly in the dataset. It is defined as:

$$\text{UCR score} := 1(\min(a - L, a - 100) < t^* < \max(b + L, b + 100))$$

with a and b being the begin and end of the true anomaly with length $L$, $t^*$ the timestamp of the point with the highest anomaly score and $1(\cdot)$ the indicator function. For subsequences $S_{ij}$ we use the middle point $t^* = i + \left\lfloor \frac{j - i}{2} \right\rfloor$. The tolerance of 100 timestamps is added to account for very short anomalies [42]. A UCR score of 0.5 would mean that for half of the analysed time series the annotated anomaly was successfully found having the highest anomaly score. A UCR score of 1 but a low F1-Score at the same time indicates the detection of the true anomaly with the highest anomaly score but the presence of false positive or false negative results. A low UCR score but a high F1-Score on the other hand would indicate that the anomaly was found without many false positives or false negatives, but the anomaly score for the true anomaly ranked lower than for false detection.

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 analyse the influence of varying subsequence length on MDI and MERLIN and thus their ability to utilise addition 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.

---

9The scoring function is not named in [42], so we call it UCR score
3.1 Performance Analysis by Method

We compare the performance of the six anomaly detection methods using macro-averaged AUC ROC, F1 score and UCR score as well as the average runtime for a single time series. MDI achieves the highest AUC ROC and UCR score among the six compared methods while MERLIN performs better in terms of F1 Score. Among the deep learning methods AE achieves the highest AUC ROC and UCR score of 0.58 and 0.28 respectively while TranAD has a slightly higher F1 score compared to AE. RRCF fails to detect a notable amount of anomalies in the test set and has by far the lowest F1- and UCR score. The numerical results are shown in Table 2. The scores shown in Table 2 are all rather low across all methods which is caused by times series in the test set producing low or zero scores. We discuss meanings of different combinations of high and low scores as well as their reasons in Section 4. As MDI and MERLIN are deterministic methods, they are not subject to sampling effects and therefore have a standard deviation of 0.0 among the 6 repetitions of the experiment. When comparing the average runtime for a single time series from the UCR anomaly Archive, MDI achieves the best result of 74 seconds. AE shows twice the runtime of MDI and those for RRCF (162s) and TranAD (183s) are slightly higher but all below 200 seconds. GANF and MERLIN have slower runtimes of more than 200 seconds on average for a single time series with MERLIN having the worst runtime of 291 seconds which is almost four times that of MDI. Those runtimes are also subject to the concrete implementations. MDI is implemented in C++ and has a Python interface, all other methods are purely implemented in Python. For the deep learning methods AE, GANF and TranAD the training time is included.

To gain insights regarding the title question of this paper, we aggregated the results by model class, which are shown in Figure 5 as violin plots. The violin plots visualise the kernel density estimates for the two classes ”Classical Machine Learning Methods” which contains MDI, MERLIN and RRCF and ”Deep Learning Methods” which contains AE, GANF and TranAD. All density curves have two peaks: one around 0.5 for AUC ROC and 0.0 for F1- and UCR score respectively and a smaller one around 0.9 (F1 score) and 1 (AUC ROC and UCR score). The peaks around 0.5 and 0 contain those results, where the methods failed to detect the anomaly. The peaks around 0.9 and 1 mark those where the anomaly detection was successful. For the classical ML methods, the area under the peaks at 0.9 and 1 is larger compared to the deep learning methods across all measures, indicating more successful anomaly detection for the ”Classical ML” class. Consequently, the area under the peaks at 0.5 and 0 is larger for the deep learning methods.
3.2 Performance Analysis by Anomaly Type

