ANOMALY DETECTION IN UNIVARIATE TIME-SERIES: A SURVEY ON THE STATE-OF-THE-ART

Mohammad Braei
Department of Computer Science
Technische Universität Darmstadt
mohammad.braei@stud.tu-darmstadt.de

Dr.-Ing. Sebastian Wagner
Telecooperation Group
Technische Universität Darmstadt
s.wagner@tk.tu-darmstadt.de

April 2, 2020

ABSTRACT

Anomaly detection for time-series data has been an important research field for a long time. Seminal work on anomaly detection methods has been focussing on statistical approaches. In recent years an increasing number of machine learning algorithms have been developed to detect anomalies on time-series. Subsequently, researchers tried to improve these techniques using (deep) neural networks. In the light of the increasing number of anomaly detection methods, the body of research lacks a broad comparative evaluation of statistical, machine learning and deep learning methods. This paper studies 20 univariate anomaly detection methods from the all three categories. The evaluation is conducted on publicly available datasets, which serve as benchmarks for time-series anomaly detection. By analyzing the accuracy of each method as well as the computation time of the algorithms, we provide a thorough insight about the performance of these anomaly detection approaches, alongside some general notion of which method is suited for a certain type of data.

1 Introduction

Detecting anomalies has been a research topic for a long time. In a world of digitization, the amount of data transferred exceeds the human ability to study it manually. Hence, automated data analysis becomes a necessity. One of the most important data analysis tasks is the detection of anomalies in data. Anomalies are data points which deviate from the normal distribution of the whole dataset, and anomaly detection is the technique to find them.
The impact of an anomaly is domain-dependent. In a dataset of network activities, an anomaly can imply an intrusion attack. An anomaly in a financial transaction can hint on financial fraud, anomalies in medical images can be caused by diseases. Other objectives of anomaly detection are industrial damage detection, data leak prevention, identifying security vulnerabilities or military surveillance.
Anomaly detection methods are specific to the type data. For instance, the algorithms used to detect anomalies in images are different to the approaches used on data streams. This paper focuses on methods for anomaly detection in time-series data. Anomaly detection on time-series has been a long-time topic of interest. In 1979, Tukey [88] proposed a statistical approach to detect anomalies on time-series. Chang et al. [18] proposed to use the likelihood ratio test (LRT) to detect anomalies on time-series.
With the ever-growing computational power in recent decades, machine learning approaches increased in popularity for data science tasks such as classification and pattern detection. Therefore, many researchers started to use these machine learning methods to detect anomalies in time-series. For instance, clustering methods such as k-Means can be used to detect anomalous points in time-series data.

In the last decade, deep learning approaches achieved tremendous progress in computer vision tasks. This success motivated researchers to leverage these methods to detect anomalies. Various deep learning approaches such as Multi -Layer Perceptrons (MLPs), Convolution Neural Network (CNNs) and Long-Short Term Memory (LSTMs) were proposed as anomaly detection techniques. While there is a wide spectrum of anomaly detection approaches today, it becomes more and more difficult to keep track of all the techniques. As a matter of fact, it is not clear which of the three categories of detection methods, i.e., statistical approaches, machine learning approaches or deep learning approaches is

more appropriate to detect anomalies on time-series data. To the best of our knowledge, there is no study comparing approaches of these three categories in their accuracy and performance.

This paper presents a quantitative comparison of multiple approaches of each category. We select a wide range of methods to cover well-performing techniques of each class. One of the main contributions of this work is that it focuses on time-series data. In order to provide a reliable comparison, the methods are evaluated on multiple time-series datasets.
This paper is structured as follows:

1. **Basics:** In this section, the main concepts of anomaly detection on time-series, which are fundamental to the subject, are defined. These concepts are vital to understand the algorithms in the following sections.
2. **Selected Anomaly detection approaches for time series:** This section introduces different anomaly detection algorithms of the three main categories.
3. **Approach:** While first introducing related and similar works, here we explain how the evaluation of the different methods is carried out.
4. **Experiments:** As a preceding step to the evaluations, this section lists the setup of the experiments by listing all hyperparameters of the used algorithms.
5. **Results:** This section illustrates and discusses the evaluation results.
6. **Conclusion:** The last section concludes the paper while also providing an insight into future work.

2 Foundations

In this section we define basic concepts which are fundamental in the anomaly detection process.

2.1 Anomalies and outliers

There is no consent about the distinction of anomalies and outliers. On one hand, the following citation is mostly referenced to prove the equality of anomaly and outliers:

> "*Outliers are also referred to as abnormalities, discordants, deviants, or anomalies in the data mining and statistics literature.*" – Aggarwal [2]

On the other hand, there are definitions which regard outliers as a broader concept, which also includes noise in addition to anomalies [80]. Others consider outliers as corruption in data, while anomalies are irregular points, but with a specific pattern [35].
As we are evaluating time-series in this paper, we consider the two terms, outlier and anomaly, interchangeably.

The important point is to deliver a formal definition for the concept of anomaly. This is essential, because different definitions of anomalies imply varying methods to detect them. Thus, it is necessary to define the main characteristics of anomalies and highlight the boundaries by the definition. The most common definition of anomalies is the following:

> "*Anomalies are patterns in data that do not conform to a well defined notion of normal behavior.*" – Chandola et al. [17]

Chronologically, one of the first definitions was given by Grubbs [34]. He defined outliers in 1969 as:

> "*An outlying observation, or "outlier," is one that appears to deviate markedly from other members of the sample in which it occurs.*"

Ord [69] used the following definition:

> "*An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.*"

And finally Hawkins [39] defines outliers as follows:

> "*An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.*"

All these definitions highlight two main characteristics of anomalies:

- The distribution of the anomalies deviates remarkably from the general distribution of the data.
- The big majority of the dataset consists of normal data points. The anomalies form only a very small part of the dataset.

These two aspects are fundamental to the development of anomaly detection methods. Especially the second property prevents us from using common classification approaches that rely on balanced datasets, and enables us to use approaches like auto-encoders as a semi-supervised method to detect anomalies, which will be explained in the following sections. Thus, in this paper anomaly and outliers are defined as follows:

Definition 2.1. *An anomaly is an observation or a sequence of observations which deviates remarkably from the general distribution of data. The set of the anomalies form a very small part of the dataset.*

It is also important to distinguish between anomalies and noise. Noise can be a mislabeled example (class noise) or errors in the attributes of the data (attribute noise) [80] which are not of interest to data analysts [2]. While for instance, in a set of medical images an anomaly may show a tumor, noise is just a random variation of brightness and color information which is unwanted. Thus, noise is not of interest to the analyst, while an anomaly is.
Whereas the difference between anomalies and noise is highlighted here, it still remains a difficult task to differentiate between them in some sort of data. This is illustrated in Figures 1 and 2:

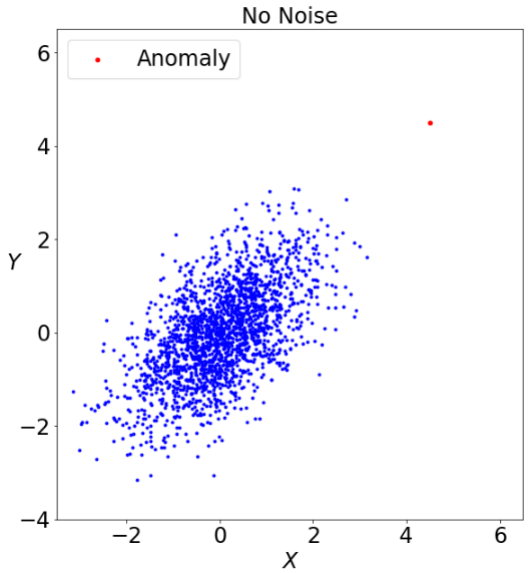

Figure 1: Data without noise.

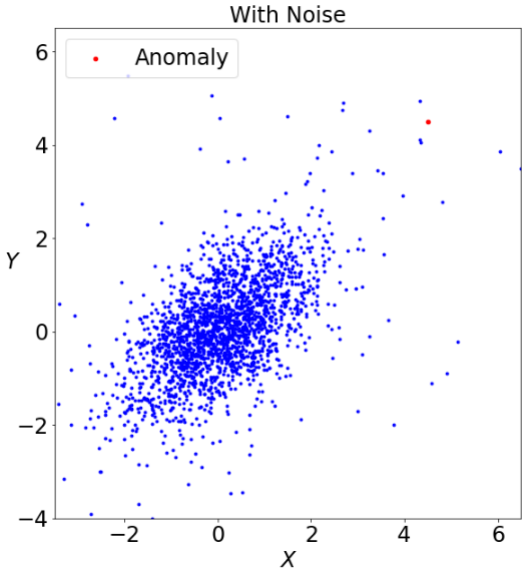

Figure 2: Data including noise.

In both figures the data distribution is the same. In Figure 1 the anomalous point marked in red seems to be obvious as it deviates significantly from the rest. But in Figure 2 it is difficult to distinguish the anomaly point from the other points in the sparse space. This example shows that the difficulty to distinguish between anomalies and noise depends on the dataset. Thus, a deep understanding of the dataset is necessary to do so.

Anomalies should also be distinguished from novelties [17, 74]. Novelty patterns are data points which have not been observed in the data yet. The difference to anomalies is, that novelties are considered *normal* after being detected once. For example, a new communication pattern to a server after implementing a new protocol. However, due to the fact that most methods used for novelty detection are also used for anomaly detection and vice versa, in this paper we treat them equally.

2.2 Types of Anomalies

Anomalies can appear in different forms. Commonly, three different types of anomalies exist:

1. **Point anomalies:** If a point deviates significantly from the rest of the data, it is considered a point anomaly. For example, a big credit transaction which differs from other transactions is a point anomaly. Hence, a point X_t is considered a point anomaly, if its value differs significantly from all the points in the interval $[X_{t-k}, X_{t+k}], k \in \mathbb{R}$ and k is sufficient large.

2. **Collective anomalies:** There are cases where individual points are not anomalous, but a sequence of points are labeled as an anomaly. For example, a bank customer withdraws $500 from her bank account every day of a week. Although withdrawing $500 occasionally is normal for the customer, a sequence of withdrawals is an anomalous behavior.

3. **Contextual anomalies:** Some points can be normal in a certain context, while detected as anomaly in another context: Having a daily temperature of 35° C in summer in Germany is normal, while the same temperature in winter is regarded as an anomaly.

Knowing a-priori, which kind of anomaly the data might contain, assists the data analyst to select the appropriate detection method. Some approaches that are able to detect point anomalies fail to identify collective or contextual anomalies altogether.

2.3 Stochastic Processes and Time-series

The data that is analyzed in this paper are time-series. Thus, it is fundamental to provide a definition of it. Primarily, another term has to be defined, which regularly occurs simultaneously with time series: *Stochastic Process*. Wei [91] provides a comprehensive definition:

Definition 2.2. *A stochastic process is a family of time indexed random variables $Z(\omega, t)$, where ω belongs to a sample space and t belongs to an index set.*

Hence, if t is fixed then Z is a random variable over the sample space.
A *time series* is a realization of a certain stochastic process. A formal distinct definition for time-series is as follows:

Definition 2.3. *A time-series is a sequence of observations taken by continuous measurements over time. Generally, the observations are picked up in equidistant time intervals:* $T = (t_0^d, t_1^d, ..., t_t^d), d \in \mathbb{N}_+, t \in \mathbb{N}$ *where d defines the dimension of time series*

A time-series can be a sequence of observations from one source, i.e., one sensor. In this case $d = 1$ and the series is univariate. If we collect information from more than one sensor, $d > 1$, we have a multivariate time-series. In this paper, we consider only **discrete** time-series, therefore $t \in \mathbb{N}$, perceived in equal time intervals.
Instances of univariate time-series are cash transactions or weather histories. Figure 3 shows a time plot of a sample time-series of an observed stock market value of an asset over five years.

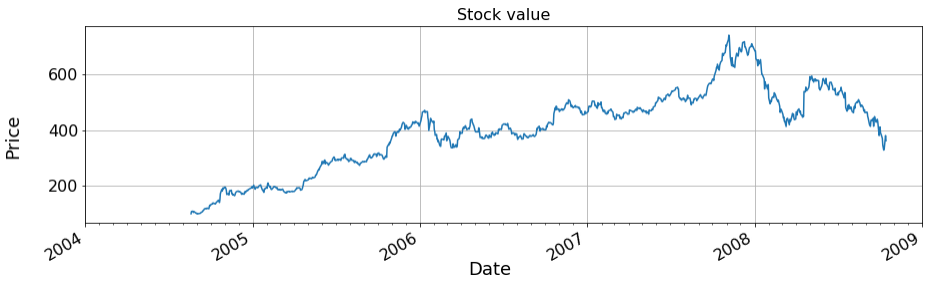

Figure 3: Sample time-series showing the prices of a sample stock over five years

An example of a multivariate time-series is the collected data from several sensors installed in a car.
One main difference between time-series and other datasets is that the observations do not only depend on components d, but also on the time feature n. Thus, time-series analysis and the used statistical methods are mostly different from the methods used for random variables that assume independence and constant variance of the random variables.
To data analysts, time-series are important in a variety of fields like economy, healthcare and medical research, trading, engineering and geophysics. These data are used for forecasting and anomaly detection.

2.4 Time-series patterns

Time-series have some important properties which we will define briefly. They are significant to the statistical anomaly detection methods used later.

2.4.1 Trend

A time-series has a trend, if its mean μ is not constant, but increases or decreases over time. A trend can be linear or non-linear. The time-series in Figure 3 has a positive trend from 2005 until 2008, and a negative trend afterwards.

2.4.2 Seasonality

Seasonality is the periodic recurrence of fluctuations. The time-series is called seasonal because seasonal factors like time of the year or day of the week, or other similarities are influencing it. Thus, it always has a fixed period of time that is limited to a year. Figure 4 shows a seasonal time-series. It is the monthly home sales index for 20 major US cities between the years 2000 and 2019 [86].

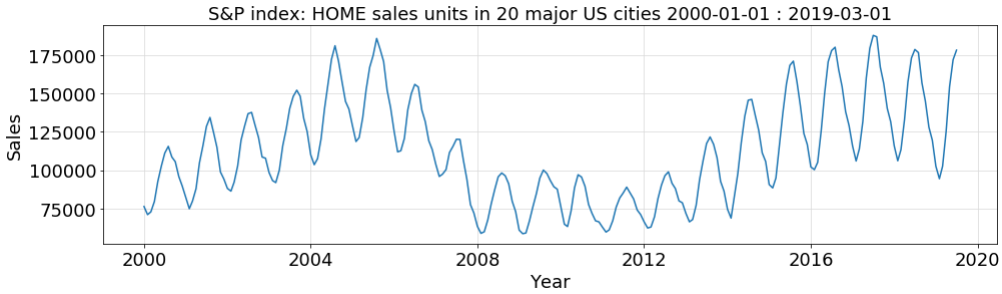

Figure 4: Sample time-series showing the prices of a stock over five years

2.4.3 Cycles

A cyclic time-series is influenced by time factors where the period is not fixed and the duration is above a year, e.g., a decade. The time-series in Figure 4 also has an approximate 12 year cycle.

2.4.4 Level

The time-series level is equal to the mean of the series. If a time-series has a trend, then it is often said that the level is changing.

2.4.5 Stationarity

Intuitively, a stationary time-series is a time-series having the same characteristics over every time interval. Formally, we can express it as follows [43]:

Definition 2.4. *X_t is a stationary time-series, if $\forall s \in \mathbb{R}$: the distribution of $(x_t, ..., x_{t+s})$ is equal.*

The above definition implies that a stationary time-series $x_1, .., x_T$ will have the following characteristics:

1. **Constant mean**, thus no trend exists in the time-series.
2. The time-series has a **constant variance**.
3. There is a **constant autocorrelation** over time.
4. The time-series has **no seasonality**, i.e., no periodic fluctuations.

2.4.6 White noise

White noise ϵ_t is a stochastic process, which is uncorrelated over time from a fixed distribution with a constant mean $\mu = 0$ and a constant and finite variance σ. Thus, white noise is stationary. One important characteristic of white noise is that its autocorrelation function (ACF) and its partial autocorrelation function (PACF) are zero, meaning that there is no dependence between two timestamps. Usually, in many theoretical models it is assumed that white noise is Gaussian: $\epsilon \sim \mathcal{N}(0, \sigma)$.

