Anomaly detection of timeseries: A comparison of statistical vs classical machine learning vs deep learning approaches

Master thesis in the department of Computer Science by Mohammad Braei
Datum: November 29, 2019, Date of submission: November 29, 2019

1. Review: Prof. Max Mühlhäuser
2. Review: Dr. Sebastian Kauschke
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

Abstract

Anomaly detection on timeseries data has been an important research field for a long time. While the original anomaly detection methods have been based on statistical approaches, in recent years more and more machine learning algorithms have been developed to detect anomalies on time series. Furthermore, many researchers tried to improve these techniques using neural networks. In the light of the continuously increasing number of anomaly detection methods of these three classes, the research community suffers from the lack of a broad comparative evaluation of statistical, machine learning and deep learning methods. This thesis tries to overcome this shortcoming by studying 20 univariate and 24 multivariate anomaly detection methods from the mentioned categories. Additionally, the evaluation will be done on publicly available datasets, which serve as benchmarks for time series anomaly detection. To provide a reliable comparison, in addition to the accuracy of each method, the computation time of the algorithms is measured. By analysing univariate as well as multivariate timeseries, we hope to provide a thorough insight about the performance of these three classes of anomaly detection approaches.

Contents

1 Introduction

Detecting anomalies has been a research topic for a long time. In a rapid digitizing world where the amount of data transfer exceeds the human ability to study it manually, the common interest relies on automated analyse of data. One of the most important data analysing 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 could imply an intrusion attack. An anomaly in a financial transaction could signify a financial fraud. Anomaly detection in medical images could help to detect diseases. Other objectives of anomaly detection are industrial damage detection, data leak prevention, identifying security vulnerabilities or military surveillance.
Anomaly detection methods are based on the type of the analysed data. For instance, the algorithms used to detect anomalies in images are different to the approaches used on data streams. In such context, this thesis focuses on methods for anomaly detection in time series data.
Anomaly detection on time series have been of interest for a long time. In 1979, Tukey [1] proposed a statistical approach to detect anomalies on time series. Chang et al. [2] proposed to use the likelihood ratio test(LRT) to detect anomalies on time series.
With the increase of computation abilities in recent decades, machine learning approaches achieved high popularity in data science tasks like classification and pattern detection. Therefore, many researchers started to use similar machine learning methods to detect anomalies in time series. For instance, they tried to use clustering methods like k-Means 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 use similar methods to detect anomalies in time series data. Therefore, different deep learning approaches like Multi -Layer Perceptron (MLP), Convolution Neural Network (CNN) and Long-Short Term Memory (LSTM) 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 these 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 yet comparing approaches of these three categories in their accuracy and performance.
In a situation like this, this thesis presents a quantitative comparison of multiple approaches of each category. It is going to 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 evaluates both univariate and multivariate anomaly detection approaches. In order to provide a reliable comparison, the methods will be evaluated on multiple time series datasets.
This thesis is structured as follows:

1. **Basics:** In this chapter, 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 chapters.

2. **Selected Anomaly detection approaches for time series:** This chapter introduces different anomaly detection algorithms of the three main categories. For each category, several approaches for univariate and multivariate time series are explained.
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 chapter lists the setup of the experiments by listing all hyperparameters of the used algorithms.
5. **Results:** Finally, this chapter illustrates the evaluation results.
6. **Conclusion:** The last chapter makes a final conclusion of the performed experiments while at the same time providing an insight of future works.