The second interest in this study are the differences between the analyzed methods to detect certain types of anomalies. We therefore aggregated our results by the 16 anomaly classes described in Section 3.2.2. 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 actually represent the same data but was distorted in one case that 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 result does not differ between multiple runs. AE and GANF find this anomaly in 11/12 and 10/12 cases respectively. TranAD detects the “steep_increase” anomaly in 7/12 cases and reliably only in the undistorted version of the time series. The one detection in the distorted version comes from the run made with a fixed random seed. RRCF is the only method, that detects 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. This is also the only type of anomaly RRCF is able to 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 a F1 score of 0.86. For the remaining anomaly types, the results are more diverse. MERLIN and TRANAD find the majority of the 23 “outlier” anomalies with TranAD performing better on this type. The 23 “noise” anomalies however are detected by AE 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 0.5 for the classes “time warping”, “steep 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 tendency in the results for AE 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 “steep increase” with the latter having a F1 Score of 0.92. MERLIN shows a UCR score above or equal 0.5 for “steep increase”, “outlier”, “missing drop”, “local peak”, “frequency change” and “amplitude change” anomalies, making “steep increase” and “missing peak” the only classes where 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 “steep increase”. GANF achieves a UCR score of 0.83 for “steep increase” anomalies. TranAD achieves a UCR score above or equal 0.5 for the classes “outlier” and “steep increase” but for the latter the corresponding low F1 Score indicates a high number of false positives. For the ”sampling_rate” anomalies it is vice versa, as the F1 Score is 0.61 here but the UCR score is 0.0. We will discuss those differences in Section 4 in more detail. MDI and MERLIN together detect the anomalies of more than two third of the annotated anomaly types. For the classes “flat”, “reversed”, “time shift” and ”unusual pattern”, no method achieved a UCR score above or equal 0.5.

3.3 The influence of subsequence length on MDI and MERLIN

To ensure a fair comparison, we fixed the subsequence length range for MDI and MERLIN in the results presented in Section 3.1 and 3.2 to \( L_{\text{min}} = 75 \) and \( L_{\text{max}} = 125 \) time steps, regardless the specifics of the individual time series like cycle length or expected length of the anomaly. The goal of this experiment was to examine the influence of the subsequence length on the results for MDI and MERLIN and their ability to utilise additional information about the problem domain given with the range of subsequence lengths. We therefore compare the baseline results from Section 3.1 to two strategies for setting the subsequence range. For the “dynamic” strategy, we provide additional
F1 Score
UCR Score

0
1
0
1
0
2000
2000
2000
2000
0
amplitude_change (24)
smoothed_increase (1)

836500

0
2000 4000 6000 8000 10000
1
0
1
9250 9300 9350 9400 9450
1
0
1

051_UCR_Anomaly_DISTORTED.TkeepSecondMARS_3500_9330_9340

For the "fixed" strategy, we choose a fixed length of 100 time steps and therefore reduce the given information by setting the range of subsequence lengths based on the length $L_{S_{a,b}} = b - a$ of the true anomaly $\tilde{S}_{a,b}$ to $L_{S_{a,b}} \pm 25\%$. For the "fixed" strategy, we choose a fixed length of 100 time steps and therefore reduce the given information compared to the baseline.

Figure 8a shows the results for MDI with a decreased AUC ROC and F1 score when fixing the subsequence length to 100 and hence reducing the given information and an increased AUC ROC and F1 score when choosing the range for the subsequence length dynamically. The results for UCR score do not reflect this. The highest UCR score is still achieved with the baseline configuration but the differences to the fixed and dynamically chosen subsequence length are rather small. For MERLIN the results displayed Figure 8b show a similar behaviour, but the differences between the strategies are stronger. For MERLIN the positive effect of additional information is present across all three metrics.