2.5 Anomaly detection

After having a general definition for anomaly and time-series, we will define what anomaly detection means and what kind of methods exist.
In literature, different terms are used that have the same or similar meaning to *Anomaly Detection*: *Event detection, novelty detection, (rare) event detection, deviant discovery, change point Detection, fault detection, intrusion detection or misuse detection* [36]. The different terms reflect the same objective: to detect rare data points that deviate remarkably from the general distribution of the dataset. The amount of deviation is usually regarded as a measure of strength of the anomaly or—probabilistically— the likelihood of being an anomaly, which is called *anomaly score*. Thus formally, anomaly detection can be defined as a function ϕ:

$$\begin{aligned} \phi : \mathbb{R}^n &\to \mathbb{R} \\ \phi(x) &\mapsto \gamma \end{aligned} \tag{1}$$

where γ is the anomaly score, $x \in X \subseteq \mathbb{R}^n$, and X is the dataset.
To convert the continuous value γ into a binary label – normal vs. anomaly – a threshold $\delta \in \mathbb{R}$ is defined where all points with an anomaly score greater than δ are marked as an anomaly. Thus, let $\phi_{score} := \phi$, then the binary labeling anomaly detection method ϕ_{binary} can be defined as:

$$\begin{aligned} &\phi_{binary} : \mathbb{R}^n \to \{normal, anomaly\} \\ &\phi_{binary}(x) \mapsto \begin{cases} anomaly, & \text{if } \phi_{score}(x) > \delta \\ normal, & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

Anomaly detection using ϕ_{binary} is not a trivial binary classification. As stated in Definition 2.1, anomalies form a very small part of the dataset. Often the anomalous part of a dataset is less then 1%. Therefore, usual binary classifiers would achieve above 99% accuracy if all data points would be labeled as normal, making anomaly detection a more difficult task. Nevertheless, to achieve satisfying results in anomaly detection, the proper anomaly detection method has to be selected which is dependent on the properties of the inspected data. The following properties are important for selecting the appropriate approach:

1. Temporal vs Non-Temporal data: Non-Temporal data can be medical images, protein sequences etc. Temporal data include time-series, but also data with timestamps of unequal interval.
2. Univariate vs Multivariate data: Univariate data takes only one dimension, e.g. the stock price, while multivariate data contains multiple dimensions. Instances of multivariate data are images or time-series observed by several sensors.
3. Labeled or unlabeled data: A dataset is labeled if an annotation exists for each element in the dataset, which determines if it is a normal or anomalous data point. A labeled dataset with normal and anomalous points is the object of supervised anomaly detection methods. It is possible that the dataset is completely labeled but only consists of normal points. Then, it can be analysed by semi-supervised methods. Finally, unlabeled data is the object of unsupervised anomaly detection methods.
4. Types of anomalies in the dataset: Section 2.2 introduced different anomaly types. This information affects the selection of the anomaly method. Point anomalies are detected by methods for rare classification. To detect collective anomalies, the methods focus on unusual shapes in the data, while searching for deviation. This aids in finding contextual anomalies.

In this paper, we focus on univariate temporal data and specifically time-series containing labeled normal and anomalous points. Therefore, we will introduce the general concepts of these kinds of methods here.

2.5.1 Anomaly detection on time series

Anomaly detection on non-temporal data like spatial data is different than on time-series. For example one of the main methods to detect anomalies in spatial data is by measuring the deviation of the abnormal points to the rest of the data. Another way is to cluster the whole dataset and mark all points as anomalies that lie in less dense regions. The main assumption about spatial data is that the data points are independent from each other.

This is different in time-series data. Here, the data points are not completely independent, but it is assumed that the latest data points in the sequence influence their following timestamps. Following this, values of the sequence change

smoothly or show a regular pattern. Thus, sudden changes in the sequence will be regarded as an anomaly. To show this behavior, consider the following example which demonstrates a time-series listing the temperature of an engine recorded every 10 minutes: $30°$, $31°$, $33°$, $32°$, $34°$, $35°$, $35°$, $85°$, $87°$, $88°$, $89°$, $89°$. If these points are regarded as independent points, most methods will not identify any anomalous behavior, but detect two equally distributed clusters. But in a time-series, the sudden change from $35°$C to $85°$C should be detected as an anomaly. The dependency between timestamps also results in the fact that anomalies in time-series are generally contextual or collective.
Aggarwal [2] breaks down anomaly detection methods for time-series into two main categories:

1. Anomaly detection based on prediction of the time series
2. Anomaly detection based on unusual shapes of the time series

Most statistical anomaly detection methods on time-series are based on time-series prediction. On the other side, there are several machine learning methods, which try to detect anomalies using clustering methods on time-series. The selected method is dependent on whether the time-series is univariate or multivariate. As the focus of this paper are univariate time-series, we will provide an overview of their characteristics.

2.5.2 Anomaly detection on univariate time series

Anomaly detection in time-series is strongly linked to time-series analysis and forecasting methods. To detect anomalies in univariate time-series, a forecasting model is fitted to the training data. Then, the test data is used to make predictions. To make a prediction on the test data, usually a sliding window is used. A sliding window is a subsequence of a time-series, which is fed as the input to the model enabling it to predict the value for following timestamp. Formally, let w be the width of the sliding window, and suppose we want to predict x_i, and ψ is the forecasting model. To forecast x_i the following function and input data is used:

$$\begin{aligned} &\psi : \mathbb{R}^w \to \mathbb{R} \\ &\hat{x}_i = \psi((x_{i-w}, ..., x_{i-1})) \end{aligned} \tag{3}$$

The anomaly score can be computed by measuring the distance between the predicted value $\hat{x}_i$ and the real value x_i:

$$e_i = d(x_i, \hat{x}_i) \tag{4}$$

where d is a distance function. In univariate time-series, usually the euclidean distance is used. The deviation e_i – also called error value – is proportional to the anomaly score. If the anomaly score is above a threshold $\delta \in \mathbb{R}$, it is marked as an anomaly.
As mentioned, there are also approaches which try to detect anomalies in time-series by looking for unusual shapes by using machine learning approaches. For instance, Zhang et al. [98] use One-Class Support Vector Machines to detect anomalies in time-series.Therefore, in contrast to spatial data, a sliding window with width w is defined. Then, for each timestamp x_i the preceding w-timestamps are analysed using some clustering or density methods. These methods are based on the assumption, that an contextual or collective anomaly will show a deviating shape which can be detected by these clustering or density methods. We will return to this issue in Section 3.

2.5.3 Supervised vs. Semi-supervised vs. Unsupervised Anomaly detection methods

If the time-series dataset is labeled, such that for each timestamp it is known if it is an anomaly or not, and additionally the dataset contains normal and anomalous timestamps, then a supervised anomaly detection method can be used. Supervised anomaly detection methods are able to detect an appropriate value for $\delta \in \mathbb{R}$ to classify all timestamps x_i as an anomaly if the corresponding anomaly score is $\phi(\hat{x}_i) > \delta$.
Semi-supervised approaches can be used if the dataset only consists of normal points and no anomaly exists. Then a model is trained, which fits to the distribution of the time-series and detects any new point deviating from this distribution as an anomaly. One-Class SVN, autoencoders or GANs are usual methods used for this sort of data.
Finally, unsupervised anomaly detection methods assume that the time-series data is unlabeled. Most unsupervised anomaly detection methods try to determine δ by analyzing the distribution of all $e_i, i \in \{1, ..., N\}$ and use the τ-percentile value as $\delta \in \mathbb{R}$. One widespread approach is to set $\delta = 3\sigma$ where σ is the standard deviation of the distribution of $e_i, i \in \{1, ..., N\}$.
In this paper we will focus on supervised anomaly detection methods and use labeled datasets in our experiments to evaluate them.

2.5.4 Statistical vs. Machine Learning vs. Deep Learning Anomaly detection approaches on time series

Munir et al. [63] categorize outlier detection methods in probabilistic models, statistical models, linear models, proximity based models, and outlier detection in high dimensions while referencing to Aggarwal's book [2].
We believe that all anomaly detection methods on time-series data can be divided in three main categories:

1. Statistical methods
2. Classical machine learning methods
3. Methods using neural networks (Deep Learning)

This categorization is goal-driven as we want to inspect if they behave differently. Some studies merge the second and third class in machine learning approaches [59]. The boundary of the third category using neural networks is rather clear as it only contains methods using some kind of a neural network. In Section 3 we will define what a neural network is. In contrast, the boundary between statistical and machine learning approaches are vague. Generally, statistical approaches assume that the data is generated by a specific statistical model [13]. On the other hand, machine learning methods consider the data generation process as a black box and try to learn from the data only. The machine learning methods are based on the implicit assumption that the underlying data generation process is not relevant as long as the machine learning methods are able to produce accurate predictions [71]. Thus, the machine learning methods rely on data modelling. Breiman [13] regards these two approaches as two different cultures and recommends to use the machine learning approach. There is an ongoing debate about which of these methods perform better. In this paper, we want to evaluate each of them quantitatively to provide more clarity on this subject.

3 Selected Anomaly detection approaches for time series

In this section, various anomaly detection methods are introduced. We divide the approaches in three categories: statistical approaches, classical machine learning approaches and anomaly detection methods using neural networks.

3.1 Anomaly detection using statistical approaches

As statistical approaches, we have selected some well-researched regressive models such as AR, MA, ARMA, ARIMA and some of the models that have worked well in the Makridakis competitions (also known as M-Competitions), and also some recently published papers. Although M-Competitions compare statistical forecasting methods, the anomaly detection methods on time-series is closely linked to the forecasting approaches. In this regard, they provide a good reference for effective statistical algorithms in time-series analysis.

1. **Autoregressive Model (AR)**
   One of the most basic stochastical models for univariate time-series is the Autoregressive model (AR). AR is a linear model where current value X_t of the stochastic process (dependent variable) is based one a finite set of previous values (independent variables) of length p and an error value ϵ:

   $$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + c + \epsilon_t \tag{5}$$

   The AR model in Equation 5 with a preceding window length of p is also called AR process of order p or AR(p). The error values ϵ_t are considered to be uncorrelated and have a constant mean of zero and constant variance σ. In this model, ϵ is used to determine the anomaly score.
   The values of the coefficients $a_1, ..., a_p, c$ can be approximated by using the training data and solving the corresponding linear equations with least-squared regression. After that, ϵ_t for each X_t can be computed, which represents the anomaly score. Hence, the anomaly score is equal to the difference between the forecasted value and observed one [51].
   AR models assume that the data is stationary. Thus, it is important to analyse the data and transform it if necessary.
2. **Moving Average Model (MA)**
   While the AR model considers X_t as a linear transformation of the last p observations of a time-series $\{x_t, x_{t-1}, ..., x_{t-p}\}$, the *moving average Model (MA)* considers the current observation X_t as a linear combination of the last q prediction errors $\{\epsilon_t, \epsilon_{t-1}, ..., \epsilon_{t-q}\}$:

   $$X_t = \sum_{i=1}^{q} a_i \cdot \epsilon_{t-i} + \mu + \epsilon_t \tag{6}$$

   The MA model in Equation 6 with a preceding window of length q is also called MA process of order q or MA(q). μ is the mean of the time-series and the coefficients $\{a_0, ..., a_q\}$ are learned from the data. In contrast to AR, learning the coefficients in MA is more complicated. While in the AR model preceding values $\{x_t, x_{t-1}, ..., x_{t-p}\}$ are known, in the MA model the values of $\{\epsilon_t, \epsilon_{t-1}, ..., \epsilon_{t-q}\}$ are unknown at the

beginning. The errors are known after the model is fitted. Thus, they are optimized sequentially. Therefore, a closed solution for the MA models does not exist and an iterative non-linear estimation algorithm is used to solve MA models [77]. After fitting the model, we use the deviation to detect anomalies like in the AR model.

3. **Autoregressive Moving Average Model (ARMA)**
Another model is the combination of AR and MA, which is often used for univariate time-series in practise. A time-series of the ARMA(p,q) model is dependent on last p observations and q errors:

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + \epsilon_t \tag{7}$$

$\{X_T\}$ is an ARMA(p,q) process if $\{X_T\}$ is stationary.
ARMA models use less variables in practice compared to AR and MA. However, the main challenge is to select appropriate values for p and q. The bigger these two values are, the more likely is it that the model overfits, resulting in too many false negatives in the anomaly detection process. On the other side, if they are chosen too small, the model will underfit and too many false positives will arise, i.e., data points are detected as anomalies although they are not. In both cases, the model is not able to detect the anomalies correctly.
There are several ways to fit the model and find appropriate values for p and q:

   (a) **Using correlograms**: First of all the data must be transformed, if necessary, to become stationary. Each ARMA model has its own specific autocorrelation and partial autocorrelation graphics, which can be visualized in a correlogram. The same is true for AR and MA. Thus, the autocorrelation function (ACF) and partially autocorrelation function (PACF) of the time-series is computed. Then, it will be discovered which p and which q of the ACF and PACF correlogram of ARMA(p,q) are similar to ours. This is an iterative process where different values of p and q are evaluated.

   (b) **Leave-One-Out Cross-Validation**: Another proposed way is using the observed data to minimize the error value by assigning different combinations of p and q using leave-one-out cross-validation [2].

   (c) **Box-Jenkins Method** [11]: The Box-Jenkins method which was introduced by George Box and Gwilym Jenkins proposes an iterative method:
      1. Identification: Use the data information to select the best model that represents the data by setting the p and q values. Additionally, evaluate whether the data is stationary and transform it if this is not the case. For this purpose, the ACF and PACF plots are helpful.
      2. Estimation: The model is fitted to the data so that the parameters a_i and b_i can be estimated.
      3. Diagnostic checking: The fitted model is checked with the data to evaluate its performance and if any inadequacies are witnessed. If the result is inadequate, we return to step 1.

   These approaches are not only used for the ARMA model, but are general methods for all statistical approaches.

4. **ARIMA Model**
One of the main problems with datasets is the fact that they can be non-stationary. Stationarity is a precondition for models like ARMA. The ARIMA model is a generalization of the ARMA model. In addition to the p and q parameter, it is also defined by a d parameter which defines the number of times the time-series is differenced. For $d = 1$, the time-series $\{x_0, ..., x_T\}$ is differenced as follows:

$$X'_i = X_i - X_{i-1}, \forall i \in \{1, ..., T\} \tag{8}$$

The effect of differencing is best shown through plotting some data. Figure 5 shows the stock data from Section 2.3 for the years 2005 to 2008. The data shows a positive trend. Therefore, we do not have stationary data here. However, Figure 6 shows the daily changes of the same stock over the three years, and the data is stationary now:

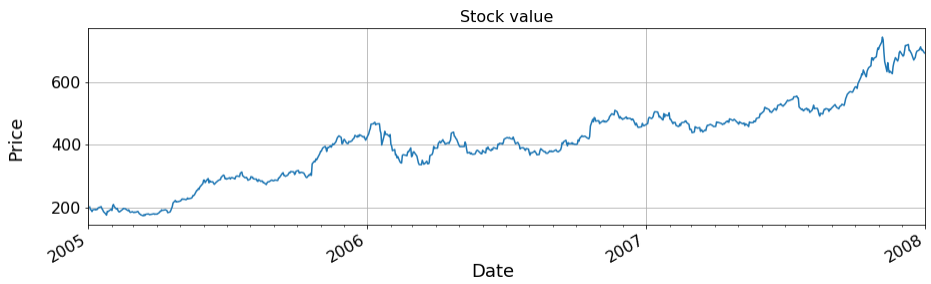

Figure 5: Stock value from 2005 to 2008

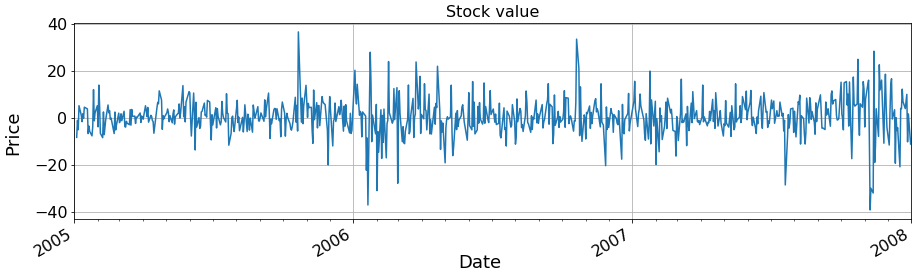

Figure 6: Value changes of the stock value (via differencing method)

Figure 6 plots the differences between the consecutive data points of Figure 5.
Differencing removes a trend in the time-series resulting in a constant mean. If the trend is non-linear, differencing must be done several times, thus, $d > 1$.
Differencing is also used to remove seasons. The *Seasonal Differencing* is as follows:

$$X'_t = X_t - X_{t-n} \text{ where n is the duration of the season} \tag{9}$$

After fitting the ARIMA model, anomalies are detected by evaluating the deviation of the predicted point to the observed one.

5. **Simple Exponential Smoothing (SES)** [15]
While in the previous models the prediction is a linear optimization problem, SES uses a non-linear approach by taking the previous time-series data to predict, assigning exponential weights to the observations:

$$\begin{aligned} X_{t+1} =& \alpha X_t + \alpha(1-\alpha)X_{t-1} + \alpha(1-\alpha)^2 X_{t-2} + ... + \alpha(1-\alpha)^N X_{t-N} \\ & \text{where } \alpha \in [0,1] \end{aligned} \tag{10}$$

Thus, X_{t+1} is a weighted combination of the previous data points. The parameter α defines the rate at which the weights decrease, which is exponential. Therefore, it is called *Exponential Smoothing*. The smaller α is, the more weight is given to data points that are more distant.

6. **Double and Triple Exponential Smoothing (DES, TES)** [44]
SES assumes that the data is stationary. SES can be extended to also handle non-stationary data, which is called Double Exponential Smoothing. Here an additional parameter β is introduced to smooth the trend in the series. If the data also contains seasonality, Triple Exponential Smoothing is used. This extension also contains a parameter γ to control the effect of seasonality.

