Machine learning and deep learning approaches for multivariate time series prediction and anomaly detection

Thill, M.

Chapter 2

Fundamentals

2.1 Online Anomaly Detection

Especially in an online setting, which is often a requirement in practice, anomaly detection is usually a more difficult task than in the offline case, for several reasons: an online algorithm cannot process a time series several times (e.g., using batch learning approaches). Instead, the algorithm has to gradually update its knowledge with every new data instance arriving. Every instance is processed at maximum once. Also, often stability issues, such as numerical overflows or diverging behavior, arise when models have to be learned incrementally, with one example at a time, especially if the algorithm operates on an infinite data stream. Furthermore, back-dating of anomalies is usually not possible in an online setup and, hence, anomalous events have to be detected instantaneously. This would also require the algorithm to be computationally feasible in order to run in real-time. On the other side, online algorithms can also have advantages. The most notable advantage might be the adaptivity capabilities, which allow online algorithms to learn in dynamically changing environments and to adapt to concept drifts or concept changes. In general, online learning approaches have several properties that are convenient for many learning tasks, especially if streaming data is involved:

- Online approaches can incrementally adapt their models with each new example. In contrast to offline methods, it is not necessary to keep the whole dataset in the memory.
- Online models can be used simultaneously to make predictions and learn.
- Concept drift adaptation: While offline machine learning approachescannot track concept drifts, online learning algorithms generally can adapt to new situations. Also, in situations where the data generating process adjusts its behavior to the model (e.g., spammers who change their e-mails according to the spam filter's current configuration), online-adaptable models can be preferable.

Next to the mentioned advantages, online learning methods are also challenging in several aspects:

- As mentioned before, one main issue that is often faced is a trade-off between the online learning algorithm's adaptation speed and its stability. For example, a fast learning algorithm can quickly adapt to new concepts in the data and might become

unstable easily, e.g., in the presence of random noise. On the other side, an algorithm that adapts itself less aggressively could be less sensitive to noise but might not fully trackconcept changes in the data. This problem is also known as the stability-plasticity dilemma [21].

- Many of the existing machine learning algorithms are inherently offline and might require significant adjustments to achieve an online formulation.
- Additional complications arise if the rate of change and frequency of the concept drifts varies over time.

2.2 Scoring Process for Anomaly Detection Tasks

In order to compare the performance of the different algorithms on the described benchmarks (Section 2.3), suitable performance metrics are required. Similarly to binary classification tasks, an algorithm can classify every instance in a time series either as normal or anomalous. Subsequently, one can compare the classes predicted by the algorithm to the ground truth labels. For time series taken from real-world problems, the anomalies are often labeled manually. This might lead to some inaccuracies in the labeling process since experts (e.g., based on knowledge in the domain) might interpret certain instances differently. For artificial time series, the labeling process can generally be automatized and is mostly very accurate. The exact scoring procedure using anomaly windows is described in the following.

2.2.1 Anomaly Score & Anomaly Threshold

In practice, there are two approaches to report anomalous behavior: 1. The algorithms directly assign a binary label to each data point or, 2. they output a so-called *anomaly score* for each data point, as an intermediate indicator for the degree of abnormality of the corresponding point. This anomaly score is a continuous (usually between 0 and 1) value and reflects the algorithm's predicted probability of a data point being anomalous. Low values are an indicator of nominal behavior and high values indicate that an unusual situation has been observed. The first approach has the disadvantage that there is no possibility to control the sensitivity of the algorithm since labels are directly assigned to each point. The second approach allows to specify a so-called *anomaly threshold*, with which a sensitivity level can be set. Points with a score above the threshold are classified as anomalous, all other points are classified as nominal. By reducing the threshold the algorithm will get more sensitive and detect more anomalies. However, at the same more false alarms will be issued. Finding a suitable threshold is heavily dependent on on the application. For example, in critical medical health monitoring systems the anomaly detection algorithms might have to be more sensitive (having a relatively low threshold).