3.4 RRCF on sliding window statistics

RRCF applied to points-wise features is tailored towards finding point anomalies due to it's principle of isolating single points. In this experiment we compare this 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 \( \text{tree\_size} \) as described in Section 2.1. The results are shown Figure 2. 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 analysed methods. Also the UCR score increased for RRCF@sequences by a factor of 5 from 0.03 to 0.15. The F1 Score does not change notably. While RRCF applied to point-wise features was the only method detecting the "smoothed increase" anomaly, this anomaly is not detected anymore. Instead, RRCF applied to subsequence-based features now detected the 'steep increase' anomalies like all other five methods. For all other anomaly types except "midding drop", the UCR score increase for RRCF@sequences. The highest increase is made for the "time warping" anomaly from 0.0 to 0.4.

## 4 Discussion

Before we discuss the different results that we got for the individual methods we will present how to interpret the different metrics and their combinations. The rather low macro averaged scores across all methods shown in Table 2 have different reasons. For example can a low F1 score being caused by an insufficient anomaly score prohibiting the detection of the true anomaly or it can be subject to a poor choice for the threshold leading to an increased number of false positives. A UCR score of 1 marks the correct detection of the true anomaly with the highest anomaly score whereas a UCR score of 0 tells nothing about whether the anomaly was found but a false positive result has a higher anomaly score or was not detected at all. Figure 10 illustrates this by different results for the autoencoder model. A high F1 score and a UCR score of 1 at the same time indicate the successful detection of the true anomaly without any or with very few false positives results, depending on the value for F1 score as shown in Figures 10a and 10d. Low F1 score and a UCR score of 0, as shown in Figures 10b and 10f, denote that the anomaly was not detected caused by an insufficient anomaly score. A high AUC ROC value in this case can indicate that the anomaly could have been
Figure 10: Interpretation of results shown on examples of the autoencoder model. The top figure represents the time series with the true anomaly marked with 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.

detected with a low anomaly score but the threshold was chosen too high such that the subsequence was not classified as anomalous. Figures [10b] and [10f] illustrate as well, that AUC ROC is in general not a suitable measure to assess the quality of results in highly unbalanced problems in a meaningful way as in both cases the anomaly score is not suitable for detecting the true anomaly. A high F1- but a UCR score of 0, as shown in Figure [10c] flag that the true anomaly was detected but a false positive result has a higher anomaly score.

A UCR score of 1 but a low F1 score indicate the correct detection of the true anomaly with the highest anomaly score, but either a poor threshold value, as shown in Figure [10e], or the subsequence length of the detected anomaly being much longer as in Figure [11a] or shorter than the ground truth label as in Figure [12b] 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, that is considered in the definition of the UCR score in Equation [1]. This is shown in Figure [12a].

The best results in terms of F1- and UCR score are achieved by MDI and MERLIN with MERLIN having a slightly higher F1 score while MDI scores slightly higher in terms of UCR score. The differences are around 0.02. This difference in F1 score is most probably caused by the different methods for choosing the threshold. Both methods return a score only for the detected anomalous sequences but MDI either needs given the number of anomalies that should be returned or MDI might return a score for up to every subsequence. As we did not want to give MDI an advantage over it’s competitors, we choose the latter option and determined a threshold using POT instead of using the minimum anomaly score as done 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- and UCR score which seems caused by its mechanism to isolate single points and being therefore tailored towards point anomalies.

Among the deep learning methods, the results are mixed. While TranAD shows the lowest UCR score it achieves the best F1 score which indicates that it detects less often the true anomaly with the highest anomaly score as the autoencoder model but might be a little less prone to false positives. However, the differences are quite small.
Although the low results for RRCF, the classical machine learning methods show a superior performance compared to the deep learning methods when aggregating the results by method class as presented in Figure 5. The difference is most notable in the F1 score. An assumption might be, that those methods perform better in the unsupervised setting that do not require a training phase, but this is contradicted by the RRCF results.