7. **Time-series Outlier Detection using Prediction Confidence Interval (PCI)** [96]
This approach uses a sequence of previous data which are weighted non-linear to forecast the next data point. Then, by using the threshold, they classify a data point as anomaly or normal.
Thus, to calculate X_t, it uses a window of past observed points of the series:

$$X_t = \frac{\sum_{j=1}^{2k} w_{t-j} X_{t-j}}{\sum_{j=1}^{2k} X_{t-j}} \tag{11}$$

where w_{t-j} is the weight for X_{t-j} and it is proportional to the inverse of the distance between X_t and X_{t-j}. This gives temporal closer points X_t more weight. If the anomaly detection is done offline, a two sided window can be computed:

$$X_t = \frac{\sum_{j=1}^{k} w_{t-j} X_{t-j} + \sum_{j=1}^{k} w_{t+j} X_{t+j}}{\sum_{j=1}^{k} X_{t-j} + \sum_{j=1}^{k} X_{t+j}} \tag{12}$$

Then, the approach computes an upper and lower bound for the anomaly detection:

$$PCI = X_t \pm t_{\alpha,2k-1} \cdot s\sqrt{1 + \frac{1}{2k}} \tag{13}$$

Here, the factor $t_{\alpha,2k-1}$ is the p-th percentile of a Student's t-distribution with $2k-1$ degrees of freedom, s is the standard deviation of the model residual, and k is the window size used to calculate s. If X_t is outside the boundaries, it is marked as an anomaly.
Thus, this method has some hyperparameters: α to calculate the plausible range of PCI and k as the window

size. Here, the analyst faces again the challenge to overcome overfitting and underfitting by adjusting these parameters correctly.
The authors used this method for hydrological time-series data. In their corresponding experiments, they recommend an α value from the interval $[0.85, 0.99]$ and k value from the interval $[3, 15]$
This method is a simplification of the previous methods as the coefficients are not fitted by the model like AR, MA, ARMA or other autoregression approaches. It does also not use exponential weights like the ES methods. However, it was included in the evaluation of this paper, because it is a more recent approach that was published in 2014. Nevertheless, it is debatable if this method is statistical or a ML method.

3.2 Anomaly detection using classical machine learning approaches

Machine learning algorithms try to detect anomalies in time-series datasets without assuming a specific generative model. They are based on the fact that it is not necessary to know the underlying process of the data, to be able to make time-series prediction and time-series anomaly detection. Therefore, these methods are well advanced outside the field of statistics (Breiman [13]). Many researchers argue that a theoretical foundation of a model can be neglected, if the method performs effectively in practise (Januschowski et al. [47]). In this context, we introducte section several univariate anomaly detection methods in this section, which are based on classical machine learning algorithms. Later, the performance of these algorithms are compared to the statistical approaches introduced so far, and the deep learning approaches in Section 3.3.

1. **K-Means Clustering – Subsequence Time-Series Clustering (STSC)**
One of the clustering algorithms for anomaly detection is using K-Means clustering [56]. This method is also called *Subsequence time-series Clustering (STSC)* [45]. To use K-Means as an anomaly detection method for time-series data, a sliding windows approach is used [95, 28]. This implies that given a time-series $\{X_N\} = (x_1, x_2, ..., x_N)$ and window length w and a slide length γ, the time-series $\{X_N\}$ results in a set of sub sequences $\mathcal{S} \subseteq \mathbb{R}^{(N-w)\times w}$:

$$\mathcal{S} = \{(x_0, x_1, ..., x_w)^T, (x_{0+\gamma}, x_{1+\gamma}, ..., x_{w+\gamma})^T, ..., (x_{N-w}, x_{N-w+1}, ..., x_N)^T\} \tag{14}$$

After defining the desired number of clusters k, the k-Means algorithm is executed on the dataset $\mathcal{S}$ until it converges, resulting in k centroids [12]. The centroids are the mean of the vectors in the specific cluster. The set of k centroids shape the set $\mathcal{C}$.
To detect anomalies, the distance of each subsequence $s \in \mathcal{S}$ to its nearest centroid is computed, which results in the sequence $\mathcal{E}$:

$$\mathcal{E} = (e_0, e_1, ..., e_{|\mathcal{S}|}) \tag{15}$$

where e_i for $i \in \{0, ..., |\mathcal{S}|\}$ is:

$$e_i = \min_{\forall c \in \mathcal{C}}(d(s_i - c)) \tag{16}$$

where d is a distance function. Usually, the euclidean distance is used for univariate data.
Thus, the sequence $\mathcal{E}$ represents the error value of each sliding window. By defining a threshold $\delta \in \mathbb{R}$, a window $s_i \in \mathcal{S}$ is an anomaly if the corresponding error value $e_i > \delta$.
The main challenge of this approach is specifying an appropriate value k. The complexity of this method is O($kNrw$) where k is the number of clusters, r the number of iterations until convergence, N the number of objects (here $N = |\mathcal{S}|$) and w the length of the sliding window [49].
Note: Keogh and Lin [49] have demonstrated in their work that using sub-sequences of time-series for clustering algorithms is meaningless. They showed, that the cluster centers found for several runs of the K-means algorithm on the same dataset are not significantly more similar to each other than the cluster centers of a random walk dataset. That means that after being asked to present the centroids on a dataset, they could just present the centroids of a random walk and nobody would be able to distinguish between them [49]. They also tried other algorithms like hierarchical clustering, which is a deterministic approach compared to K-means, but received the same result. The same was proved on several datasets which furthermore confirmed their claim that using sub-sequences of the time-series data for clustering techniques is meaningless. They also tried different distance measures like Manhattan, L_∞ and Mahalanobis distance. Furthermore, by using K-Means with $k = 3$ and $w = 128$ on the famous Cylinder, Bell and Funnell (CBF) dataset, they showed that the resulting centroids are sinus waves, which are totally different to the instances in the CBF dataset. Several authors tried to analyse this behavior mathematically [45, 30, 67], and there have been a lot of attempts to solve these problem, or at least to show time-series patterns that would work with STSC [20, 75]. But the problems remain generally unsolved [102].
We will use STSC in our evaluation as it still is one of the basic clustering approaches and serves as a comparing artifact.

2. **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**
Another anomaly detection method based on clustering is Density-Based Spatial Clustering of Application with Noise algorithm (DBSCAN) [25]. In comparison to other clustering methods like STSC or CBLOF [41], it also analyses the density in the data.
The method classifies the data points into three different categories:

   - Core points
   - Border points
   - Anomalies

   To classify the points, the user has to specify two parameters: ϵ and μ where ϵ is the distance to declare the neighbors of the analysed point and μ is the minimum number of points each normal cluster has to have.
   To classify a point, first the ϵ-neighbors of each point have to be determined. Thus, for the dataset $\mathcal{D}$ where $\mathcal{D} = \{x_i | x_i \in \mathbb{R}, i \in \{1, ..., n\}\}$, the ϵ-neighbors of x_i is:

   Definition 3.1. $\epsilon - neighbors(x_i) = \{x_j | x_j \in \mathcal{D} : \phi(x_i, x_j) \leq \epsilon, x_i \neq x_j\}$

   where ϕ is a distance function.
   Then, a point $x_i \in \mathcal{D}$ is a *Core Point* if :

   Definition 3.2. *CorePoint(x_i)*$= true \Leftrightarrow \epsilon - neighbors(x_i) \geq \mu$

   Border points are declared as follows:

   Definition 3.3. $BorderPoint(x_i) = true \Leftrightarrow \exists x_j \in \mathcal{D} : x_j \neq x_i \wedge x_j \in \epsilon - neighbors(x_i) \wedge CorePoint(x_j) = 1$

   It is also possible to set a threshold δ for border points, such that it should have more than δ CorePoints as neighbors.
   Finally, anomalies are defined as follows:

   Definition 3.4. $Anomaly(x_i) = true \Leftrightarrow CorePoint(x_i) = false \wedge BoarderPoint(x_i) = false$

   Çelik et al. [103] have used DBSCAN for anomaly detection on a univariate time-series dataset, which contains the daily average temperature observations for 33 years. They first split the dataset into sequences containing the data of a month. Then, the data is normalized using the mean and variance of the data's sequence. After that, DBSCAN is run on each sequence and the anomalies are detected as described before. The main challenge is to select appropriate values for the parameters ϵ-Distance and μ as the minimum number of points in each cluster.

3. **Local Outlier Factor (LOF)**
Another popular clustering algorithm is Local Outlier Factor (LOF) clustering. In contrast to DBSCAN, it is not based on density, but finding the nearest neighbors (K-NN) [8], while also focusing on local outliers. LOF was initially designed to detect anomalies on spatial data [14]. But Oehmcke et al. [66] extended the approach to use it also for time-series data.
Let $\mathcal{D}$ be a dataset and $x \in \mathcal{D}$. To calculate the LOF value of a data point x, the following steps have to be performed:

   (a) Compute the k-distance δ_k of x:
   let $k \in \mathbb{N}_+$ and ϕ a distance function, then δ_k = k-distance of x if:

   $$\phi(x, y) = \delta_k, \text{ where } x \in \mathcal{D} \text{ and } y \text{ is the k-th neigherst neighbor to } x$$

   Then k-distance neighborhood of x is the follows:

   $$N_{k-distance(x))}(x) = \{y | y \in \mathcal{D}, \phi(x, y) \leq \delta_k\}$$

   And the reachablity distance RD of x is defined as:

   $$RD_k(x, y) = max\{k - distance(y), \phi(x, y)\}$$

   (b) Then the local reachability density (LRD) of x is computed:

   $$LRD_k(x) = 1/\left(\frac{\sum_{y \in N_{k-distance(x)}} RD_k(x, y)}{|N_{k-distance}(x)|}\right)$$

   (c) Finally, the local outlier factor of x can be computed:

   $$LOF(x) = \frac{\sum_{y \in N_{k-distance(x)}} \frac{LRD_k(y)}{LRD_k(x)}}{|N_{k-distance}(x)|}$$

Oehmcke et al. [66] used a sliding window with length w to classify a w-long sequence as an anomaly. Thus, given a time-series X_t, it is split into a training set A and test set B. Using the window length w, each set is transformed as follows:

$$A(w,t) \subseteq P(\{X_T\}) = \{(x_i, ..., x_{i+w}) | i \in \{1, ..., t-w\}, t \leq |\{X_T\}|\} \tag{17}$$

$A(w,t)$ is a set of time-series of length w.

$$B(w,t) \subseteq \{X_T\} = \{x_{t-\frac{w}{2}}, ..., x_{t+\frac{w}{2}}\} \tag{18}$$

And $B(w,t)$ is the sequence we want to analyse.
To determine if $B(w,t)$ is an anomaly or not, we compute the anomaly score ϕ of $B(w,t)$ and if $\phi > \delta$ where $\delta \in \mathbb{R}$ is a threshold then we mark $B(w,t)$ as an anomaly.
The anomaly score ϕ of $B(w,t)$ specifies, how much $B(w,t)$ is different than the sets in $A(w,t)$. This is done by computing the LOF value of $\{\{B(w,t)\} \cup A(w,t)\}$. If $LOF(B(w,t)) > \delta$, then the sequence is marked as an anomaly.
The main challenges of this approach are the following items:

- Determine an appropriate value k for the k-nearest neighbors to compute the LOF value. In general, it is considered that prior knowledge is available to determine an appropriate value for k. The authors of the paper suggest using an ensemble strategy to compute k [31].
- Concatenating the time-series into a vector and computing the distance to other vectors removes the ordered information of a time-series. Here, temporal data is converted into spatial data, where each dimension is equally important. But in time-series the ordered sequence contains important information, which is used in some statistical approaches like Exponential Smoothing.
- Determine an appropriate distance function ϕ. While Goldstein and Uchida recommend using the Euclidean distance, there have been many cases where the Euclidean distance is not suitable, especially to also consider the relationship between variables in a multidimensional space.
- The LOF value only relies on the direct neighbors, which also makes it more appropriate to detect local anomalies.
- Another drawback of the LOF algorithm is the complexity of $O(n^2)$ as compared to DBSCAN with a complexity of $O(n \cdot log\ n)$.

4. **Isolation Forest**
One of the machine learning approaches to detect anomalies in time-series is isolation forest using a sliding window. Isolation forest, also known as iForest, was introduced by Liu et al. [53]. It builds an ensemble of Isolation Trees (*iTrees*), which are binary trees isolating data points. As anomalies are more likely to be isolated than non-anomalous points, it is more likely that they are closer to the root of an iTree [54]. Figure 7 shows how an iTree is generated on a sample data structure. An anomaly is detected after two partitions while the first normal point is detected after the fourth partition:

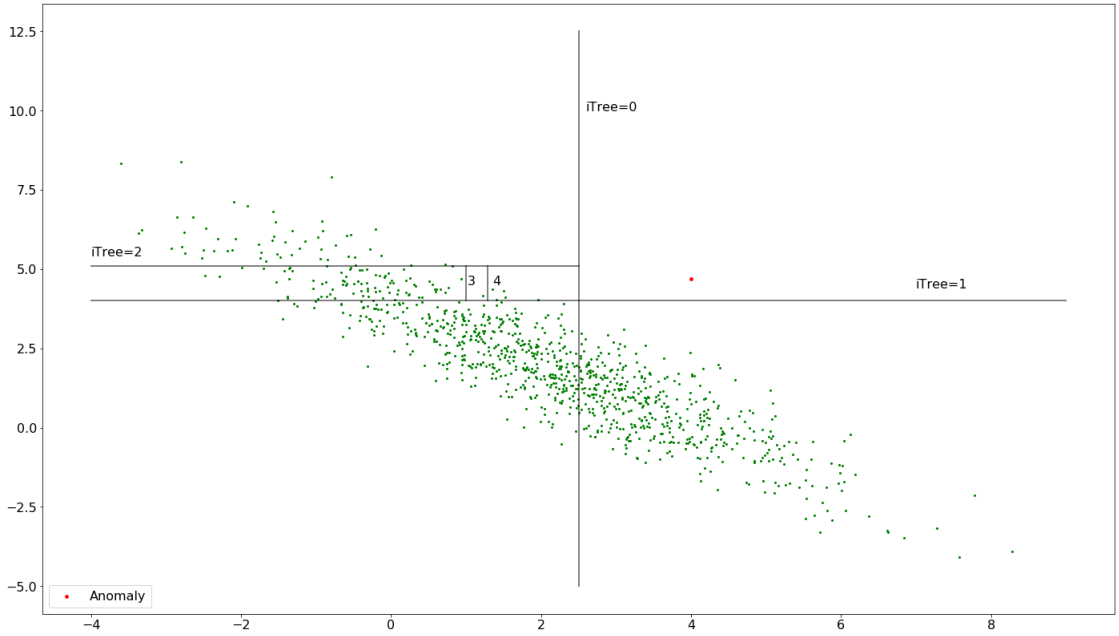

Figure 7: Isolation Forest: The anomaly is isolated by the random generated tree after two partitions

Thus, this method considers points with shorter path lengths as candidates that are highly likely to be anomalies. The anomaly detection process using Isolation Forests is performed generally in two steps:

(a) Training: Create n iTrees for the given training set.

(b) Evaluation: Pass the test instance through the isolation trees to determine the anomaly score.

There are some methods that have extended iForest to detect anomalies on time-series. One main approach to detect anomalies in univariate data is to analyse the dataset in sequences defined by the window length w. Thus, let $\{X_T\} = (x_1, x_2, ..., x_T)$ be a univariate time-series, w the window length and $W \subseteq \mathbb{R}^{p \times w}$. Then:

$$\begin{aligned} W :=& (W_1, W_2, ..., W_p) \\ =& ((x_1, ..., x_w)^T, (x_2, ..., x_{w+1})^T, ..., (x_p, ..., x_{w+p-1})^T) \end{aligned} \tag{19}$$

After that, the anomaly score on each sequence is computed, which is proportional to the average path length of an instance. Using supervised learning, the threshold can be computed on the training set and used for the test set later.

Ding and Fei [24] have extended the concept to compute the anomaly score $S(x, p)$, where x is the data point and w is size of the window:

$$\begin{aligned} S(x, w) &= 2^{-\frac{E(h(x))}{c(w)}} \\ E(h(x)) &= \frac{1}{L} \sum_{i=1}^{L} h_i(x) \end{aligned} \tag{20}$$

where $h_i(x)$ denotes the length of the i-th iTree, $E(h(x))$ the average of $h(x)$ from a collection of iTrees and $c(p)$ is the average of $h(x)$ given w and L, the number of iTrees.

The main challenges of the isolation forest algorithm are the following parameters:

- Window length w: If the length is too short, then there will not be enough data to construct an appropriate model. On the other hand, if the length is too long, older and sometimes less relevant data will be considered as much as more recent data points. Ding and Fei [24] have shown in their experiment results that fixed sliding windows for different datasets result in bad performance.
- Number of iTrees in the iForest: The higher the number of iTrees, the closer the average value is to the expected value. The downside is that a higher number of iTrees will increase the computation time [54]. For w as the window length and L as the number of iTrees, iForest has a time complexity of $O(L \cdot w^2)$ and a space complexity of $O(L \cdot w)$.
- Contamination: Many implementations of iForest, e.g. the implementation in *sklearn*, have a contamination parameter where the proportion of anomalies in the dataset is set. This also marks the threshold for

the anomaly. Improper assignment of this parameter could result in a higher rate of false positive or false negatives.