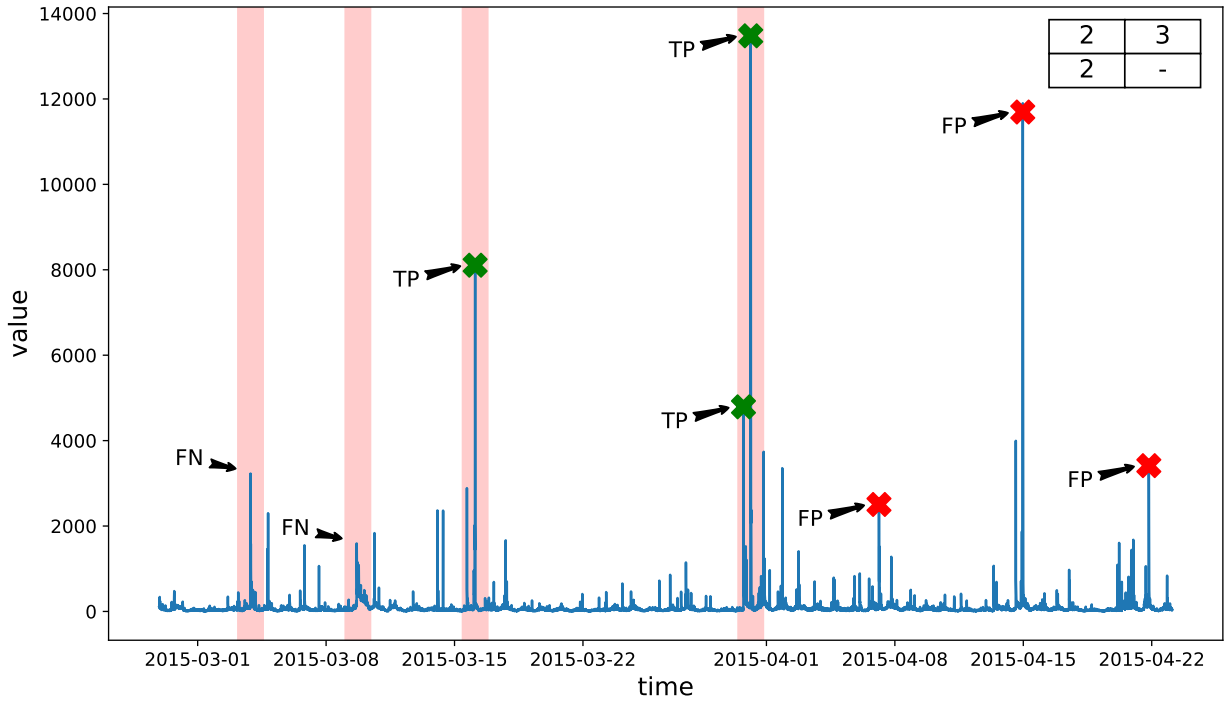

Figure 2.1: A time series taken from the Numenta Anomaly Benchmark (NAB) with in total 4 anomalies (anomaly windows indicated by red shaded vertical bars). In this plot, we illustrate the construction of the confusion matrix (Section 2.2.2) for some sample detections. The confusion matrix for this particular example is depicted in the top-right corner. The last anomaly window contains two detections, however, only the first detection will be counted as TP. In this example, we do not show TN in the confusion matrix since TN usually is of limited interest for anomaly detection tasks.

2.2.2 Anomaly Windows & Confusion Matrix

Usually, anomalies in time series are not single points and certain instances of time. Most anomalies span a longer range in the time series. In many cases, it is also not entirely clear, where exactly an anomaly begins and where it ends. A useful approach used in practice is to simply specify a so-called anomaly window of appropriate length, which is centered around the ground truth label(s). In order to correctly detect an anomaly, an algorithm would have to classify at least one point within the anomaly window as anomalous. Detections that fall into the anomaly windows are considered as true positives (TP). However, only the first true positive in each window is counted. The remaining detections in the same window are ignored. If an algorithm fails to flag any instance inside an anomaly window as anomalous, this will be considered as one false negative (FN). All detections outside the window are considered as false positives (FP). All other points, which are not marked as anomalous, are considered as true-negatives (TN). However, TNs play only a minor role in the evaluation of the algorithms, as they represent the vast majority of all points. In some cases, especially

for very long time series (such as ECG data), we will permit to group a small range of false positives (e.g., if they span less than one heart beat) into a single false positive, in order to prevent situations where false positives will dominate the scoring process. We will use the anomaly windows throughout this thesis to denote anomalies. Table 2.1 summarizes the construction of the confusion matrix.

Table 2.1: Confusion matrix for time series anomaly detection tasks.

		Actual	
		Anom.	Norm.
Predicted	Anom.	TP	FP
	Norm.	FN	TN

The scoring process using anomaly windows is illustrated in Figure 2.1.

2.2.3 Algorithm Performance Measures

In our setup, the real anomaly labels are not passed to the algorithms (i.e., the task is treated as an unsupervised learning task like in a real-world application setting). Hence, each anomaly detection algorithm has to learn the normal structure of a time series and has to be able to identify anomalies among the observed time series values. The ground truth labels are used only for evaluation purposes, or to obtain anomaly thresholds for all algorithms (for example, using the EAC-criterion, as described below), which allow a fair comparison.

From the aforementioned quantities TP, FP, FN, and TN, additional metrics can be derived: Precision (also called positive predictive value) takes into account how many false-positive errors were produced and is defined as the number of correctly identified anomalies divided by the number of all predicted anomalies:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.1}$$

Recall (also called sensitivity) measures the detection rate of true anomalies and is defined as the number of correctly identified anomalies divided by the overall number of true anomalies:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.2}$$

Both measures are independent of TN, which usually only plays a minor role in anomaly detection tasks due to the typically highly unbalanced class labels. Depending on the anomaly threshold of the algorithm, there is a tradeoff between precision and recall. While precision and recall are important metrics, it is not always possible to compare two algorithms

based on these measures. For example, one algorithm might achieve a high recall but a low precision and another algorithm, vice versa, a high precision and low recall. In such situations, a direct comparison is infeasible. In order to fairly assess algorithms based on these two objectives (precision and recall), the algorithms are typically compared based on the so-called equal accuracy (EAC) where precision and recall are approximately equal. We will use EAC as one performance indicator. Another possibility is to generate a multi-objective plot, where precision and recall are plotted against each other for a large range of anomaly thresholds.