Runtime-wise MERLIN shows the longest time of 291 seconds that is needed on average for processing a single time series. The runtime for MERLIN is mainly determined by that for the discord discovery algorithm which is called for every subsequence length $L_i \in \{L_{min}, ..., L_{max}\}$. The complexity of that algorithm is quadratic in the size of the set of potential discords which is determined in the candidate selection phase but becomes effectively linear for for small candidate sets that are produced by a good choice for the parameter $r$ [11]. As MERLIN starts with the highest possible value for $r$ and decreases it, it is unlikely to run into the case where a small $r$ value causes the candidate subset to become prohibitively large. MDI in turn employs an subsequence proposal technique based on Hotelling’s $T^2$ method [31] which selects interesting subsequences based on point anomaly scores instead of performing full scans over the data. Apart from those differences in selecting candidate subsequences, the concrete implementations of the algorithms have a strong influence on their runtime. While we implement MERLIN purely in Python, MDI is implemented in C++ and has a Python interface. All methods except RRCF detect the "steep increase" anomaly shown in Figure 7a with varying UCR scores from 0.58 for TranAD up to 1.0 for MDI and MERLIN. The low F1-Score for MDI indicates a high number of false positive results which are in this case subject to the length of the detected subsequence as shown in Figure 11a. The lower F1 score for GANF is mainly caused by the anomaly being detected within the 100 time steps tolerance around the anomaly which spans across ten time steps. The low F1 score for TranAD however is caused by a poor threshold leading to an increased number of false positive results as shown in Figure 11b.

RRCF is the only method that detects the "smoothed increase" anomaly shown in Figure 7b but not the "steep increase" anomaly like the other five methods. This behaviour can be explained by the working principle of RRCF 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 RRCF does not find the outlier anomalies seems contradicting but is due to the time series containing other extreme values, e.g. with 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 are able to utilise additional information about the anomalies but this effect is stronger for MERLIN. Providing additional information in terms of subsequence length of the true anomaly increased the F1 score for MDI but decreased the UCR score by 0.02 which indicates that MDI utilised the additional information to reduce the number of false positive result. For MERLIN the F1- and UCR score increased, meaning that the information about the true anomaly length helped MERLIN to find anomalies it missed before.

In our last experiment we changed from point-wise features for RRCF to subsequence based statistics which increased the UCR score and AUC ROC. The macro-averaged F1 score however decreased slightly which is caused by the "smoothed increase" anomaly not being detected anymore. Instead, RRCF@sequences finds the "steep increase" anomaly like all other five methods, which indicates that this anomaly can only be detected on a subsequence level. The low F1 scores for RRCF@Sequences on those time series with a UCR score of 1 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 we used for RRCF@Sequences. Figure 12 illustrates these two cases.

5 Conclusions

In this work we compared six anomaly detection methods with three being classical machine learning methods and three being based on deep learning. We performed extensive experiments on the UCR Anomaly Archive benchmark dataset.
which we annotated with the contained anomaly types. We compared the methods aggregated on a dataset- and on an anomaly type level to gain insights regarding two main questions. Does a potential superior performance of the deep learning methods justify to sacrifice the intrinsic interpretability of the classical methods? And what are commonalities and differences of the analysed methods regarding the detection of different anomaly types? We showed with our experiments that the classical machine learning methods MDI and MERLIN clearly outperform the deep learning methods. The third classical method RRCF however failed to find a substantial amount of anomalies but improved when using sequence based statistical features instead of the raw data points. Among the deep learning methods, the most anomalies have been found by the Autoencoder model which 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 mechanism that lead to those 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 fail for the "smoothed increase" while it is vice versa for the method that uses point-wise features. Unfortunately those classes are too small too give a significant result. Even though MDI and MERLIN show the best results in this comparison, the detected anomaly types are very diverse. But MERLIN and MDI together found most of the anomalies of 11/16 anomaly types. However, the anomaly types "unusual pattern", "time shift", "reversed" and "flat" could not be detected reliably by any of the analysed models. A more theoretical analysis of those results should be done in a subsequent study as well.

References

[1] 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. Proceedings of the IEEE (2021) 1-40 2020, [arXiv:cs.LG/2009.11732] https://doi.org/10.1109/JPROC.2021.3052449.