5. **One-Class Support Vector Machines (OC-SVM)**
The original support vector machine algorithm was invented as a linear supervised approach by Vapnik and Chervonenkis [89] in 1963. Boser et al. extended the algorithm by introducing the kernel trick, which made SVM capable of making non-linear classification. After that, a new approach to detect novelties using SVM was introduced called One-Class SVM (OC-SVM)[82]. OC-SVM is a semi-supervised approach where the training set consists of only one class: the *normal* data. After the model is fitted on the training set, the test data is classified as being similar to the normal data or not, making it able to detect anomalies.
The original OC-SVM method was able to detect anomalies in a set of vectors and not on time-series. Most papers recommend to project the time-series into a vector set. Ma and Perkins [55] propose to unfold the time-series into a phase space using a time-delay embedding process (Packard et al. [70]). Zhang et al. [98],[78] recommend to create windows with length w of the time-series dataset, so that for a given time-series $\{X_T\} = (x_1, x_2, ..., x_T)$, a window length w and $W \subseteq \mathbb{R}^{p \times w}$, the dataset is first converted into :

$$\begin{aligned} W :=& (W_1, W_2, ..., W_p) \\ =& ((x_1, ..., x_w)^T, (x_2, ..., x_{w+1})^T, ..., (x_p, ..., x_{w+p-1})^T) \end{aligned} \tag{21}$$

Then, a function p projects the time-series into a two dimensional space:

$$\begin{aligned} &f : \mathbb{R} \to \mathbb{R}^2 \\ &f(X_t) \mapsto \begin{cases} [X_t X_t], & \text{for t=1} \\ [X_t X_{t-1}], & \text{else} \end{cases} \end{aligned} \tag{22}$$

While in OC-SVM, the result is biased on time-series points with large values, it is recommended that the data is normalized.

6. **Extreme Gradient boosting (XGBoost, XGB)**
A machine learning technique which also performed well in the Kaggle and KDDCup competitions is Extreme Gradient boosting (XGBoost). One of the main advantages of XGBoost is its scalability [21]. XGBoost is derived from the Tree boosting algorithm using the second order method introduced by Friedman et al. [29]. Let D be a dataset of n examples with dimensionality m: $D = \{(x_i, y_i) | x_i \in \mathbb{R}^m, y \in \mathbb{R}, i \in \{1, ..., n\}\}$. Then, tree boosting uses a sequential sequence of tree models to make a prediction $\hat{y}$ for x_i:

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \text{ where } f_k \in \mathbb{F} \tag{23}$$

where $\mathbb{F}$ is the space of regression trees. The corresponding loss function is:

$$\begin{aligned} &\mathcal{L}(\phi) = \textstyle\sum_i l(\hat{y_i}, y_i) + \sum_k \Omega(f_k) \\ &\text{where } \Omega(f) = \gamma T + \tfrac{1}{2}\lambda ||w||^2 \end{aligned} \tag{24}$$

where T is the number of leaves in each tree and w the leaf weights.
The loss function in Equation 24 contains functions, which are not possible to optimize with traditional optimization methods in Euclidean space. To work around this obstacle, the model is trained in an additive manner and the taylor approximation of the loss function is used to make it optimizable in the Euclidean space.

$$\mathcal{L}^{(t)} \approx \sum_i [l(\hat{y_i}^{t-1}, y_i) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{25}$$

where g_i is the derivative of the loss: $\partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and h_i the second derivative: $\partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$.
Tianqi Chen et al. provide more details in their main paper about XGBoost [21].
Thus, XGBoost is used as a regression model to forecast time-series. To detect anomalies in univariate time-series, we extend the algorithm to compute the error in the prediction. Based on the training data, we are able to compute λ percentile of the error distribution and mark it as the threshold $\delta \in \mathbb{R}$ to detect anomalies.

3.3 Anomaly detection using neural networks

Since neural network have achieved tremendous results in computer vision tasks like object detection, classification and segmentation or similar tasks there have been increasing interest to use them for time-series forecasting and time-series

analysis. They are similar to classical machine learning approaches with regard to the fact that they do not presume any knowledge of the underlying data generation process. Their popularity is based on their empirical results.
Many researchers have tried to evaluate the performance of neural networks compared to the classical approaches like ARIMA. Sharda and Patil [85] compared 101 time-series using forward neural networks and the ARIMA model. Tang and Fishwick [87] also compared neural networks with ARIMA models focusing on 16 time-series with different complexities. Using neural networks for time-series forecasting paved the way for using neural networks to detect anomalies in time-series. In this section, we selected the most popular approaches used in recent years.

1. **Multiple Layer Perceptron (MLP)**
   The most fundamental artificial neural network architecture (ANN) is the Multilayer Perceptron (MLP) [38] which is a fully-connected feed-forward neural network. According to Hyndman and Athanasopoulos [43] a neural network used for time-series prediction is a *Neural Network Autoregression Model (NNAR Model)*. They characterize a NNAR model by the lagged input p and the nodes in the hidden layer k: NNAR(p, k). Thus:
   $$\text{NNAR}(p, 0) \Leftrightarrow ARIMA(p, 0, 0) \tag{26}$$
   where no seasonal restriction exists. Thus, p, which is the lagged input, represents also the window size w of the sliding window used on the time-series. The window size is equal to the number of neurons in the input layer of the MLP (Figure 8):

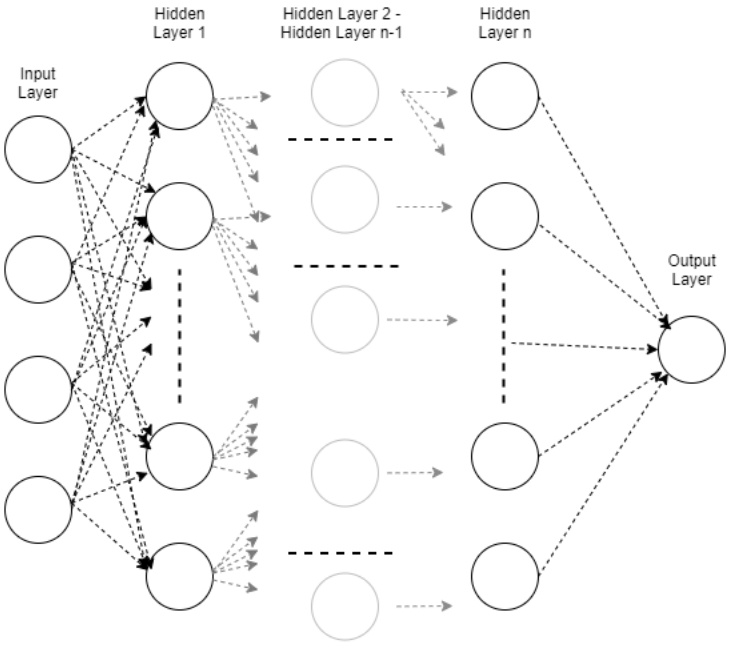

Figure 8: Multilayer Perceptron (MLP)

While Hyndman et al. use a neural network with one hidden layer and extend the number of neurons in the layer, it is also possible and sometime preferable to increase the number of hidden layers [5].
Haselsteiner and Pfurtscheller [37] used two different topologies of MLP for time-series classification which is a similar problem to anomaly detection. On the one hand, they implement an MLP using time-series with sliding window and on the other hand, a MLP with finite impulse response filters (FIR-MLP) is used [6].
In this survey, we will focus on MLP by sliding window on the time-series. The MLP network is used to make predictions. After that the error of the prediction is used to classify a data point as normal or as an anomaly considering the error value proportional to the anomaly score.
Thus, let $\{X_N\}$ be a time-series, $x_i \in \{X_N\}$, w the window length, and f the function of the MLP, then:
$$\hat{x}_{t+1} = f(x_{t-w}, ..., x_t), \forall t \in \{w, ..., n\} \tag{27}$$
Hence, the label for a window of time-series $(x_{t-w}, ..., x_t)$ is the next datapoint of the time series: x_{t+1}.
An MLP and all other neural network approaches can also be used to predict more than one timestamp:
$$(\hat{x}_{t+1}, ..., \hat{x}_{t+p_w}) = f(x_{t-w}, ..., x_t) \tag{28}$$

p_w is the number of timestamps the MLP predicts which is called the *prediction window* or *Forecasting Horizon*[63].
The prediction of the MLP is then used to detect anomalies. Let $\delta \in \mathbb{R}$ be the anomaly threshold, then x_{i+1} is marked as an anomaly, if:

$$f(x_{i-w}, ..., x_i) - x_{i+1} > \delta \tag{29}$$

The training set of the time-series data can be used to detect a proper value for δ.
One of the main challenges of neural networks is the hyperparameter tuning task. A MLP has a remarkable amount of hyperparameters:

(a) Depth of the MLP (Amount of the hidden layers of the network)
(b) Width of the MLP (Amount of nodes in each layer)
(c) Length of the window w
(d) Learning rate
(e) Optimization function

These parameters can be optimized using random search or more advanced techniques [9].

2. **Convolutional Neural Networks (CNN)**
Another artificial neural network approach which is used for anomaly detection in time-series data are deep convolutional neural networks (CNNs). CNNs are mainly used in computer vision for tasks like object detection, classification and segmentation [50, 76, 99]. In contrast to MLPs, where layers are fully connected, a CNN uses convolutional layers that are partially connected, reducing the amount of parameters, enabling them to go deeper and train faster. CNNs, in contrast to MLP, focus on local patterns in the data. In addition to the convolution layers, CNNs also use pooling layers for regularization to avoid overfitting. One of the pooling operations, which achieves good results regularly [81] is the maximum pooling [92].
In the recent years, there has been increasing interest in using CNNs for time-series analysis. Munir et al. [63] use a CNN architecture, called deep-learning based anomaly detection approach (DeepAnT), to forecast time-series and detect anomalies based on the error of the prediction. Zheng et al. [100] use a similar CNN architecture for classification of time-series data, a method that can also be extended to detect anomalies.
Using CNNs for time-series analysis is different to using CNNs for image classification. While the input for image classifying CNNs is 2D, univariate time-series CNNs use 1D input. Therefore, the kernels of the convolution layers are 1D, too. Figure 9 shows the architecture used in DeepAnT:

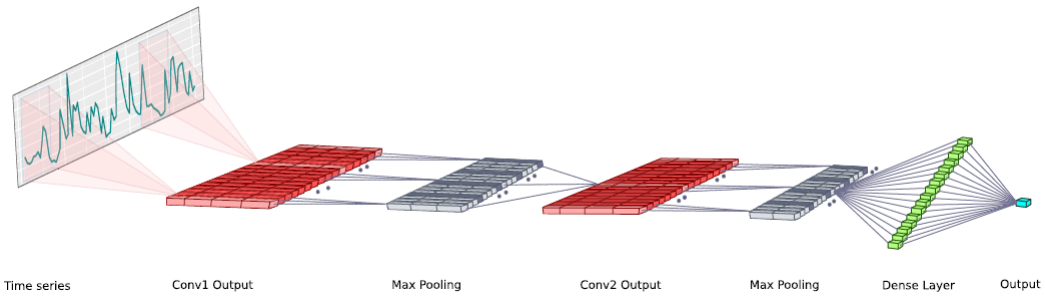

Figure 9: DeepAnT architecture for time-series prediction [63]

The first layer after the input layer is a 1-dimensional convolution layer followed by a max-pooling layer. As figure 9 shows, DeepAnT uses two pairs of convolution and max-pooling layers. However, this could vary based on the dataset. Therefore, one of the major hyperparameters of neural networks in general and CNNs in particular is the architecture of the model. The amount of convolution and max-pooling layers differ in the architectures used in this paper w.r.t. the dataset. We will analyse this in Section 5.

After the convolution and max pool layers, a dense layer is used which is fully connected to the output node. If the prediction window is greater than one, the amount of the output nodes will increase accordingly. As an activation function for the convolution layer and the dense layer, rectified linear units (ReLU) are used [65]. Ioffe and Szegedy suggest another regularization technique called Batch Normalization [46]. Experiments in computer vision showed that Batch Normalization results in a higher learning rate, acts as an alternative for dropout layers and decreases the importance of careful parameter initialization. Therefore, we also implement a CNN architecture using Batch Normalization for univariate anomaly detection to evaluate its performance in anomaly detection.

The CNN model is used to make a prediction in the same way as the MLP model. To detect the anomalies, the same algorithm is used.
Let $\delta \in \mathbb{R}$ be the anomaly threshold, f the function implemented by the CNN model, then x_{i+1} is marked as an anomaly, if:

$$f(x_{i-w}, ..., x_i) - x_{i+1} > \delta \tag{30}$$

In addition to the hyperparameters that MLP also had to handle, CNN expects the following:

(a) Architecture of the CNN, i.e., using Batch Normalization, Dropout or Max Pooling layers
(b) Amount of kernels in each convolution layer
(c) The size of the kernel
(d) Depth of the Convolution Layer

3. **Residual Neural Network (Resnet)**
An extension of the CNN model, which achieved good results in the last years is the Residual Neural Network (ResNet) architecture. ResNet introduces a new artifact called residual blocks developed by He et al. [40]. Residual blocks add the output of a convolutional block with the input of it using a skip connector. A convolutional block consists of several convolution layers, activation layers and regularization artifacts such as Max-Pooling or Batch Normalization layers.
To express residual blocks formally, let x_i be the input and ϕ be a convolution block, then the output of a residual block y is as follows:

$$y = \phi(x_i) + x_i \tag{31}$$

Usually, an activation function ψ like the RelU activation function is used as well:

$$y = \psi(\phi(x_i) + x_i) \tag{32}$$

Figure 10 shows a residual block that is used in this paper:

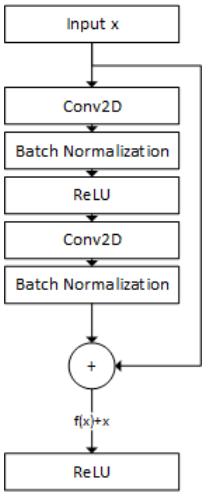

Figure 10: Residual Block used for time-series consisting of convolution block of two convolution layers with one ReLU and two Batch Normalization layers

Residual Blocks are used to avoid the vanishing gradient problem, which occurs often in deeper CNNs.
Wang et al. [90] use ResNet to classify time-series data. They use three residual blocks with 64, 128 and 128 filters. As compared to our residual block in Figure 10, they use three convolution layers and Batch Normalization layers with ReLU activation function. We tried different amount of residual blocks, which will be explained in detail in Section 5.
ResNet is best suited for large amounts of data. Therefore, it could overfit on the time-series data if the size of the data is too limited. But as we have achieved good results with ResNet in computer vision tasks, it would be of interest to evaluate this model on time-series data to detect anomalies, too.