The F_1-score combines precision and recall into a single value and is defined as their harmonic mean:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{2.3}$$

Since the F_1-score considers both, precision and recall, and weights both equally, the F_1-score will be the most important measure during our later experiments. A perfect algorithm would achieve a precision and recall of 1, which would also result in $F_1 = 1$. Two additional metrics, which are used less commonly for anomaly detection (since they are based on TN) are the false-positive rate (FPR) and the positive likelihood ratio (PLR):

$$FPR = \frac{FP}{FP + TN}, \tag{2.4}$$

$$PLR = \frac{\text{recall}}{FPR}. \tag{2.5}$$

Finally, other relevant performance indicators of an algorithm are its computation time (for training and inference) and its memory requirements. Usually, we will report average computation times on a certain computer and the number of trainable parameters of the models.

In the literature, another commonly reported metric is the area under the ROC (Receiver Operation Characteristic) curve, in short AUC-ROC, which summarizes the performance of an algorithm when the anomaly threshold is varied. However, we do not use AUC-ROC for our scoring process, since it is not particularly well suited for imbalanced data sets (ROC curves plot recall against FPR). In such cases, Precision-Recall (PR) curves, as used by us, give better information. Simiarly to AUC-ROC, it is also possible to compute the AUC-PR, however, we usually plot the PR curve, which is more intuitive than a single metric.

2.3 Time Series Anomaly Detection Benchmarks

In the following, we describe several time series anomaly detection benchmarks that we use in this work for benchmarking purposes. All benchmarks are publicly available and serve as a common reference point for comparing the investigated algorithms.

2.3.1 Yahoo's Webscope S5 Dataset

The Webscope S5 dataset is a relatively large anomaly detection benchmark which is publicly available [87]. It consists of 367 time series (with overall 572,966 data points). On average, each time series has approximately 1,500 instances. The Webscope S5 benchmark is split again into the 4 datasets A1, A2, A3 and A4 containing 67, 100, 100 and 100 time series. While class A1 has real data from computational services, classes A2, A3, and A4 contain synthetic anomaly data with increasing complexity. Example plots are shown in Figs. 2.2–2.5. The ground truth anomaly information is available for all time series. For Yahoo's Webscope S5 data we place anomaly windows of size 10 around the labeled anomalies. In our setup, the ground truth anomaly labels are not provided to the anomaly detection algorithms, which have to learn to separate anomalies from normal behavior in an unsupervised fashion.

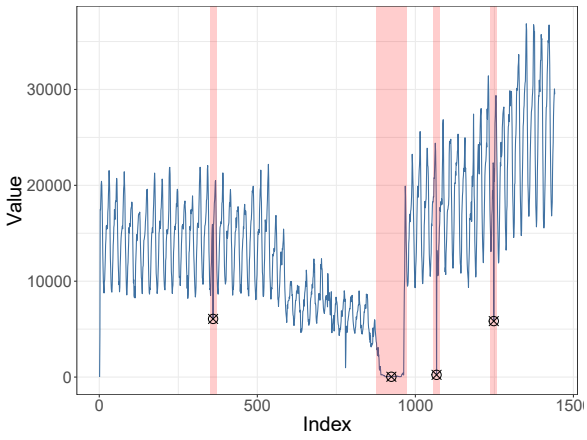

Figure 2.2: Example plot for the A1 data. Black crosses mark the true anomalies and the red vertical bars indicate their anomaly windows. The time series of the A1 data were taken from real world applications and anomalies were manually labeled.

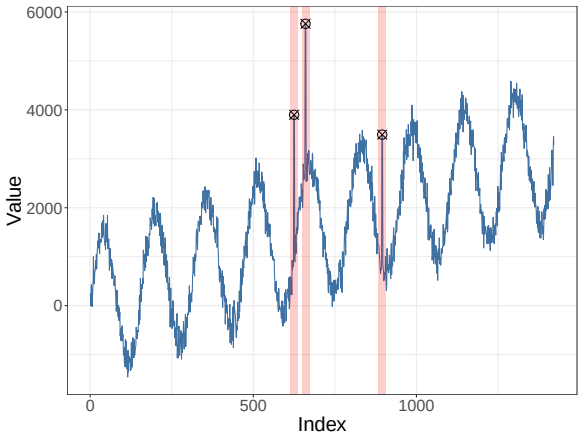

Figure 2.3: Example plot for the A2 data. The time series of the A2 data were generated synthetically. The time series include a trend, a seasonal (periodic) component and noise. The anomalies were added at random instances.

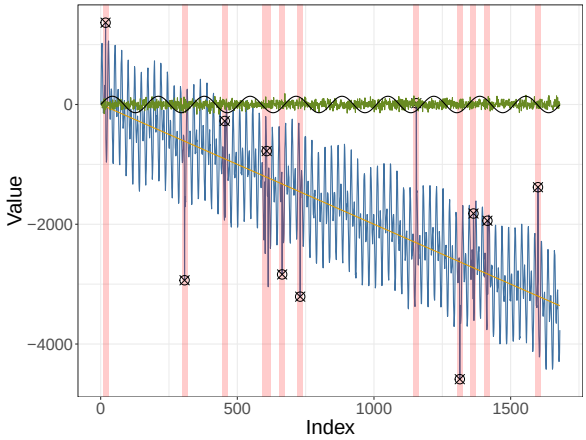

Figure 2.4: Example plot for the A3 data. The time series of the A3 data (blue) were generated synthetically. The time series include a trend, three seasonal (periodic) components and noise (green). The anomalies were added at random instances.