[2] Šabić, E.; Keeley, D.; Henderson, B.; Nannemann, S. Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data. AI & SOCIETY 2021, 36, 149–158.

[3] 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 Proceedings of the 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, 2019; pp. 703–716.

[4] Buczak, A.L.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials 2016, 18, 1153–1176. https://doi.org/10.1109/COMST.2015.2494502

[5] Sarda, K.; Acernese, A.; Nolè, V.; Manfredi, L.; Greco, L.; Gielmo, L.; Vecchio, C.D. A Multi-Step Anomaly Detection Strategy Based on Robust Distances for the Steel Industry. IEEE Access 2021, 9, 53827–53837. https://doi.org/10.1109/ACCESS.2021.3070659

[6] Park, D.; Hoshi, Y.; Kemp, C.C. A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder. IEEE Robotics and Automation Letters 2018, 3, 1544–1551. https://doi.org/10.1109/LRA.2018.2801475

[7] Freeman, C.; Merriman, J.; Beaver, I.; Mueen, A. Experimental Comparison and Survey of Twelve Time Series Anomaly Detection Algorithms. Journal of Artificial Intelligence Research 2022, 72, 849–899. https://doi.org/10.1613/jair.1.12688
[8] Laptev, N.; Amizadeh, S.; Flint, I. Generic and Scalable Framework for Automated Time-Series Anomaly Detection. In Proceedings of the Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; Association for Computing Machinery: New York, NY, USA, 2015; KDD ’15, p. 1939–1947. https://doi.org/10.1145/2783258.2788611

[9] Guha, S.; Mishra, N.; Roy, G.; Schrijvers, O. Robust Random Cut Forest Based Anomaly Detection on Streams. In Proceedings of the Proceedings of The 33rd International Conference on Machine Learning; Balcan, M.F.; Weinberger, K.Q., Eds.; PMLR: New York, New York, USA, 2016; Vol. 48, Proceedings of Machine Learning Research, pp. 2712–2721.

[10] Barz, B.; Rodner, E.; Garcia, Y.G.; Denzler, J. Detecting Regions of Maximal Divergence for Spatio-Temporal Anomaly Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 2018, 41, 1088–1101. https://doi.org/10.1109/TPAMI.2018.2823766.

[11] 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); IEEE: Sorrento, Italy, 2020; pp. 1190–1195. https://doi.org/10.1109/ICDM50108.2020.00147.

[12] Dai, E.; Chen, J. Graph-Augmented Normalizing Flows for Anomaly Detection of Multiple Time Series. 2021.

[13] 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. https://doi.org/10.14778/3514061.3514067.

[14] Wu, R.; Keogh, E. Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress. IEEE Transactions on Knowledge and Data Engineering 2021. https://doi.org/10.1109/tkde.2021.3112126.

[15] Gupta, M.; Gao, J.; Aggarwal, C.C.; Han, J. Outlier detection for temporal data: A survey. IEEE Transactions on Knowledge and data Engineering 2013, 26, 2250–2267.

[16] Goldstein, M.; Uchida, S. Behavior analysis using unsupervised anomaly detection. In Proceedings of the The 10th Joint Workshop on Machine Perception and Robotics (MPR 2014). Online, 2014.

[17] Blázquez-García, A.; Conde, A.; Mori, U.; Lozano, J.A. A review on outlier/anomaly detection in time series data 2020. [arXiv:cs.LG/2002.04236].

[18] Braei, M.; Wagner, S. Anomaly Detection in Univariate-Time-series: A Survey on the State-of-the-Art 2020. [arXiv:cs.LG/2004.00433].

[19] Chalapathy, R.; Chawla, S. Deep Learning for Anomaly Detection: A Survey 2019. [arXiv:cs.LG/1901.03407].