4. **WaveNet**
WaveNet was developed by Oord et al. [68] as a deep generative model to create raw audio waveforms. Especially the ability to approximate the predictive distribution of each audio sample conditioned the previous

ones, makes it a proper candidate for time-series forecasting. Thus, WaveNet is a probabilistic model that tries to approximate the joint probability of a waveform $x = \{x_1, x_2, ..., x_T\}$:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}) \tag{33}$$

which makes audio sample x dependent on all samples before.
To accomplish that, WaveNet uses a specific kind of convolution layers: dilated convolution layers. Normal convolution layers use filters. A filter uses a convolution operation on the data with the same size as the size of the filter. Figure 11 shows a CNN with regular convolution layer:

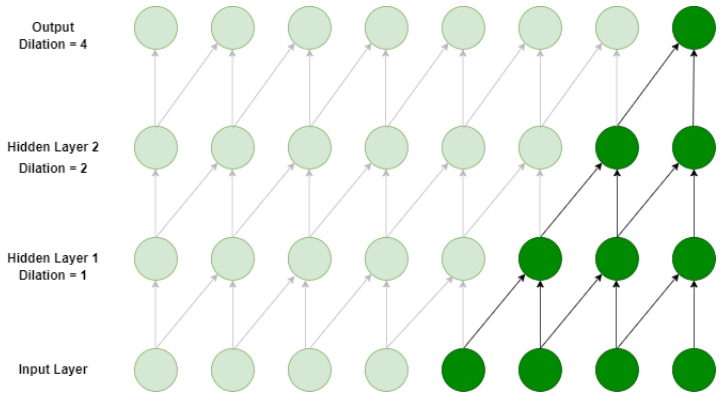

Figure 11: CNN with regular convolution layers with filter size of 2

In contrast, the dilated convolution layer performs the convolution operation on data bigger than the filter size. This is accomplished by skipping some input values using a skip step. Figure 12 shows a CNN with delation convolution layers:

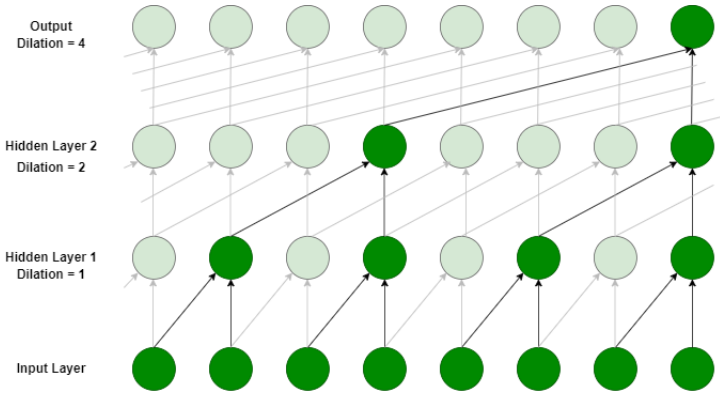

Figure 12: CNN with dilated convolution layers with filter size of 2 and changing skip steps (delation) = $\{1, 2, 4\}$

One big advantage of the dilation convolution layers is that it will learn long term and short term dependencies while normal convolution layers are designed to extract local patterns.
Borovykh et al. [10] used the concept of WaveNet to design a CNN for time-series forecasting. Thus, to predict a timestamp $x_t \in \{X_N\}$, a sequence of timestamps with width w is used as the input for the function f which is expressed by the CNN:

$$\hat{x}_t = f((x_{t-w}, ...x_{t-1})^T) \tag{34}$$

The dilation of the convolution layers increases by a factor of 2, which is illustrated in Figure 12. Therefore, the window width w can be much bigger than the value used in normal convolution layers.
In this paper, we will extend this approach to also detect anomalies where we use the same approach we used

for the MLP and CNN:
Let $\delta \in \mathbb{R}$ be the anomaly threshold, then x_t is marked as an anomaly, if:

$$\hat{x}_t - x_t > \delta \tag{35}$$

5. **Long Short Term Memory (LSTM) network**
Another ANN, which is designed for sequence data is the LSTM network. LSTM networks belong to the recurrent neural networks (RNN) architectures. In contrast to MLP and CNN, where the data is just flowing forward and therefore also called *feed-forward neural networks*, RNN networks have a feedback connection enabling them to use the output information for the next input of the sequence. Formally, the output of a neuron in a feed forward neural network is as follows:

$$y_t = \phi(x_t^T \cdot w_x + b) \tag{36}$$

where ϕ is a non-linear activation function. In contrast, the output of a neuron of a recurrent neural network is the following:

$$y_t = \phi(x_t^T \cdot w_x + y_{t-1}^T \cdot w_y + b) \tag{37}$$

A neuron in a simple RNN is computed as shown in Equation 37. Hochreiter and Schmidhuber [42] developed a new version of recurrent cells called Long Short Term Memory (LSTM). Figure 13 shows the structure of an LSTM cell.

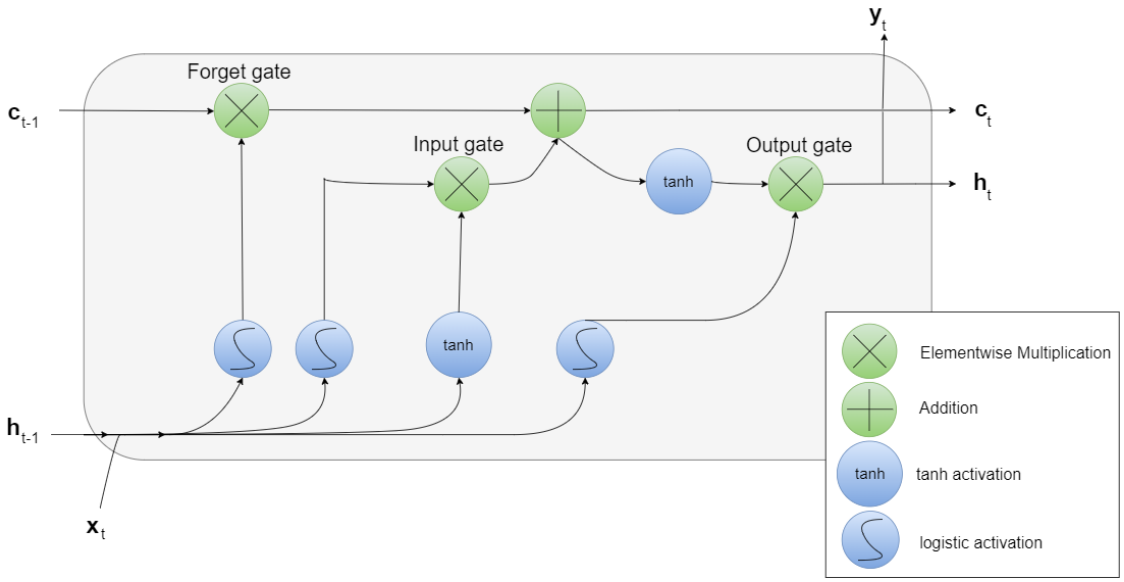

Figure 13: LSTM Cell

In contrast to a simple recurrent neuron, LSTM reuses two vectors: c_t and h_t. h_t is added with the new data x_t making it a short term memory. On the other side, c_t is multiplied with the new value making it a long term memory. The three gates regulate how much of the data is kept, forgot and delivered to the output. This design aims to recognize important input, and by using addition to the output of the input gate storing it in the long-term state. Additionally, by using the logistic regression and element-wise multiplication, it determines which elements of the long-term memory should be erased. And finally the output gate specifies which part of the new long-term memory is going to be output.
While most neural network architectures like MLP, CNN and simple RNN suffer from the vanishing gradient problem, where the weight updates in the back-propagation step becomes very small, LSTM cells overcome this problem due to its gates, especially the forget gate.
There have been different works using LSTM for univariate and multivariate time-series analysis. It's recurrent manner makes it an appropriate method for sequence data especially time-series. Additionally, most LSTM methods do not use a sequence of timestamp as input for the LSTM model as other approaches like MLP, CNN, ResNet did, as its long and short memory keeps the information of the recent timestamps. Hence, the input of the LSTM consist of just one timestamp, which accelerates the learning process.

Chauhan and Vig [19] use a LSTM model to predict healthy electrocardiography (ECG) signals. By using the probability distribution of the prediction error, it is able to mark timestamps as normal or anomalous.

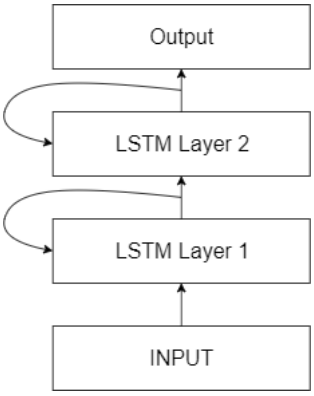

Figure 14: Stacked LSTM: The original LSTM model consisted of just one LSTM layer. Stacked LSTM has multiple LSTM layer

Malhotra et al. [60] use a stacked LSTM (Figure 14) model consisting of two hidden LSTM layers to predict the next l timestamps. Hence, the prediction window $p_w = l$ where $l > 1$. Let again $\{X_T\}$ be a univariate time-series with and $x_i \in \{X_T\}$, then:

$$(\hat{x}_{i+1}, ..., \hat{x}_{i+l}) = f(x_i) \tag{38}$$

$$\forall i \in \{1, ..., t\}, l < i < t - l, \text{ each } x_i \text{ is predicted l-times} \tag{39}$$

Then, the error value of each prediction is computed:

$$e^{(i)} = (e_1, ..., e_l) = (f(x_{i-l})_l - x_{i+1}, f(x_{i-l+1})_{l-1} - x_{i+1}, ..., f(x_i)_1 - x_{i+1}) \tag{40}$$

After that, the error vector is used to fit to a multivariate Gaussian distribution $\mathcal{N} = \mathcal{N}(\mu, \Sigma)$. By using Maximum Likelihood Estimation (MLE), the parameters μ and Σ can be computed, which enables it to give any $e^{(i)}$ a likelihood p^i. Finally given a anomaly threshold $\delta \in \mathbb{R}$:

$$x_i \text{ is an anomaly} \longrightarrow P(e^{(i)})_{e^{(i)} \sim \mathcal{N}(\mu, \Sigma)} > \delta \tag{41}$$

Like the methods we used for the MLP and CNN, we can use the training set to compute an appropriate value for δ.

6. **Gated recurrent unit (GRU)**
In 2014, Cho et al. [22] proposed a simplified version of the LSTM cell: Gated recurrent unit (GRU). GRU couples the input and forget gate into one *forget gate*. The state vectors c and h are merged into one vector h. Additionally, the output gate is removed and the full state vector is the output at every timestamp. Figure shows the architecture of GRU:

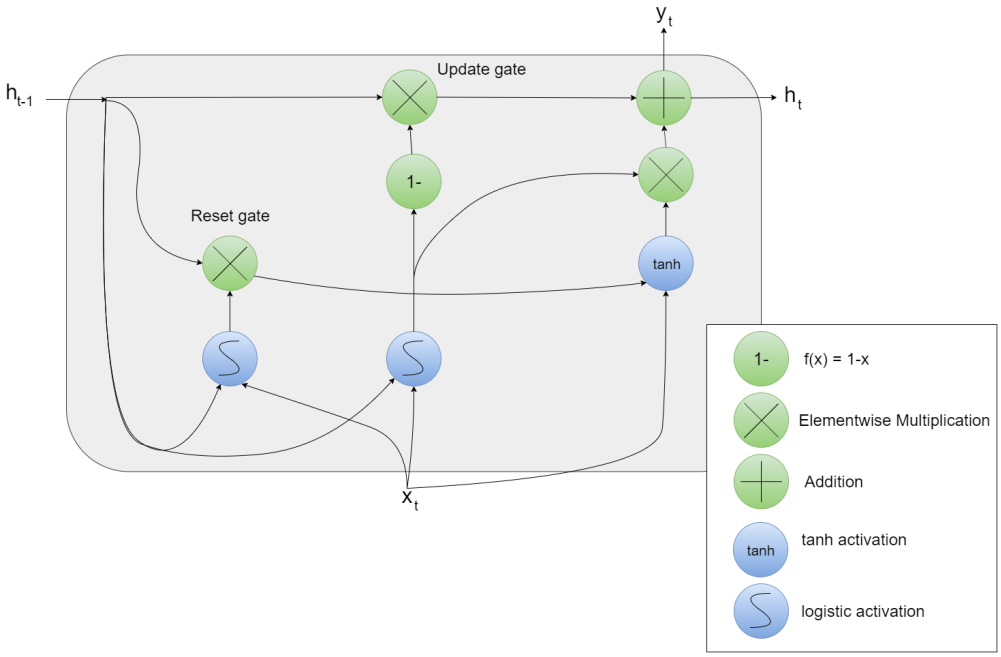

Figure 15: Gated recurrent unit (GRU)

Research has shown that GRUs perform as well as LSTMs, although requiring less computation due to its simplified structure [33].
GRUs have also been used for anomaly detection on time-series data. Wu et al. [93] used a stacked GRU model to detect anomalies on online time-series data.
In this work, we will also evaluate a GRU model to detect anomalies in univariate time-series. The model will be equal to the LSTM model for anomaly detection for time-series data we explained before. The only difference is that the LSTM cells will be replaced by GRU cells.

7. **Autoencoder**
One method to detect anomalies is to reduce the dimensionality of the data and to project it on a lower space, i.e., latent space, where more correlated variables remain. The main assumption about the distribution of data is the fact that normal and abnormal data are significantly different on this space, which the definition of anomalies (Definition 2.1) implies. Then, projecting back to the original space will show significant differences in some data points, which represent the anomalous data instances. This makes the autoencoder appropriate for anomaly detection.
Autoencoders belong to the family of feed-forward neural networks, and are optimized to output the same information that was inserted in the network. The challenge is that the first half of the hidden layers reduces the dimension of the dataset, and the second half increases the dimension back to the original value. These two parts are named accordingly the *Encoding* and *Decoding* part. Formally, let X be the dataset and ψ the decoding function and ϕ the encoding function and f the corresponding function of the autoencoder, then:

$$\hat{X} = \psi(\phi(X)) \tag{42}$$

The optimization function of the autoencoder tries to minimize the deviation between X and $\hat{X}$:

$$\min_{\theta_\psi, \theta_\phi} \left\| X - \hat{X} \right\|_2 = \min_{\theta_\psi, \theta_\phi} \| X - \psi(\phi(X)) \|_2 \tag{43}$$

where θ_ϕ and θ_ψ are the weights of the decoding and encoding part. Figure 16 shows the concept of an autoencoder graphically:

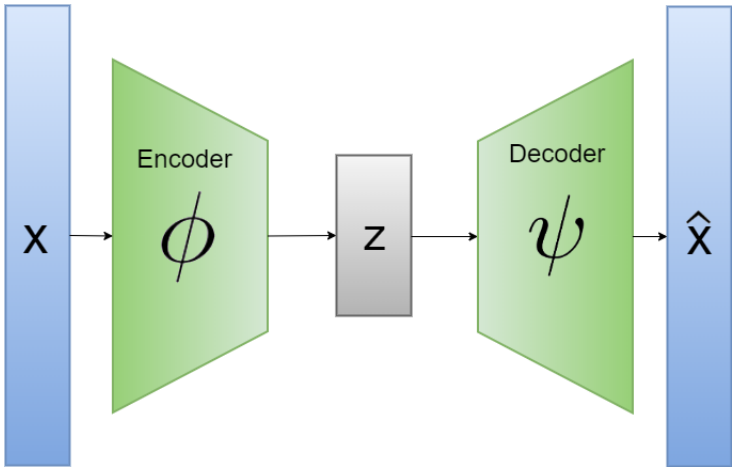

Figure 16: Autoencoder: The encoding layers ϕ reduce the dimension of x to z and the decoding layers ψ project the data back to the original dimensions resulting in $\hat{x}$

There have been different approaches using autoencoders to detect anomalies on spatial data. Zhou and Paffenroth [101] use robust autoencoders to detect anomalies on images. Baur et al. [7] use deep autoencoders to detect anomalies on 2D brain MR images.
Sakurada and Yairi [79] use an autoencoder to detect anomalies on time-series and compare it with linear and kernel PCA. They implement a normal autoencoder and a denoising autoencoder. Denoising autoencoders contaminate the input X with some noise and try to reproduce the noise-free input. Here in this paper, we will implement the normal autoencoder to detect anomalies on time-series. Thus, let $\{X_N\}$ be a univariate time-series and w be the width of the sliding window on our time-series, then the input for the autoencoder will be a vector $(x_i, x_{i+1}, ..., x_{i+w})^T \in \{X_N\}$. Then the autoencoder will compute the following:

$$(\hat{x}_i, \hat{x}_{i+1}, ..., \hat{x}_{i+w})^T = \psi(\phi((x_i, x_{i+1}, ..., x_{i+w})^T)) \tag{44}$$

Using the training set, the autoencoder tries to minimize the error using the following optimization function:

$$\min_{\theta_\psi, \theta_\phi} \left\| (x_i, x_{i+1}, ..., x_{i+w})^T - (\hat{x}_i, \hat{x}_{i+1}, ..., \hat{x}_{i+w})^T \right\|_2, \forall i \in \{i, ..., N-w\} \tag{45}$$

Then, the test set can be used to detect the anomalies. Let $f_{\psi\phi}$ be the trained autoencoder, for an $x_j \in \{X_{TEST}\}$ we first make a prediction:

$$\hat{x}_j = f_{\psi\phi}(x_j) = \psi(\phi(x_j)) \tag{46}$$

After that, the error value e_j for the prediction of x_j will be computed:

$$e_j = \|x_j - \hat{x}_j\| \tag{47}$$

Last but not least, having an anomaly threshold $\delta \in \mathbb{R}$, x_j is marked as abnormal if and only if $e_j > \delta$. As described in the previous approaches, the training data is used to determine an appropriate value for δ.
Autoencoders are best suited to semi-supervised learning approaches where the training data only consists of normal points. This results in learning a latent space of the normal data points and resulting in deviations when later an anomaly is fed into the model.

4 Approach

In this section, we first introduce related works regarding anomaly detection. After that, the different datasets used for the evaluations are mentioned. Additionally, the data preprocessing tasks are explained. At the end, the evaluation metrics applied are listed.

4.1 Related Works

Time-series analysis have been an important topic for a long time. In the last decades, while different machine learning approaches achieved noticeable progress in various areas, there has been much effort to benefit from them in time-series

analysis. There have been a lot of published works using machine learning to make better prediction. The Makridakis Competitions, also known as M-Competitions, are taking place regularly to evaluate different forecasting methods [58]. Afterwards, there have been attempts to compare the best performing approaches. Makridakis et al. [59] published an article where they compared the best performing statistical forecasting methods of the M-3 Competition with different machine learning approaches [57]. Their paper focuses on univariate time-series models.