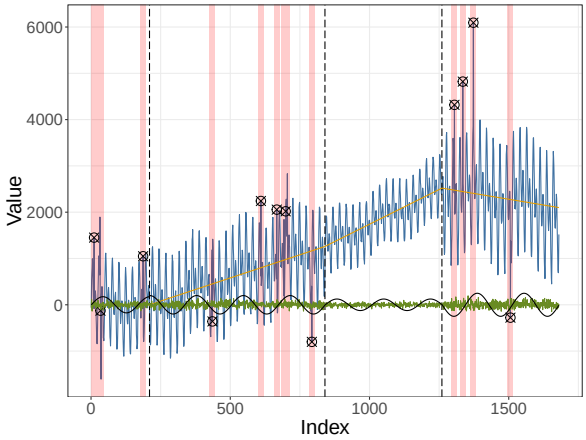

Figure 2.5: Example plot for the A4 data (blue). Similar to the A3 data. Additionally, change point anomalies are introduced (indicated by the vertical dashed lines) in which the characteristics (such as slope or frequency) of the individual components are changed.

2.3.2 The Numenta Anomaly Benchmark

The Numenta Anomaly Benchmark (NAB) [89] is a publicly available dataset that consists of 58 time series with in total 365,558 data points – the shortest series containing 1,127, the longest containing 22,695 and the average series containing approx. 6,300 instances. The majority of the time series are real-world data coming from application areas such as server monitoring, network utilization, sensor readings from industry and social media statistics [89]; 11 time series were generated artificially, from which 5 are anomaly-free. In total, over all 58 time series, 115 anomalies were labeled, most of which were identified and documented manually by a human expert. The ground truth anomaly labels are available for all considered time series. An example taken from NAB is shown in Figure 2.1.

2.3.3 The MIT-BIH Arrhythmia Database

The MIT-BIH arrhythmia database [52, 112, 113] contains two-channel electrocardiogram (ECG) signals of 48 patients of the Beth Israel Hospital (BIH) Arrhythmia Laboratory. Each signal was recorded with a sampling frequency of 360 Hz and has a length of approximately half an hour, which corresponds to 650 000 data points each. The two channels recorded are the modified limb lead II (MLII) and a modified lower lead V1, in a few cases V5. Each recording contains on average 2160 ± 365 heartbeats, and in total, there are 54 087 heartbeats. The individual signals have a quasi-periodic behavior, with differing heartbeat patterns for each subject. To understand the complexity of anomaly detection in quasiperiodic ECG data, we show in Fig 2.6 an anomaly example. The normal signal is quasiperiodic (peak height and other small details may differ from period to period to some extent). The anomalous region has a very similar peak, yet it comes earlier than expected. The model which predicts anomalies thus has to learn not only the shape of the peak but when to expect it. This requires to process long-range information since the local neighborhood of the anomalous peak looks absolutely normal.

There are many different events in all ECG time series, which were labeled by human experts. The whole list of heartbeat annotations is found in [112]. InTable 2.2, we summarize nine event classes, which are considered as anomalous.
The unsupervised anomaly detection approaches investigated in this thesis rest on the assumption that most time-series data is normal (nominal) and that anomalous events are relatively seldom.

Since there are time series in the database which contain many events (in several cases, the vast majority of heart beats are anomalous events) we limit ourselves to time series with a small to moderately number of events.

ECG-13 For our initial experiments in Chapter 6, we consider only those time series with 50 or fewer events. Overall, 13 out of 48 time series will be used. We call this data set the *ECG-13* data. The selected time series contain 130 anomalous events from 6 anomaly classes, which are listed in Table 2.2.

ECG-25 Later, in Chapter 7, we will extend the benchmark and select those 25 time series with 250 or fewer anomalous events and call this data set the *ECG-25* data. The ECG-25 data contains 721 events. Additionally, for the ECG-25 data, we also perform some simple pre-processing steps, which are commonly done in practice: Since several of the raw 2-dimensional ECG signals contain a lot of noise and exhibit simple baseline drifts, each signal is filtered with a bandpass filter parameterized with the cutoff frequencies of 2 and 20 Hz. These are values which are commonly used for the processing of ECG signals in practice (cf. Pan-Tompkins algorithm [127]). This bandpass filter removes most of the high-frequency noise in the signal as well as drifts in the baseline. In order to reduce the training time of the models, we down-sample (using an anti-aliasing filter) each ECG signal in ECG-25 by a factor of $n_{samp} = 5$. This shortens the length of the ECG signal from originally 650 000 time-steps to just $T = 130\,000$ time-steps without losing too much information from the signal. The down-sampling was done to speed-up the training time of the individual algorithms.

For the ECG-13 and ECG-25 data, we normalize each input time series to zero mean and unit variance before generating the training samples. In the MIT-BIH benchmark, only the R-peaks (the tops of the main spikes on the ECG line) of the QRS complex are labeled (either as normal or with the corresponding arrhythmia type). However, it is not possible to locate the position of an anomaly in exactly one point of the ECG signal.

Thus, as for the other benchmarks, we specify an anomaly window for each anomaly. The anomaly window is centered around the given label and contains 400 (80 after downsampling the time series in ECG-25) data points before and after the label, which corresponds to approximately 2 seconds (or roughly one heartbeat before and after the label). A more detailed database description can be found in [113].