[20] Pang, G.; Shen, C.; Cao, L.; van den Hengel, A. Deep Learning for Anomaly Detection: A Review. ACM Computing Surveys, 2020 2020. [arXiv:cs.LG/2007.02500] https://doi.org/10.1145/3439950.

[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 2021. [arXiv:cs.CV/2110.14051].

[22] Bulusu, S.; Kailkhura, B.; Li, B.; Varshney, P.K.; Song, D. Anomalous Example Detection in Deep Learning: A Survey 2020. [arXiv:cs.LG/2003.06979].

[23] Taylor, S.J.; Letham, B. Forecasting at scale. The American Statistician 2018, 72, 37–45.

[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 Mining and Knowledge Discovery 2018, 32, 83–123.

[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 Proceedings of the 2018 World Wide Web Conference; International World Wide Web Conferences Steering Committee: Republic and Canton of Geneva, CHE, 2018; WWW ’18, p. 187–196. https://doi.org/10.1145/3178876.3185996.
[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). IEEE, 2015, pp. 38–44.

[27] Fluss, R.; Faraggi, D.; Reiser, B. Estimation of the Youden Index and its associated cutoff point. Biometrical Journal 2005, 47, 458–472. [https://doi.org/10.1002/BiM.200410135]

[28] Graabæk, S.G.; Ancker, E.V.; Christensen, A.L.; Fugl, A.R. An Experimental Comparison of Anomaly Detection Methods for Collaborative Robot Manipulators 2022. [https://doi.org/10.38227/techrxiv.19006643.v1]

[29] Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation-Based Anomaly Detection. ACM Transactions on Knowledge Discovery from Data 2012, 6, 3:1–3:39. [https://doi.org/10.1145/2133360.2133363]

[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); IEEE: Honolulu, HI, USA, 2020; pp. 1–9. [https://doi.org/10.1109/ICCCN49398.2020.9209704]

[31] MacGregor, J. Statistical Process Control of Multivariate Processes. IFAC Proceedings Volumes 1994, 27, 427–437. IFAC Symposium on Advanced Control of Chemical Processes, Kyoto, Japan, 25-27 May 1994. [https://doi.org/doi.org/https://doi.org/10.1016/S1474-6670(17)48188-2]

[32] Yankov, D.; Keogh, E.; Rebibapraguda, 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); IEEE: Omaha, NE, USA, 2007; pp. 381–390. [https://doi.org/10.1109/ICDM.2007.61]

[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, ICIPRAM 2019, Prague, Czech Republic, February 19-21, 2019. Revised Selected Papers; Springer-Verlag: Berlin, Heidelberg, 2019; p. 95–118. [https://doi.org/10.1007/978-3-030-40014-9_5]

[34] Rumelhart, D.E.; McClelland, J.L. Learning Internal Representations by Error Propagation; MIT Press, 1987; pp. 318–362.

[35] Baldi, P. Autoencoders, Unsupervised Learning, and Deep Architectures. In Proceedings of the Proceedings of ICML Workshop on Unsupervised and Transfer Learning; Guyon, I.; Dror, G.; Lemaire, V.; Taylor, G.; Silver, D., Eds.; PMLR: Bellevue, Washington, USA, 2012; Vol. 27, Proceedings of Machine Learning Research, pp. 37–49.

[36] Bank, D.; Koenigstein, N.; Giryes, R. Autoencoders 2020. [arXiv:cs.LG/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; Guyon, I.; Luxburg, U.V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R., Eds. Curran Associates, Inc., 2017, Vol. 30.

[38] Wu, R.; Keogh, E. UCR Anomaly archive. [https://www.cs.ucr.edu/~eamonn/time_series_data_2018/UCR_TimeSeriesAnomalyDatasets2021.zip], 2021.

[39] Laptev, N.; Amizadeh, S.; Billawala, Y. S5-a labeled anomaly detection dataset, version 1.0 (16M), 2015.