There have also been different studies comparing ML methods but for a specific sort of data. For instance, Almaguer-Angeles et al. [3] compare 22 ML algorithms detecting anomalies on IoT-Datasets. There are also papers where the authors compare their anomaly detection approach with different approaches. Chakraborty et al. [16] compare their neural network approach with the ARMA model. Furthermore, there is also research, which compares anomaly detection methods, but only evaluates on a single dataset. Lazarevic et al. [52] compare different clustering techniques like Mining Outliers Using Distance to the k-th Nearest Neighbor, the Nearest Neighbor approach, Mahalanobis-distance Based Outlier Detection, the LOF approach and SVM, but using just one dataset. Munir et al. [64] compared their approach, FuseAD, with other state-of-the-art anomaly detection methods like LOF, iForest, OC-SVM, PCA, Twitter anomaly detection (TwitterAD) [73], and DeepAnT [63] on Yahoo Webscope dataset which is a time-series anomaly detection dataset.
To the best of our knowledge, there exists no extensive research comparing statistical approaches with classical machine learning approaches and neural networks detecting anomalies in univariate time-series. This is despite the fact that anomaly detection in time-series is becoming increasingly important. The most similar study to our approach is the work of Makridakis and Hibon [57] which compares univariate forecasting methods using statistical approaches and ML methods.
In this paper, we use different univariate time-series datasets to compare the univariate approaches that have been introduced in Section 3. All the approaches are supervised, thus using a part of the datasets for training and other part for testing.

4.2 Datasets

To evaluate the introduced univariate methods, several time-series datasets have been selected. Most of them are benchmarks for anomaly detection. One of the important criteria was that the dataset should indeed be a time-series dataset. There are different articles evaluating anomaly detection methods on time-series while choosing non-time series datasets like Forest Cover Type (ForestCover) dataset. Even the KDD Cup '99 dataset is critical, as it is not based on equal time intervals. Thus, we will only use a small portion of it to make it compatible with the time-series requirements.
Therefore, several univariate datasets consisting of real and synthetic data have been chosen. Furthermore, time-series data are preferred, which are also used in other studies enabling us to compare our results with theirs when using similar methods.

4.2.1 UD1 – Real Yahoo Services Network traffic

This dataset, which is published by Yahoo [94], is a univariate time-series dataset containing the traffic to Yahoo services. The anomalies are labeled by humans. This dataset consists of 67 different time-series each containing about 1400 timestamps. The timestamps have been observed hourly. Most of the time-series are stationary and on average each time-series consists of 1420 timestamps where 1.9% are anomalies.

4.2.2 UD2 – Synthetic Yahoo Services Network traffic

Yahoo made another dataset [94] available which consists of 100 synthetic univariate time-series data containing anomalies. Each time-series contains about 1421 timestamps. The anomalies have been inserted randomly therefore representing point anomalies. On average, each time-series consists of 0.3% anomalies.

4.2.3 UD3 – Synthetic Yahoo Services with Seasonality

This dataset also consists of 100 synthetic univariate time-series [94], each containing about 1680 timestamps. In contrast to the former one, this dataset also contains seasonality. The anomalies are inserted at random points marking the changing points. On average, the anomalous rate of each time-series is about 0.3%.

4.2.4 UD4 – Synthetic Yahoo Services with Changepoint Anomalies

The dataset also contains 100 synthetic univariate time-series. Each time-series has about 1680 timestamps. The difference to the former one is the fact that this dataset also contains changepoint anomalies where the mean of the

time-series changes. For our evaluation, we focus on the main anomalous points and ignore distinguishing between anomalous types. On average, 0.5% of the dataset is anomalous.

4.2.5 NYCT – NYC Taxi Dataset

This is a univariate time-series dataset containing the New York City (NYC) taxi demand from 2014-–07–01 to 2015–01–31 with an observation of the number of passengers recorded every half hour containing 10320 timestamps. It is from the Numenta Anomaly Benchmark (NAB), which is a benchmark for evaluating algorithms for anomaly detection, especially on streaming data. It contains five collective anomalies, which occur on the NYC marathon, Thanksgiving, Christmas, New Years day, and a snow storm.

4.3 Data Preprocessing

4.3.1 Standardize data

One of the main data preprocessing tasks performed on the datasets, before evaluating the anomaly detection methods, is standardizing. A dataset is standardized, if its mean μ is zero and its standard deviation σ is one. Thus, let $\mathcal{D}$ be the dataset and μ the mean of $\mathcal{D}$ and σ the standard deviation. Then, to standardize $\mathcal{D}$:

$$\hat{x} = \frac{x - \mu}{\sigma}, \forall x \in \mathcal{D} \tag{48}$$

Standardization is not equal to normalization, where $\mathcal{D}$ is changed, such that:

$$x \in [0, 1], \forall x \in \mathcal{D} \tag{49}$$

Normalization is sensitive to outliers. Thus, it would be inappropriate to normalize our datasets, which contain outliers. Standardization in Equation 48 helps many ML methods to converge faster. Shanker et al. [84] have shown that especially in smaller datasets, the neural network yields better results when the data is standardized. Ji et al. [48] mention in their paper that standardizing time-series data is necessary for the SVM approach. Also other approaches which are evaluated in this paper like DBSCAN, K-Means, and K-NN benefit from standardization.

4.3.2 Deseasonalizing and Detrending

Most statistical approaches for time-series analysis presume stationarity in mean and variance. But for machine learning approaches and neural networks, this is a debatable requirement. There are studies neglecting the need for transforming the time-series into a stationary sequence. Neural networks are universal function approximators and therefore several researchers claim that they are able to detect non-stationary trends. Therefore, Gorr [32] believes that they are able to detect non-linear trends and seasonality in the time-series dataset. Sharda and Patil [85] show empirically, that neural networks are able to detect automatically the seasonality in univariate time-series by evaluating about 101 different time-series. There have also been other works achieving similar results [27, 87].

There have also been studies achieving contradicting results. Faraway [26] compared models using detrended and deseasonalized time-series to networks using the origin data. They show empirically, that the model using detrended and deseasonalized data achieve better results. Zhang and Qi [97] make an empirical study comparing the performance of neural networks using detrending and deseasonalizing performs better. In particular, they found out that the combination of both achieves the best results concluding that NN are not able to detect seasonal and trend variations effectively. Therefore, to get more precise results when comparing different approaches, we analyze the performance of the methods using the raw time-series dataset (except for standardization) compared to detrended and deseasonalized time-series.

4.4 Evaluation Metrics

4.4.1 F-Score

To compare the different anomaly detection methods, several metrics can be taken into consideration. One of the metrics used in similar approaches is the F-Score:

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{50}$$

Munir et al. [63] use this metric to evaluate their anomaly detection methods on different time-series. Maya et al. [62] also use the F-Measure in addition to recall and precision.

4.4.2 Area under the curve (AUC)

Another metric that is often used is the *receiver operating characteristic curve*, ROC-Curve, and the associated metric *area under the curve (AUC)*, which is the area under the ROC-Curve. This measure is very helpful, especially for anomaly detection. The ROC-Curve illustrates the correlation of the *true positive rate* and the *false positive rate* based on different threshold values. The true positive rate (TPR) is defined as follows:

Let P be the positive labeled values, e.g., timestamps which are actually anomalous, N the negative labeled values, e.g., timestamps which are actually normal, TP be the true positive classifications, i.e., timestamps which are anomalous and have been detected as anomalous by the algorithm, FP the false positive, i.e., timestamps which are normal but have been labeled falsely as anomalies, then TPR is:

$$TPR = \frac{TP}{P} \tag{51}$$

and FPR is:

$$FPR = \frac{FP}{N} \tag{52}$$

To compute the ROC-Curve, we use different $\delta \in \mathbb{R}$ as threshold for our anomaly detection method resulting in different pairs of TPR and FPR for each δ. This values can finally be plotted showing a curve starting at the origin and ending in the point (1,1). The associated metric AUC is the area under the curve. In anomaly detection, the AUC expressed the probability that the measured algorithm assigns a random anomalous point in the time-series a higher anomaly score than a random normal point. This makes AUC appropriate to compare the different anomaly detection methods. Several other papers [79, 61, 4, 52] also used the ROC-Curve and AUC-Value to compare different anomaly methods. For instance, Goldstein and Uchida [31] used this measure to compare different machine learning anomaly detection methods.

In a nutshell, the AUC-value will be the main measure used in the experiments of this paper in Section 6.

4.4.3 Computation Time Comparison

Another factor to evaluate the different approaches, is the computation time a method uses to analyse the data. The time an algorithm needs to predict if a timestamp is abnormal or not plays an essential role in the success of the approach. There are many use cases where the inference time is crucial. For instance, to detect online anomalies the approach must be able to respond quickly if new data points appear rapidly.
To measure the computation time, different approaches are possible, based on the nature of the algorithms. For instance, Maya et al. [62] computed the time used for training and inference separately. Goldstein and Uchida [31] computed the time the unsupervised anomaly detection methods needed to analyse a dataset.

The method to measure the time performance in this paper is different to the approaches used in the aforementioned papers. This is based on the fact, that those papers compare similar detection methods. For instance, Goldstein and Uchida [31] just compares machine learning approaches, especially clustering and density approaches. Also, Maya et al. [62] compares deep learning methods making it possible to measure the training and inference phase separately. This is not possible when comparing deep learning methods like LSTM with a classical machine learning method like LOF. While LSTM mostly needs a big amount of time in training, its inference time is very fast. This is completely different to clustering methods such as LOF.
Hence, the time performance of the algorithm is mainly compared based on the total training and prediction time. This will provide a relative value to compare the approaches with each other.

5 Experiments

In this section, the settings for the experiments are listed. This is necessary to evaluate the results of the different approaches.

5.1 Datasets

The datasets are divided into training and test set where 30% are used for training and 70% as test data. If the test data does not contain any anomalous point, the time-series will be ignored and excluded from the evaluation. Additionally, the data is standardized before being used by the methods as explained in Section 4. The test data forms the source of the method evaluation. The AUC-Value is computed on the test data while the train data is only used for training. For some deep learning methods, 10% of the training data is used as validation set to optimize the hyperparameters.

5.2 Experimental Setup

The main requirement for the evaluation is achieving an optimally trained model for each anomaly detection method. This requires to optimize the hyperparameters of the model in addition to the trainable parameters. Therefore, a validation set is used. This prevents overfitting in the training process.
In addition to that and especially for the anomaly detection methods, which are based on a forecasting model, hyperparameter tuning is performed on the forecasting model. An optimized forecasting model will achieve better results later in detecting anomalies. To evaluate the forecasting model, a *naive model* is created. In some literature this naive model is also called *persistent model*. The naive model in time-series forecasting is a model where the output $\hat{x}_t$ of the model is equal to the input:

$$\begin{aligned} \hat{x}_t =& f_{NaiveModel}((x_{t-w-1}, ..., x_{t-1})) \\ where\ f_{NaiveModel} &: \mathbb{R}^w \to \mathbb{R} \\ & f_{NaiveModel}((x_{t-w-1}, ..., x_{t-1})) \mapsto x_{t-1} \end{aligned} \tag{53}$$

The accuracy of the naive model forms a lower bound for the target model. Following this, the Mean Squared Error (MSE) of the target model should be lower than the naive model. Therefore, this paper suggests the following metric to evaluate the forecasting performance of the prediction model:

$$\text{NMM}(MSE_{Targetmodel}, MSE_{NaiveModel}) = \frac{MSE_{Targetmodel}}{MSE_{NaiveModel}} \tag{54}$$

where NMM stands for *Naive Model Metric*. Empirically, we have noticed that a model usually has a higher AUC-Value, when the NMM is lower. This has been especially useful when the dataset is unlabeled or the rate of anomalies in the training set is extremely low. Therefore, the hyperparameters of the model are tuned to minimize the NMM value aiming to keep it strictly lower than 1.

5.2.1 Involved Software and Hardware

5.2.1.1 Software

The algorithms are implemented in python. For the statistical approaches, mainly the module **Statsmodels** [83] was used, while some custom algorithms like PCI were implemented from scratch. The classical machine learning approaches have been mostly implemented using **Scikit-learn (Sklearn)** [72] and for the deep learning approaches the **tensorflow** [1] and **keras** library [23] have been used.

5.2.1.2 Hardware

The training and testing of the models were performed on the hardware listed in Table 1:

Artifact	Value
CPU Model name:	2xIntel(R) Xeon(R) CPU @ 2.30GHz, 46MB Cache, 1 Core
RAM	~ 12.4 GB
GPU	1xTesla K80, 12GB

Table 1: Hardware specification

All computations have been performed on a single process and a single thread.

5.2.2 Hyperparameter Tuning

In the following, we describe the general and the specific hyperparameters for the evaluation and the respective methods.

5.2.2.1 General Hyperparameters

Some of the hyperparameters are general and dependent on the dataset, unless another value is listed explicitly for a specific approach. Table 2 lists these hyperparameters:

Dataset	Hyperparameter	Value
Yahoo Services Network traffic	Sliding window width w	30
All datasets (NYCT and UD1 - UD4)	Ratio of training set	0.3
	Ratio of test set	0.7
	Ratio of validation set	0.1 of test set

Table 2: General Hyperparameters

5.2.2.2 Statistical approaches

Table 3 lists the hyperparameters of the statistical approaches:

Statistical Approaches		
Model	Hyperparameter	Value
AR-Model	maximal lag	$12 \cdot (\frac{\lvert X_{Train} \rvert}{100})^{\frac{1}{4}}$
	Fitting method	Conditional maximum likelihood using OLS
MA-Model	maximal lag	$12 \cdot (\frac{\lvert X_{Train} \rvert}{100})^{\frac{1}{4}}$
	error residual lag	Sliding window width w
	Fitting method	Conditional maximum likelihood using OLS
ARIMA-Model	maximal lag	$12 \cdot (\frac{\lvert X_{Train} \rvert}{100})^{\frac{1}{4}}$
	p	1
	d	1 if data contains trend, otherwise 0
	q	2
	Fitting method	Maximizing Conditional Sum of Squares likelihood
	Maximal iteration	500
	Convergence tolerance	10^{-8}
SES	Smoothing Parameter α	Use grid search to find the best values using MLE
ES	α, β, γ	Use grid search to find the best values using MLE
PCI	k	30
	α	98.5

Table 3: Hyperparameters of the statistical approaches

5.2.2.3 Machine Learning approaches

Table 4 lists the hyperparameters of the machine learning approaches:

Machine Learning Approaches		
Model	Hyperparameter	Value
K-Means	k	4
DBSCAN	ϵ	0.4
	μ	5
	distance function	Euclidean distance
LOF	k	10
	distance function	Minkowski distance
Isolation Forest (iForest)	Number of iTrees	10
	Contamination value	Find automatically
One-Class SVM (OC-SVM)	kernel	Radial basis function kernel(RBF)
	Upper bound of outliers	0.7
XGBoosting (XGB)	Max Tree depth	3
	Learning rate	0.1
	Number of estimators	1000
	Loss function	MSE

Table 4: Hyperparameters of the Machine Learning approaches

5.2.2.4 Deep Learning approaches

Table 5 lists the hyperparameters of the deep learning approaches:

Deep Learning Approaches		
Model	Hyperparameter	Value
MLP	Number of hidden layers	2
	Neurons in each hidden layer	100, 50
	Activation function	ReLU
	Optimizer, Loss	Adam, MSE
	Batch size,Epochs	32, 50
CNN	Architecture	3 Convolution Blocks with Max-Pooling and ReLU, then one Dense layer with 50 neurons and ReLU
	Filters	8,16,32 with kernel size 2
	Optimizer, Loss	Adam, MSE
	Batch size, Epochs	32, 50
CNN+Batch Normalization (CNN_B)	Architecture	2 Convolution Blocks with Batch-Normalization with ReLU then one Dense layer with 50 neurons and ReLU
	Filters	256,256 with kernel size 3
	Optimizer, Loss	Adam, MSE
	Batch size, Epochs	32,50
CNN+Residual Blocks (CNN Residual)	Architecture	1 Convolution Block with One Residual Block then one Dense Layer with 50 neurons and ReLU
	Filters	256,256,256 with kernel size 3
	Optimizer, Loss	Adam, MSE
	Batch size, Epochs	32, 50

WaveNet	Architecture	3 Convolution Blocks with Max-Pool and ReLU function, with Dilation rate 1,2,4 then one Dense layer with 50 neurons and ReLU
	Filters	8, 16, 32
	Optimizer	Adam, MSE
	Batch size, Epochs	32, 50
LSTM	Architecture	2 stateful LSTM Layer
	Filters	4,4
	Optimizers, Loss	Adam, MSE
	Batch size, Epochs	32, 50
GRU	Architecture	2 stateful LSTM Layer
	Filters	4,4
	Optimizers, Loss	Adam, MSE
	Batch size, Epochs	32, 50
Autoencoder	Architecture	2 Encoding Layers (32,16), 2 Decoding Layers (16,32)
	Activation functions	ReLU for Decoding and Encoding, linear for output
	Optimizer, Loss	Adam, MSE
	Batch size, Epochs	32, 50

Table 5: Hyperparameters of the Deep Learning approaches

6 Results

In this Section, the results of the different approaches are presented. The approaches from Section 3 with the hyperparameters listed in Section 5 have been executed. The following sections list the AUC-Values and computation time, respectively.