Table 2.2: Anomaly types in the ECG-13 and ECG-25 data considered for the experiments. The descriptions are taken from [112]. The second column shows the overall number of the various anomaly types for ECG-13 and the third column for ECG-25 considered ECG signals.

Code	# ECG-13	# ECG-25	Description
a	6	11	Aberrated atrial premature beat
A	44	235	Atrial premature beat
e	–	10	Atrial escape beat
f	–	22	Fusion of paced and normal beat
F	2	25	Fusion of ventricular and normal beat
J	–	3	Nodal (junctional) premature beat
V	53	374	Premature ventricular contraction
x	8	19	Non-conducted P-wave (blocked APC)
|	17	22	Isolated QRS-like artifact
Σ	130	721	

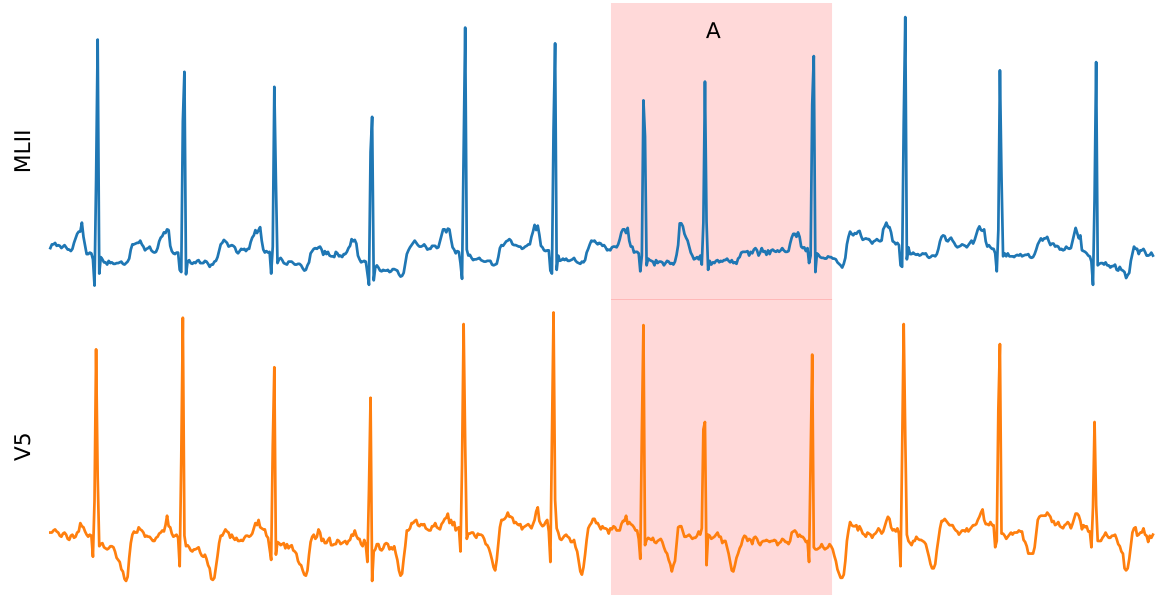

Figure 2.6: Example ECG signal from the MIT-BIH Arrhythmia database. This is just one type of anomaly out of the set of 9 different anomaly types (to be discussed later in more detail, see Table 2.2).

2.3.4 The Mackey-Glass Anomaly Benchmark

During our work on a paper [166], we developed a non-trivial synthetic benchmark, named Mackey-Glass anomaly benchmark (MGAB) [167]. The motivation for this benchmark was to be able to generate fairly long synthetic time series (with 100 000 points) with well-defined, yet non-trivial anomalies, in order to investigate and evaluate larger deep learning anomaly detection models. MGAB allows creating an unlimited amount of quasi-periodic time series data with steerable difficulty by simply adjusting a few parameters of the MGAB generation process (e.g. time delay, smoothness parameters). In Figure 2.7, an excerpt of ab MGAB time series is shown.

Mackey-Glass (MG) time series are known to exhibit chaotic behavior under certain conditions. MGAB contains 10 MG time series of length $T = 10^5$. Into each time series 10 anomalies are inserted with a procedure described in Section 2.3.4.2. In contrast to other synthetic benchmarks, the introduced anomalies are for the human eye nearly invisible in the context of the normal (chaotic) behavior. Overall, we generate 100 anomalies in 10^6 time series points. The benchmark data and the detailed procedure for generating these and similar benchmark data are publicly available at GitHub [167].[1]