[40] Hundman, K.; Constantinou, V.; Laporte, C.; Colwell, I.; Soderstrom, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In Proceedings of the Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 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. Computational and mathematical methods in medicine 2017, 2017.

[42] Wu, R.; Keogh, E. UCR AnomalyDataSets.pptx, Suplemental mmaterial to the UCR Anomaly Archive. [https://www.cs.ucr.edu/~eamonn/time_series_data_2018/UCR_TimeSeriesAnomalyDatasets2021.zip], 2021.
[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., 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 Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1067–1075.

[45] Boniol, P.; Palpanas, T.; Meftah, M.; Remy, E. Graphan: Graph-based subsequence anomaly detection. Proceedings of the VLDB Endowment 2020, 13, 2941–2944.

[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 Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; Association for Computing Machinery: New York, NY, USA, 2019; KDD ’19, p. 2828–2837. https://doi.org/10.1145/3292500.3330672

[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); IEEE Computer Society: Los Alamitos, CA, USA, 2020; pp. 841–850. https://doi.org/10.1109/ICDM50108.2020.00093

[48] Bradley, A.P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern recognition 1997, 30, 1145–1159.
F Supplemental Materials

F.1 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
- ICP: Intracranial Pressure
- MDI: Maximally Divergent Intervals
- POT: Peak Over Threshold
- RRCF: Robust Random Cut Forest
- TranAD: Transformer Network for Anomaly Detection

F.2

| Anomaly Type | Description | Example |
|--------------|-------------|---------|
| Amplitude Change | Amplitude of the signal increased or decreased within a section. | ![Example](image1) |
| Flat | Flat section was added. | ![Example](image2) |
| Frequency Change | The cycle length was modified within a section. | ![Example](image3) |
| Local Drop | A drop was added, which is shallower than the minimal value of the time series. | ![Example](image4) |
| Local Peak | A peak was added, which is lower than the maximal value of the time series. | ![Example](image5) |
| Missing Drop | A drop was removed. | ![Example](image6) |
| Missing Peak | A peak was removed. | ![Example](image7) |
| Noise | Noise was added to a section. | ![Example](image8) |
| Outlier | A global outlier. | ![Example](image9) |
| Anomaly Type   | Description                                                      | Example                  |
|---------------|------------------------------------------------------------------|--------------------------|
| Reversed      | Cycle(s) got reversed.                                           | ![Example](image)        |
| Sampling Rate | The sampling rate of the signal was increased or decreased in a section. | ![Example](image)        |
| Signal Shift  | A section was shifted up or down.                                | ![Example](image)        |
| Smoothed Increase | A otherwise steep increase was smoothed, increasing the number of individual values in this section. | ![Example](image)        |
| Steep Increase | A otherwise smooth increase was made steep, reducing the number of individual values within this section. | ![Example](image)        |
| Time Shift    | Increasing the pause between two peaks.                          | ![Example](image)        |
| Time Warping  | Moving the cycle peak without changing the cycle length.         | ![Example](image)        |
| Unusual Pattern | Replacement of one or more cycle(s) with a different pattern. | ![Example](image)        |

Table A3: Description of the annotated anomaly types in the UCR Anomaly Archive dataset.
### Table A4: Tuned and not tuned parameters used in our experiments. All other parameters within the methods have been kept to their default values.

| Method | 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                    | TBA (30+20 oder so) | no |
|        | batch size                | 32    | no     |
|        | latent space dimension    | 128   | yes    |
|        | learning rate             | 0.003 | yes    |
|        | weight decay              | $10^{-5}$ | no |
|        | n_blocks                  | 2     | yes    |
| MDI    | $L_{\text{min}}$          | 75    | no     |
|        | $L_{\text{max}}$          | 125   | no     |
| MERLIN | $L_{\text{min}}$          | 75    | no     |
|        | $L_{\text{max}}$          | 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    |
Figure A13: 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.