6.1 AUC-Values

Table 6 lists the AUC-Values obtained on the different datasets by the univariate approaches. The AUC-Value was computed on each dataset separately:

Dataset	AR	MA	SES	ES	ARIMA	PCI		
UD1	**0.911394**	0.868123	0.824894	0.830289	0.8730	0.522397		
UD2	0.994769	0.994150	0.932215	0.957964	0.9891	0.762529		
UD3	0.994116	**0.994245**	0.990782	0.989360	0.990	0.674337		
UD4	0.975152	**0.986400**	0.969333	0.971991	0.9709	0.688257		
NYCT	0.6369	0.3938	0.3452	0.3423	0.3583	0.5377		
Dataset	K-Means	DBSCAN	LOF	iForest	OC-SVM	XGBoost		
UD1	0.877623	0.806574	0.814574	0.803997	0.850292	0.896743		
UD2	0.923446	0.995251	0.995116	0.993984	**0.995276**	0.968308		
UD3	0.728037	0.696868	0.951007	0.952241	0.957272	0.80231		
UD4	0.663028	0.725566	0.952523	0.955123	0.939444	0.85375		
NYCT	**0.9137**	0.5407	0.5294	0.4922	0.5859	0.4602		
Dataset	MLP	CNN	CNN B	CNN Residual	Wavenet	LSTM	GRU	Autoencoder
UD1	0.780471	0.809449	0.785505	0.819109	0.8239	0.8121	0.8025	0.782197
UD2	0.71911	0.74924	0.74815	0.721279	0.761423	0.7348	0.7183	0.742767
UD3	0.569883	0.582761	0.581911	0.575338	0.579589	0.5781	0.5711	0.602905
UD4	0.558124	0.581019	0.604943	0.568564	0.592414	0.5891	0.5811	0.597238
NYCT	0.7962	0.8181	0.76	0.7468	0.8229	0.8404	0.7978	0.6967

Table 6: AUC-Values on each dataset by each approach

The best results are highlighted in table 6. To provide a better illustration of the results, the average AUC-Value of all datasets UD1-UD4 for each algorithm is computed and plotted as a sorted sequence in figure 17:

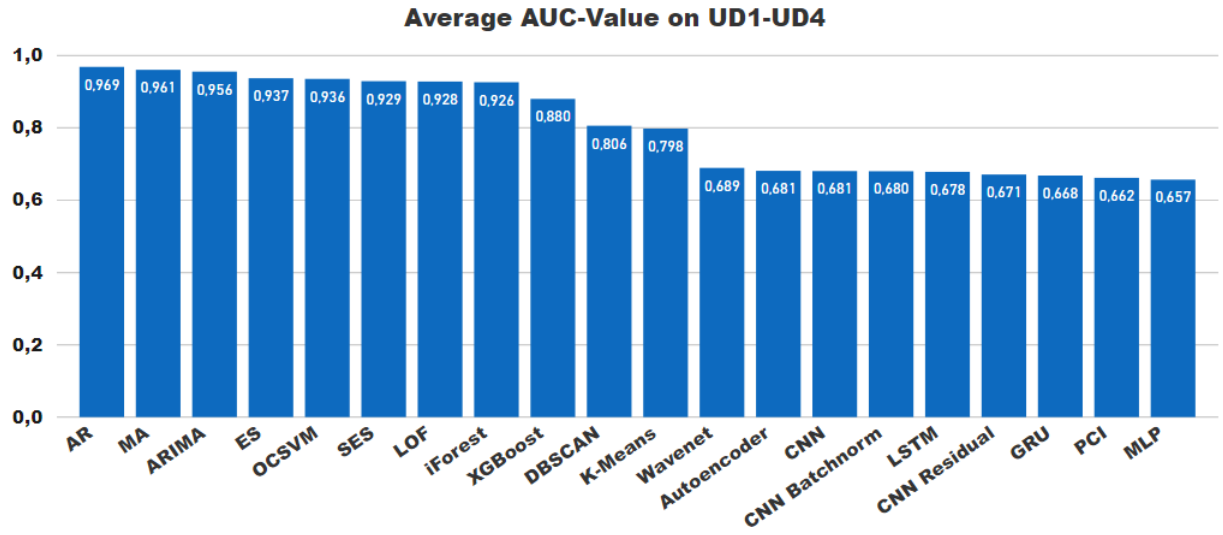

Figure 17: Average AUC-Value computed on UD1-UD4

Figure 17 shows that the statistical models achieved the best results while the deep learning methods generally performed poorly. Four of the five best performing algorithms are statistical while four of the worst performing algorithms are deep learning approaches. Most of the machine learning approaches are located in the center. Only PCI is an exception of the statistical approaches, which reached a very low AUC-value.
However, surprisingly the deep learning methods perform much better on the NYCT dataset while the statistical approaches achieve a very low AUC-Value. Figure 18 shows how the different approaches behave on the NYCT dataset:

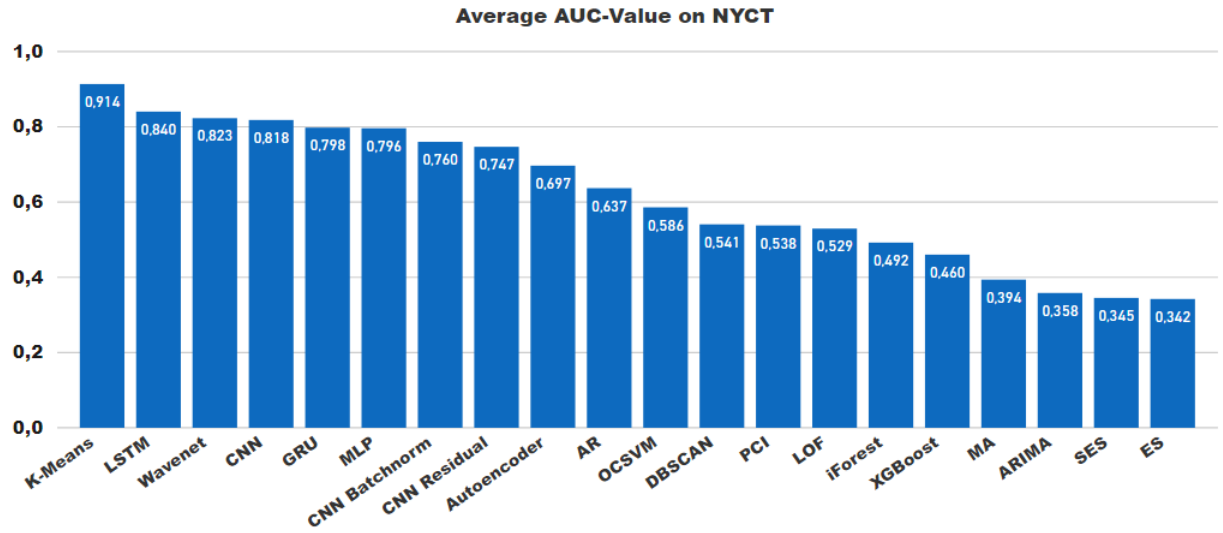

Figure 18: Average AUC-Value computed on UD1-UD4

The main reason to this poor performance of the statistical approach is the fact that the NYC time-series contains contextual anomalies while the anomalies in UD1, UD2, UD3 and UD4 are either point anomalies or collective anomalies. Most of them do not contain any contextual anomalies. The values of the anomalous points differ clearly from the rest normal points. But in the NYCT dataset the value of the anomalous points is similar to the normal points. They are anomalous due to their contextual information. We observed that the statistical models overfit to these data,

preventing them to detect the anomalies. On the other side, deep learning approaches provide a more flexible way to optimize the model according to the anomaly type by tuning the broad range of hyperparameters making them able to achieve high AUC values.

6.2 Computation Time

The computation time is measured for each algorithm on the datasets UD1-UD4 containing 367 time-series which is listed in the first line of the following table. Then the average time needed to train and detect anomalies on a single time-series is computed and listed in the second line. The time is measured in seconds.

Dataset	AR	MA	SES	ES	ARIMA	PCI		
UD1-4	51	17	2910	13268	72422	374		
One time-series	**0.139**	0.046	7.93	36.15	197.3351	1.02		
Dataset	K-Means	DBSCAN	LOF	iForest	OC-SVM	XGBoost		
UD1-4	190	319	319	38451	210	14		
One time-series	0.52	0.87	0.87	104.77	0.57	0.38		
Dataset	MLP	CNN	CNN B	CNN Residual	Wavenet	LSTM	GRU	Autoencoder
UD1-4	7396	10824	12938	37038	17252	54083	49756	9413
One time-series	20.15	29.49	35.38	100.92	47.0	147.36	135.57	25.65

Table 7: Computation Time of the different approaches on UD1-UD4 datasets

The results in Table 7 show that the autoregression models AR and MA are the fastest algorithms while in general the deep learning methods have a huge computation time. But not all statistical approaches benefit from a lower runtime. SES and ES require more time than many machine learning approaches. Deep learning approaches have higher computation time because of their time-consuming training phase. Especially, LSTM and GRU are the bottom of the table.
Some papers argue that deep learning approaches invest most of their computation time in the training phase and that the inference time in deep learning approaches is much faster than other machine learning methods. But this statement is not always true as table 8, which contains the measured training and inference time on the NYCT dataset, demonstrates:

	AR	MA	SES	ES	ARIMA	PCI		
Training Time	0.1004	0.1404	125.26	989.49	1482.87	2.85		
Inference Time	0.1004	0.6423	5.06	9.5115	1481.56	0.0093		
Dataset	K-Means	DBSCAN	LOF	iForest	OC-SVM	XGBoost		
Training Time	0.25845	0.0879	0.0917	5.5646	0.02	0.1187		
Inference Time	1.99455	8.3774	5.5646	787.29	2.98	0.0205		
Dataset	MLP	CNN	CNN B	CNN Residual	Wavenet	LSTM	GRU	Autoencoder
Training Time	4.439	48.322	39.2339	111.36	23.263	1067	813.021	17.459
Inference Time	0.561	0.6783	0.7661	1.6401	0.737	1086	385.979	0.5406

Table 8: Training and Inference compuatation time on the NYCT dataset

The training time represents the computation time we measured by fitting the model on the trainings set which consists of 30% of the NYCT dataset. The inference time was the measured time, the model needed to label the rest of the NYCT dataset which represents the test set.
Most of the deep learning approaches have a very low inference time. But as the measured values in the table show, recurrent neural network like LSTM and GRU also have a very huge inference time exceeding the inference times of almost all machine learning approaches. Hence, although the LSTM model achieved the best results on the NYCT dataset, but its inference time remains critical.

6.3 Computation Time vs AUC-Value

An interesting relation is the accuracy a method can achieve compared to the computation time it requires. Figure 19 displays how the different approaches perform in regard to AUC-Value and Computation time on the datasets UD1-UD4. The best performing algorithms would locate at the lower right part of the graph having a high AUC-Value and a low computation time. The graph shows that the statistical methods are performing best, i.e., AR and MA model. On the other hand, most deep learning methods are showing low AUC-Value. CNN with residual blocks is performing worse than any other method having low time performance and low AUC-Value at the same time.

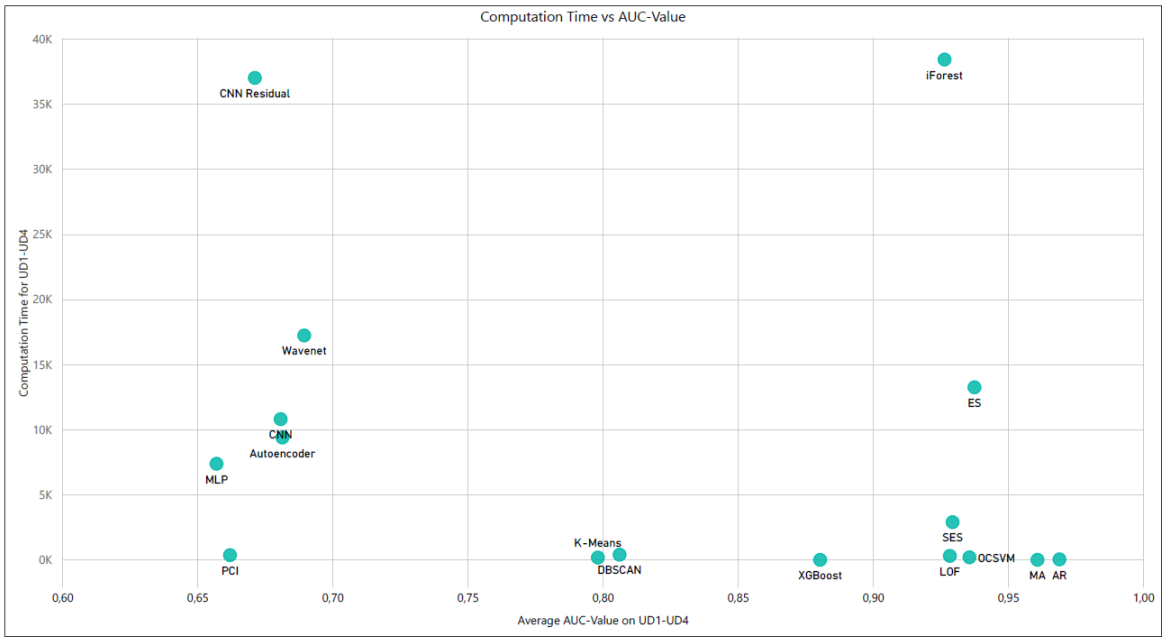

Figure 19: Average AUC-Value vs computation time

6.4 Findings

The evaluation on the univariate time-series data has shown that despite the advances in machine learning approaches and deep neural networks, still the statistical methods that rely on the generating process are generally performing best. In addition to the fact that these methods detect anomalies more accurately than the other approaches, they also perform faster and have a very low training and prediction duration. Furthermore, the optimization of the statistical methods is much easier compared to deep learning methods, because the number of hyperparameters of the statistical algorithms is low. The results have also highlighted the fact that in time-series anomaly detection the machine learning approaches usually perform better than the deep learning methods, although the computation time of several of them is critical. For instance, the isolation forest has a very high AUC-value, but needs a thousand times more time for training and inference than the AR and MA model. This is despite the fact that the number of trees selected in our experiments have been quite moderate. But at the same time, the experiments have also shown that statistical approaches have trouble dealing with contextual anomalies. In cases like this, neural network approaches and some machine learning approaches could achieve notably higher AUC-values. This finding is important, because a similar comparison of statistical approaches and machine learning approaches for forecasting univariate time-series concluded that statistic approaches dominated the machine learning and deep learning approaches across in both accuracy and for all forecasting horizons they examined [59]. Hence, these findings show how anomaly detection on univariate time-series data differ from exclusive univariate forecasting.

7 Conclusion and Future Work

In this paper a comparison between statistical and classical machine learning and deep learning approaches has been performed using a wide range of state of the art algorithms. Overall, we evaluated 20 univariate anomaly detection

methods by using five univariate time-series datasets consisting of 368 univariate time-series datasets to provide a reliable contribution for the research community about the performance of these three classes of anomaly detection methods.
The experiments showed that the statistical approaches perform best on univariate time-series by detecting point and collective anomalies. They also require less computation time compared to the other two classes. Although deep learning approaches have gained huge attention by the artificial intelligence community in the last years, our results have revealed that they are not generally able to achieve the accuracy values of the statistical methods on the univariate time-series benchmarks which only consist of point and collective anomalies. Only if the univariate dataset mainly consists of contextual anomalies, neural network could outperform the statistical methods. Although still suffering from high computation time compared to statistical approaches, some of the deep learning methods could outperform them by achieving higher AUC-rates.
The main focus of this paper has been univariate time-series. However, many real data sets consist of multivariate data. Therefore, anomaly detection on multivariate time-series will be the subject of future research. Additionally, one related field, which is also of current interest, is Online Anomaly Detection. This paper focused on anomaly detection on static data, but can be extended to evaluate streaming data.
In this paper, we evaluated the anomaly detection methods on general time-series datasets and the results have shown that the property of the data affects the performance of the algorithms. Therefore, it would be a line of future works to extend the experiments of this paper by evaluating the approaches on different domain-specific datasets.
Finally, several attempts can be made to extend the results of this paper. However, to the best of our knowledge, this is the first attempt to provide a broad evaluation of different anomaly detection techniques on time-series data. We hope that this paper and the corresponding experiments aid other researchers in selecting an appropriate anomaly detection method when analysing time-series.

References

[1] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] Charu C. Aggarwal. *Outlier Analysis*. Springer Publishing Company, Incorporated, 2nd edition, 2016. ISBN 3319475770.

[3] F. Almaguer-Angeles, J. Murphy, L. Murphy, and A. O. Portillo-Dominguez. Choosing Machine Learning Algorithms for Anomaly Detection in Smart Building IoT Scenarios. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 491–495, 04 2019. doi: 10.1109/WF-IoT.2019.8767357.

[4] Tsatsral Amarbayasgalan, Bilguun Jargalsaikhan, and Keun Ryu. Unsupervised Novelty Detection Using Deep Autoencoders with Density Based Clustering. *Applied Sciences*, 8, 08 2018. doi: 10.3390/app8091468.

[5] Jimmy Ba and Rich Caruana. Do Deep Nets Really Need to be Deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf`.

[6] A. D. Back and A. C. Tsoi. FIR and IIR Synapses, a New Neural Network Architecture for Time Series Modeling. *Neural Computation*, 3(3):375–385, 10 1991. doi: 10.1162/neco.1991.3.3.375.