[1]GitHub repository: `https://github.com/MarkusThill/MGAB/`

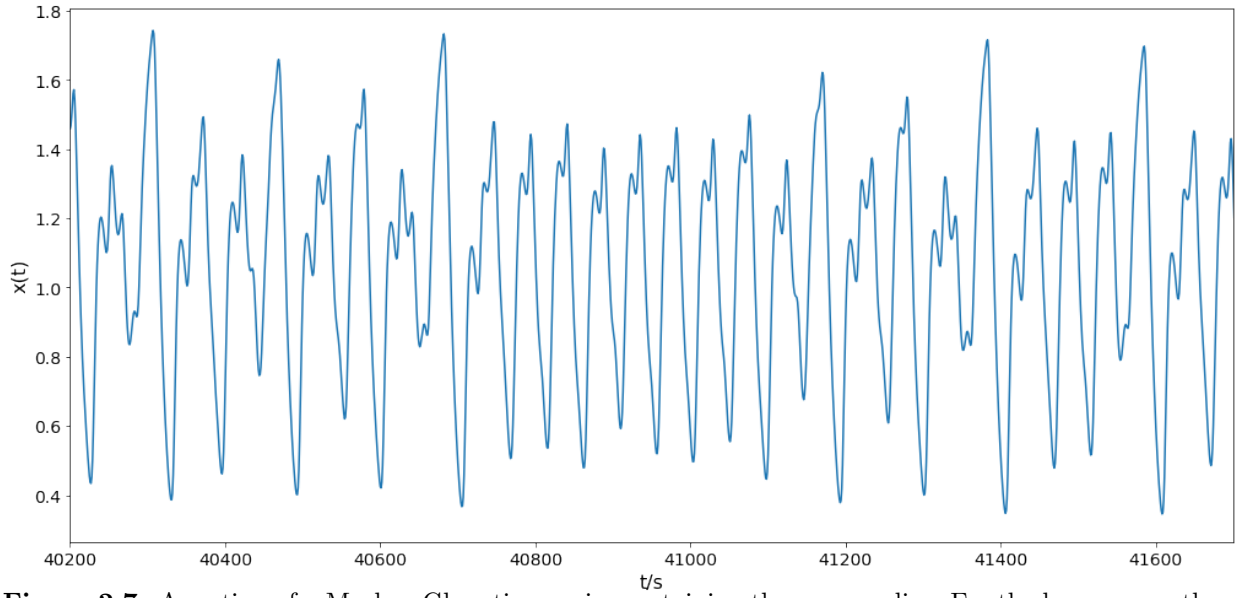

Figure 2.7: A section of a Mackey-Glass time series containing three anomalies. For the human eye, these anomalies might be hard to spot. Fig. 2.10 unveils the location of the anomalies.

2.3.4.1 The Mackey-Glass Equation

The Mackey-Glass equation [106] is a so-called non-linear time delay differential equation (DDE), which is defined as

$$\frac{dx}{dt} = \beta \cdot \frac{x(t-\tau)}{1+x(t-\tau)^n} - \gamma x(t), \tag{2.6}$$

where τ, n, β and γ are real numbers. Additionally, an initial condition and a past (history) are required in order to integrate the equation. Commonly, initial condition and past are chosen to be constant with $x(t \leq 0) = h$. Depending on the parameterization of Eq. (2.6), the resulting time series develops various patterns of chaotic and periodic dynamics. One possibility to visualize the chaotic behavior of a Mackey-Glass attractor is a time delay embedding graph, which plots the delayed series $x(t-\tau)$ against the actual Mackey-Glass time series $x(t)$. An example is shown in Fig. 2.8. In order to generate all Mackey-Glass time series used in this work, we use the JiTCDDE package for Python [11], which is designed to accurately integrate delay differential equations (DDEs).

2.3.4.2 Generating Anomalies in Mackey-Glass Time Series

In order to create the Mackey-Glass Anomaly Benchmark, we first generate a sufficiently long time series having a dimension of $d = 1$ using the JiTCDDE [11] solver with the parameters $\tau = 18, n = 10, \beta = 0.25, \gamma = 0.1, h = 0.9$. The integration step size is set

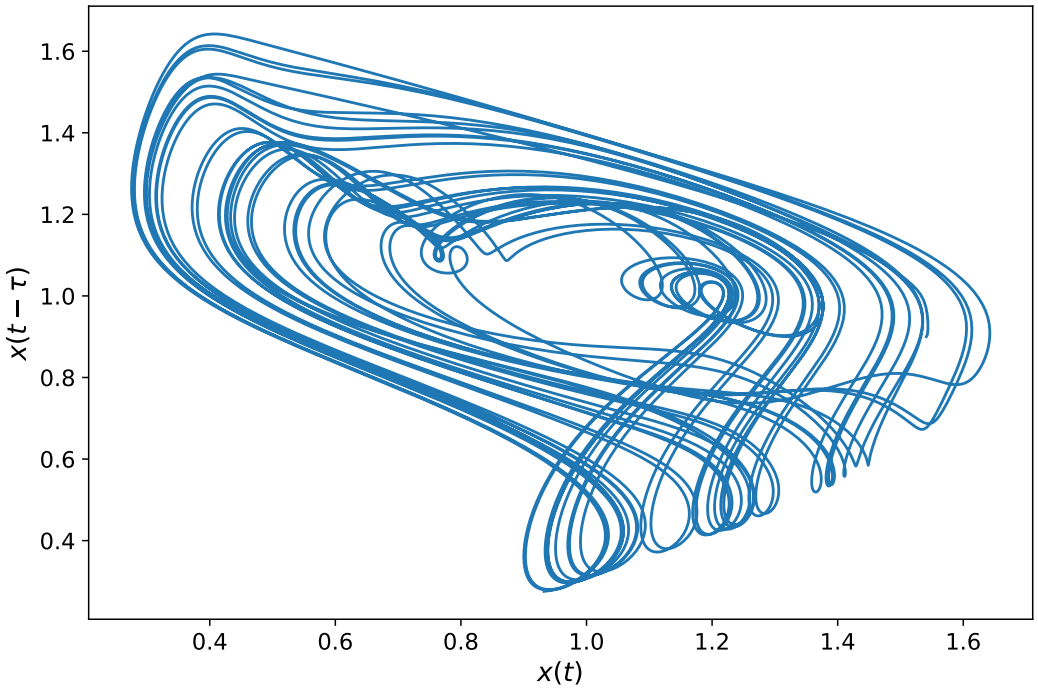

Figure 2.8: Time delay embedding of the Mackey-Glass attractor, which is described in Section 2.3.4.2, for $\tau = 10.0$, $\beta = .25$, $\gamma = .1$, $h = .9$ (history).