[7] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. Deep Autoencoding Models for Unsupervised Anomaly Segmentation in Brain MR Images. *CoRR*, abs/1804.04488, 2018. URL `http://arxiv.org/abs/1804.04488`.

[8] S. Behera and R. Rani. Comparative analysis of density based outlier detection techniques on breast cancer data using hadoop and map reduce. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 2, pages 1–4, 08 2016. doi: 10.1109/INVENTIVE.2016.7824883.

[9] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-parameter Optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11,

pages 2546–2554, USA, 2011. Curran Associates Inc. ISBN 978-1-61839-599-3. URL http://dl.acm.org/citation.cfm?id=2986459.2986743.

[10] Anastasia Borovykh, Sander M. Bohte, and Cornelis W. Oosterlee. Conditional time series forecasting with convolutional neural networks. In *Conditional time series forecasting with convolutional neural networks*, 2017.

[11] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA, 1990. ISBN 0816211043.

[12] Paul S. Bradley and Usama M. Fayyad. Refining Initial Points for K-Means Clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 91–99, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8. URL http://dl.acm.org/citation.cfm?id=645527.657466.

[13] Leo Breiman. Statistical Modeling: The Two Cultures. *Statistical Science*, 2001.

[14] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying Density-based Local Outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000. ISSN 0163-5808. URL http://doi.acm.org/10.1145/335191.335388.

[15] R.G. Brown. *Exponential Smoothing for Predicting Demand*. Little, 1956. URL https://books.google.de/books?id=Eo_rMgEACAAJ.

[16] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. Forecasting the Behavior of Multivariate Time Series Using Neural Networks. *Neural Networks*, 5(6):961–970, 1992. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80092-9. URL http://www.sciencedirect.com/science/article/pii/S0893608005800929.

[17] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–72, 2009. ISSN 03600300. doi: 10.1145/1541880.1541882.

[18] Ih Chang, George Tiao, and Chung Chen. Estimation of time series parameters in the presence of outliers. *Technometrics*, 30:193–204, 05 1988. doi: 10.1080/00401706.1988.10488367.

[19] S. Chauhan and L. Vig. Anomaly detection in ECG Time Signals via Deep Long Short-Term Memory Networks. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–7, 10 2015. doi: 10.1109/DSAA.2015.7344872.

[20] Jason R. Chen. Making Subsequence Time Series Clustering Meaningful. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 114–121, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2278-5. URL https://doi.org/10.1109/ICDM.2005.91.

[21] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL http://arxiv.org/abs/1603.02754.

[22] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

[23] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[24] Zhiguo Ding and Minrui Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, 46(20):12–17, 2013. ISSN 1474-6670. doi: https://doi.org/10.3182/20130902-3-CN-3020.00044. URL http://www.sciencedirect.com/science/article/pii/S1474667016314999. 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013.

[25] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996. URL http://dl.acm.org/citation.cfm?id=3001460.3001507.

[26] Julian J. Faraway. *Time Series Forecasting With Neural Networks : A Comparative Study Using the Airline Data*, 1998.

[27] Philip Hans Franses and Gerrit Draisma. Recognizing Changing Seasonal Patterns Using Artificial Neural Networks. *Journal of Econometrics*, 81(1):273–280, 1997. ISSN 0304-4076. doi: https://doi.org/10.1016/S0304-4076(97)00047-X. URL http://www.sciencedirect.com/science/article/pii/S030440769700047X.

[28] Ellen Friedman and Ted Dunning. Practical Machine Learning: A New Look at Anomaly Detection, 2014.

[29] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[30] Ryohei Fujimaki, Shunsuke Hirose, and Takayuki Nakata. *Theoretical Analysis of Subsequence Time-Series Clustering from a Frequency-Analysis Viewpoint*, pages 506–517. SIAM, 2008. doi: 10.1137/1.9781611972788.46. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611972788.46.

[31] Markus Goldstein and Seiichi Uchida. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLOS ONE*, 11(4):1–31, 04 2016. doi: 10.1371/journal.pone.0152173. URL https://doi.org/10.1371/journal.pone.0152173.

[32] Wilpen L. Gorr. Editorial: Research Prospective on Neural Network Forecasting. *International Journal of Forecasting*, 10(1):1–4, 1994. ISSN 0169-2070. doi: https://doi.org/10.1016/0169-2070(94)90044-2. URL http://www.sciencedirect.com/science/article/pii/0169207094900442.

[33] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *CoRR*, abs/1503.04069, 2015. URL http://arxiv.org/abs/1503.04069.

[34] Frank E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969. doi: 10.1080/00401706.1969.10490657. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1969.10490657.

[35] Nikou Günnemann, Stephan Günnemann, and Christos Faloutsos. Robust multivariate autoregression for anomaly detection in dynamic product ratings. In *WWW*, 2014.

[36] Manish Gupta, Jing Gao, Charu Aggarwal, and Jiawei Han. *Outlier Detection for Temporal Data*. Morgan and Claypool Publishers, 2014. ISBN 1627053751.

[37] E. Haselsteiner and G. Pfurtscheller. Using Time-Dependent Neural Networks for EEG Classification. *IEEE Transactions on Rehabilitation Engineering*, 8(4):457–463, 12 2000. doi: 10.1109/86.895948.

[38] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 edition, 2009. URL http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

[39] D. M. Hawkins. *Identification of Outliers*. Monographs on Applied Probability and Statistics. Chapman and Hall, London [u.a.], 1980. ISBN 041221900X. URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+02435757X&sourceid=fbw_bibsonomy.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[41] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003. ISSN 0167-8655. doi: https://doi.org/10.1016/S0167-8655(03)00003-5. URL http://www.sciencedirect.com/science/article/pii/S0167865503000035.

[42] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[43] R.J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2014. ISBN 9780987507105. URL https://books.google.de/books?id=gDuRBAAAQBAJ.

[44] Rob J Hyndman, Anne B Koehler, Ralph D Snyder, and Simone Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, 18(3):439–454, 2002. ISSN 0169-2070. doi: https://doi.org/10.1016/S0169-2070(01)00110-8. URL http://www.sciencedirect.com/science/article/pii/S0169207001001108.

[45] Tsuyoshi Idé. Why Does Subsequence Time-Series Clustering Produce Sine Waves? In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Knowledge Discovery in Databases: PKDD 2006*, pages 211–222, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46048-0.

[46] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[47] Tim Januschowski, Jan Gasthaus, Yuyang Wang, David Salinas, Valentin Flunkert, Michael Bohlke-Schneider, and Laurent Callot. Criteria for Classifying Forecasting Methods. *International Journal of Forecasting*, 36 (1):167 – 177, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.05.008. URL http://www.sciencedirect.com/science/article/pii/S0169207019301529. M4 Competition.

[48] G. Ji, P. Han, and Y. Zhai. Wind Speed Forecasting Based on Support Vector Machine with Forecasting Error Estimation. In *2007 International Conference on Machine Learning and Cybernetics*, volume 5, pages 2735–2739, 08 2007. doi: 10.1109/ICMLC.2007.4370612.

[49] Eamonn Keogh and Jessica Lin. Clustering of Time-Series Subsequences is Meaningless: Implications for Previous and Future Research. *Knowledge and Information Systems*, 8(2):154–177, 08 2005. ISSN 0219-3116. doi: 10.1007/s10115-004-0172-7. URL https://doi.org/10.1007/s10115-004-0172-7.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[51] Vipin Kumar, Arindam Banerjee, and Varun Chandola. Anomaly detection for symbolic sequences and time series data, 2009.

[52] Ar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. In *In Proceedings of SIAM Conference on Data Mining*, 2003.

[53] Fei Tony Liu, Kai Ming Ting, and Zhi hua Zhou. Isolation forest. In *In ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. IEEE Computer Society*, pages 413–422, 2008.

[54] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1):3:1–3:39, March 2012. ISSN 1556-4681. URL http://doi.acm.org/10.1145/2133360.2133363.

[55] J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1741–1745 vol.3, 07 2003. doi: 10.1109/IJCNN.2003.1223670.

[56] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200512992.

[57] Spyros Makridakis and Michèle Hibon. The M3-Competition: Results, Conclusions and Implications. *International Journal of Forecasting*, 16(4):451–476, 2000. ISSN 0169-2070. doi: https://doi.org/10.1016/S0169-2070(00)00057-1. URL http://www.sciencedirect.com/science/article/pii/S0169207000000571. The M3- Competition.

[58] Spyros Makridakis, Allan Andersen, Robert F. Carbone, Robert Fildes, Michèle Hibon, Rudolf Lewandowski, Jonathan Newton, Emanuel Parzen, and Robert L. Winkler. The Accuracy of Extrapolation (Time Series) Methods: Results of a Forecasting Competition. *International Journal of Forecasting*, 1982.

[59] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3):1–26, 03 2018. doi: 10.1371/journal.pone.0194889. URL https://doi.org/10.1371/journal.pone.0194889.

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

[61] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *ESANN*, 04 2015.

[62] Shigeru Maya, Ken Ueno, and Takeichiro Nishikawa. dLSTM: A New Approach for Anomaly Detection Using Deep Learning with Delayed Prediction. *International Journal of Data Science and Analytics*, 8(2):137–164, 09 2019. ISSN 2364-4168. doi: 10.1007/s41060-019-00186-0. URL https://doi.org/10.1007/s41060-019-00186-0.

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

[64] Mohsin Munir, Shoaib Ahmed Siddiqui, Muhammad Ali Chattha, Andreas Dengel, and Sheraz Ahmed. FuseAD: Unsupervised Anomaly Detection in Streaming Sensors Data by Fusing Statistical and Deep Learning Models. *Sensors - Open Access Journal (sensors)*, 19(11):1–15, 5 2019.

[65] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. URL http://dl.acm.org/citation.cfm?id=3104322.3104425.

[66] Stefan Oehmcke, Oliver Zielinski, and Oliver Kramer. Event Detection in Marine Time Series Data. In *KI*, 2015.

[67] M. Ohsaki, M. Nakase, and S. Katagiri. Analysis of Subsequence Time-Series Clustering Based on Moving Average. In *2009 Ninth IEEE International Conference on Data Mining*, pages 902–907, 12 2009. doi: 10.1109/ICDM.2009.147.

[68] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv*, 9 2016.

[69] Keith Ord. Outliers in statistical data. *International Journal of Forecasting*, 12(1):175–176, 1996. URL https://EconPapers.repec.org/RePEc:eee:intfor:v:12:y:1996:i:1:p:175-176.

[70] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a time series. *Phys. Rev. Lett.*, 45:712–716, 09 1980. doi: 10.1103/PhysRevLett.45.712. URL https://link.aps.org/doi/10.1103/PhysRevLett.45.712.

[71] Georgia Papacharalampous, Hristos Tyralis, and Demetris Koutsoyiannis. Comparison of stochastic and machine learning methods for multi-step ahead forecasting of hydrological processes. *Stochastic Environmental Research and Risk Assessment*, 33(2):481–514, 02 2019. ISSN 1436-3259. doi: 10.1007/s00477-018-1638-6. URL https://doi.org/10.1007/s00477-018-1638-6.

[72] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[73] Phyks. Introducing Practical and Robust Anomaly Detection in a Time Series | Twitter Blogs, 2015.

[74] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2013.12.026. URL http://www.sciencedirect.com/science/article/pii/S016516841300515X.

[75] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Time Series Epenthesis: Clustering Time Series Streams Requires Ignoring Some Data. In *2011 IEEE 11th International Conference on Data Mining*, pages 547–556, 12 2011. doi: 10.1109/ICDM.2011.146.

[76] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

[77] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley ; Sons, Inc., New York, NY, USA, 1987. ISBN 0-471-85233-3.

[78] Zhang Rui, Zhang Shaoyan, Lan Yang, and Jiang Jianmin. Network anomaly detection using one class support vector machine. *Lecture Notes in Engineering and Computer Science*, 2168, 03 2008.

[79] Mayu Sakurada and Takehisa Yairi. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. In *Proceedings of the MLSDA 2014 2Nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, pages 4:4–4:11, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3159-3. URL http://doi.acm.org/10.1145/2689746.2689747.

[80] Catia M. Salgado, Carlos Azevedo, Hugo Proenca, and Susana M. Vieira. *Noise Versus Outliers*, pages 163–183. Springer International Publishing, Cham, 2016. ISBN 978-3-319-43742-2. doi: 10.1007/978-3-319-43742-2_14. URL https://doi.org/10.1007/978-3-319-43742-2_14.

[81] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, pages 92–101, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 978-3-642-15824-7. URL http://dl.acm.org/citation.cfm?id=1886436.1886447.

[82] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 582–588, Cambridge, MA, USA, 1999. MIT Press. URL http://dl.acm.org/citation.cfm?id=3009657.3009740.

[83] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and Statistical Modeling with Python. In *9th Python in Science Conference*, 2010.

[84] M. Shanker, M.Y. Hu, and M.S. Hung. Effect of Data Standardization on Neural Network Training. *Omega*, 24(4):385–397, 1996. ISSN 0305-0483. doi: https://doi.org/10.1016/0305-0483(96)00010-2. URL http://www.sciencedirect.com/science/article/pii/0305048396000102.

[85] Ramesh Sharda and Rajendra B. Patil. Connectionist Approach to Time Series Prediction: An Empirical Test. *Journal of Intelligent Manufacturing*, 3(5):317–323, 10 1992. ISSN 1572-8145. doi: 10.1007/BF01577272. URL https://doi.org/10.1007/BF01577272.

[86] S&P/Case-Shiller 20-City Composite Home Price Index. S&p/case-shiller 20-city composite home price index [spcs20rnsa]. https://fred.stlouisfed.org/series/SPCS20RNSA, 2020.

[87] Zaiyong Tang and Paul Fishwick. Feedforward Neural Nets as Models for Time Series Forecasting. *INFORMS Journal on Computing*, 5:374–385, 11 1993. doi: 10.1287/ijoc.5.4.374.

[88] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[89] V.N. Vapnik and A.Ya Chervonenkis. A class of algorithms for pattern recognition learning. *Avtomat. i Telemekh.*, 25:937—945, 1964.

[90] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. *CoRR*, abs/1611.06455, 2016. URL http://arxiv.org/abs/1611.06455.

[91] William Wei. *Time Series Analysis: Univariate and Multivariate Methods*, volume 33. Pearson Addison Wesley, 01 1989. ISBN 978-0-201-15911-0. doi: 10.2307/2289741.

[92] Haibing Wu and Xiaodong Gu. Max-Pooling Dropout for Regularization of Convolutional Neural Networks. *CoRR*, abs/1512.01400, 2015. URL http://arxiv.org/abs/1512.01400.

[93] Wentai Wu, Ligang He, and Weiwei Lin. Local Trend Inconsistency: A Prediction-driven Approach to Unsupervised Anomaly Detection in Multi-seasonal Time Series. *ArXiv*, abs/1908.01146, 2019.

[94] Yahoo! Webscope Dataset Ydata-Labeled-Time-Series-Anomalies. Yahoo! Webscope Dataset Ydata-Labeled-Time-Series-Anomalies-v1_0, 2010. URL http://labs.yahoo.com/Academic_Relations.

[95] Takehisa Yairi, Yoshikiyo Kato, and Koichi Hori. Fault Detection by Mining Association Rules from Housekeeping Data. In *In Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.

[96] Yufeng Yu, Yuelong Zhu, Shihua Li, and Dingsheng Wan. Time series outlier detection based on sliding window prediction. In *Mathematical Problems in Engineering*, 2014.

[97] Peter Zhang and Min Qi. Neural Network Forecasting For Seasonal And Trend Time Series. *European Journal of Operational Research*, 160:501–514, 02 2005. doi: 10.1016/j.ejor.2003.08.037.

[98] Rui Zhang, Shaoyan Zhang, Sethuraman Muthuraman, and Jianmin Jiang. One Class Support Vector Machine for Anomaly Detection in the Communication Network Performance Data. In *Proceedings of the 5th Conference on Applied Electromagnetics, Wireless and Optical Communications*, ELECTROSCIENCE'07, pages 31–37, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS). ISBN 978-960-6766-25-1. URL http://dl.acm.org/citation.cfm?id=1503549.1503556.

[99] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object Detection with Deep Learning: A Review. *CoRR*, abs/1807.05511, 2018. URL http://arxiv.org/abs/1807.05511.

[100] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks. In Feifei Li, Guoliang Li, Seung-won Hwang, Bin Yao, and Zhenjie Zhang, editors, *Web-Age Information Management*, pages 298–310, Cham, 2014. Springer International Publishing. ISBN 978-3-319-08010-9.

[101] Chong Zhou and Randy C. Paffenroth. Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 665–674, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4887-4. URL http://doi.acm.org/10.1145/3097983.3098052.

[102] Seyed Jamal Zolhavarieh, Sr Aghabozorgi, and Teh Wah. A Review of Subsequence Time Series Clustering. *The Scientific World Journal*, 06 2014. doi: 10.1155/2014/312521.

[103] M. Çelik, F. Dadaşer-Çelik, and A. Ş. Dokuz. Anomaly detection in temperature data using dbscan algorithm. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 91–95, 06 2011. doi: 10.1109/INISTA.2011.5946052.