to 1. The maximal Lyapunov exponent (MLE) of $\lambda_{mle} = 0.0061 \pm 0.0002$ suggests that the generated time series is (mildly) chaotic. Subsequently, we split this series into ten same-sized individual time series and insert 10 anomalies into each time series.

The basic idea is to split the time series in two places and remove the segment in between. If the two split points are carefully chosen, the manipulation will be hardly visible later, without the knowledge of the dynamics of the MG series. This procedure is repeated to insert 10 anomalies in each time series. It is ensured that a new anomaly does not remove a previously inserted anomaly. Finally, the 10 time series are truncated to a length of $T = 10^5$. In order to make the benchmark data more similar to real-world data, we add uniform random noise to each of the 10 time series after inserting the anomalies. For this purpose, we sample 10^5 values uniformly from $[-0.01, 0.01]$. Overall, 96% of each time series are normal data points and 4% anomalous.

2.4 Time Series Characteristics

Based on certain characteristics that time series may or may not exhibit, some algorithms might be better suited for anomaly detection tasks than others. In the following, we list

several characteristics that we found important in deciding which algorithm to choose. In Section 8.1, we will summarize how well the anomaly detection algorithms investigated in this work are suited for particular time series, after classifying them according to the following characteristics.

Small data sets Some algorithms require many data for their training process. Especially DL approaches usually require significant amounts of data for training. If no suitable pre-trained model is available, it is mostly not recommended to train a DL model on small data sets. Algorithms with less trainable parameters are more suited for these cases.

Multivariate time series Many time series in practice are multivariate. However, not all algorithms are applicable to time series with more than one dimension.

Anomalies appear at different time scales In many cases, it is necessary to analyze time series at different time scales in order to detect short-term and longer-term anomalies.

Long-term correlations Some problems require that the anomaly detection algorithm learns the time series's behavior over very long ranges of time to detect anomalous patterns reliably (see Section 2.3.3 for an example). Algorithms with only a limited temporal memory or a small temporal receptive field are only partially or not suited for these types of problems.

Unpredictable time series Prediction-based anomaly detection algorithms assume that future instances of a time series can be predicted (forecasted) to some extent. However, some time series in practice are inherently unpredictable, although they show nominal behavior, which can be learned with other methods. A common approach to handle unpredictable time series is to use reconstruction-based algorithms (based on autoencoders or similar architectures).

Weak non-stationary behavior In many time series recorded under real-world conditions (such as ECG signals), one can observe weak forms of non-stationary behavior such as baseline wandering or changes (gradual or sudden) in the signal noise or quality.

Strong non-stationary behavior For strong non-stationary behavior, online-adaptable algorithms are required which can adapt to new concepts in the data. Examples for such behavior are (sudden) notable changes in the trend, in the signal frequency or in the recurring signal patterns.

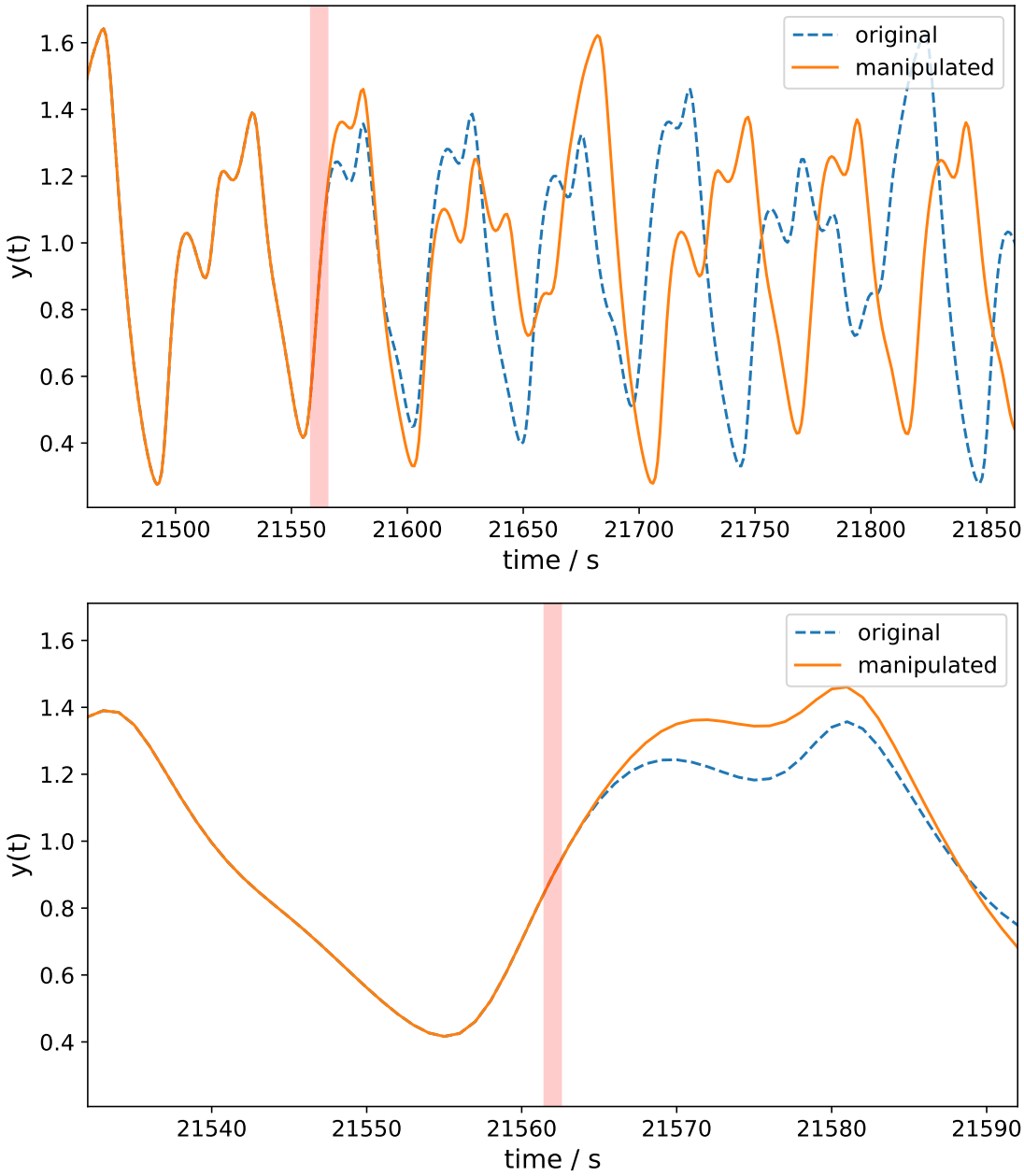

Figure 2.9: Top: Example for the creation of a Mackey-Glass time series with a temporal anomaly. The original time series (dashed line) is manipulated in such a way, that a segment is removed and the two remaining ends are joined together. In this example, the interval $[21562, 21703]$ is removed from the original curve. The resulting manipulated time series (solid line) has a smooth point of connection, but significantly differs from the original. Bottom: Zoomed-In. The red shaded area indicates the position where the anomaly was inserted.

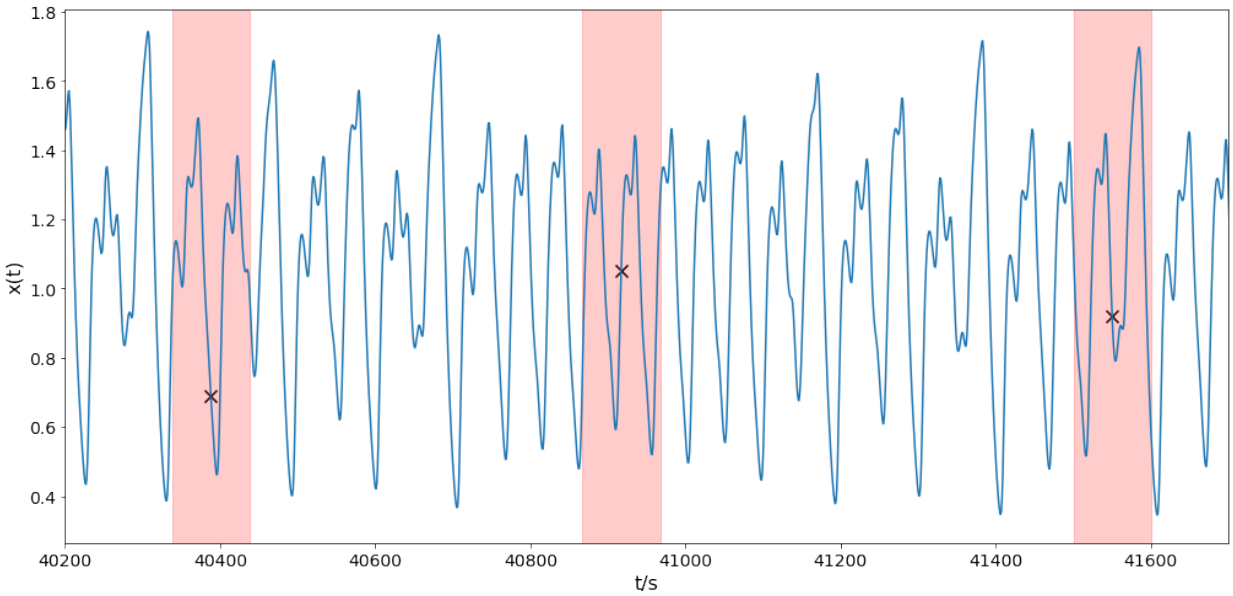

Figure 2.10: This graph shows the same section of a Mackey-Glass time series as Fig. 2.7, but now reveals the location of the anomalies in the time series. The anomalies are at $t_1 = 40388$, $t_2 = 40917$ and $t_3 = 41550$. The positions are indicated by the black crosses in the plot.