2 Basics

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.* [3]

On the other hand, there are definitions which regard outliers as a broader concept which also includes noise in addition to anomalies [4]. Others consider outliers as corruption in data while anomalies as irregular points, but having a specific pattern [5].
As we are evaluating time series in this thesis, we consider the two terms, outlier and anomaly, interchangeably.
But the important point is to deliver a formal definition for the concept of anomaly. This is essential, because different definitions of anomalies imply different 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.* [6]

But chronologically, one of the first definitions was given by Frank E. Grubbs [7]. He defined outliers in 1969:

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

And Barnet and Lewis [8] 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 D. Hawkins [9] 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 on the one hand, prevents us from using common classification approaches that rely on balanced distributed datasets and on the other hand enables us to use approaches like auto-encoders as a semi-supervised approach to detect anomalies which will be explained in the following chapters.
Thus, in this thesis 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)[4] which are not of interest to data analysts [3]. 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 2.1 and 2.2:

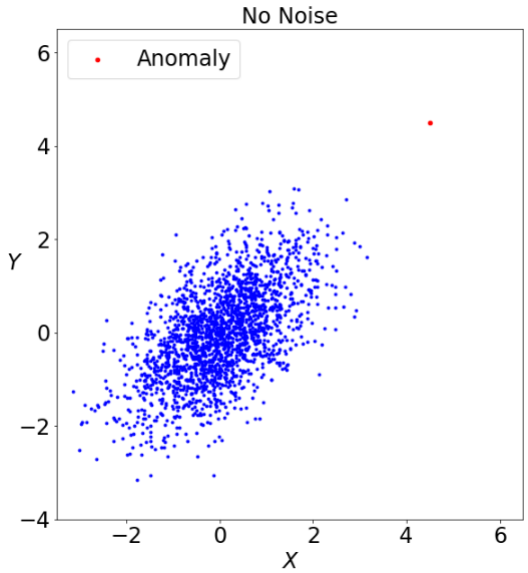

Figure 2.1: Data without Noise

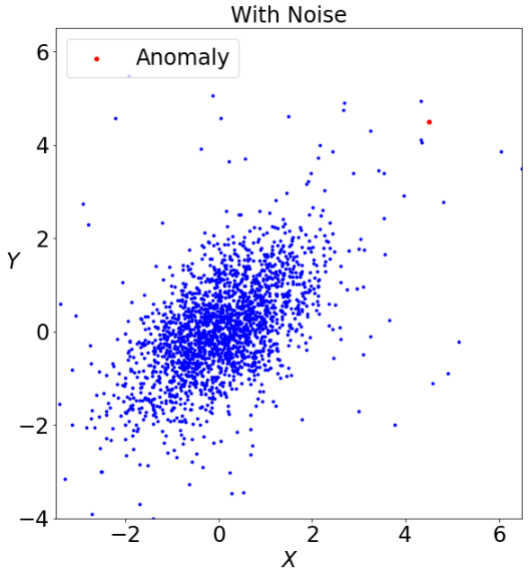

Figure 2.2: Data with noise

In both figures the main distribution is the same. In Figure 2.1 the anomalous point marked in red seems to be obvious as it deviates significantly from the rest. But in Figure 2.2 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 noises depends on the dataset. Thus, a deep understanding of the dataset is necessary to distinguish between anomalies and noises.

Anomalies should also be differentiated from novelties [6][10]. Novelty patterns are data points which haven't been observed in the data yet. The difference with 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 thesis we treat them equal.

2.2 Types of Anomalies

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

1. **Point anomalies:** If a point deviates significantly from the rest of the data, it is considered as a point anomaly. For instance, a big amount of credit transaction which differs from other transactions is a point anomaly. Hence, a point X_t is a point anomaly, if its value differs significantly from all the points in the interval $[X_{t-k}, X_{t+k}]$.
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 some context, while detected as anomaly in another context: Having an average 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.

2.3 Stochastic Process and Time series

The data that is analyzed in this thesis are time series. Thus, it is fundamental to provide a definition of it. But primarily, another term has to be defined, which regularly appears simultaneously with time series: *Stochastic Process*. William Wei [11] 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 equispaced time intervals:*

$$T = (t_0^d, t_1^d, ..., t_t^d), d \in \mathbb{N}_+, t \in \mathbb{N} \text{ where } d \text{ defines the dimension of time series}$$

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 thesis, 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 2.3 shows a time plot of a sample time series of an observed stock market value of an asset over five years.

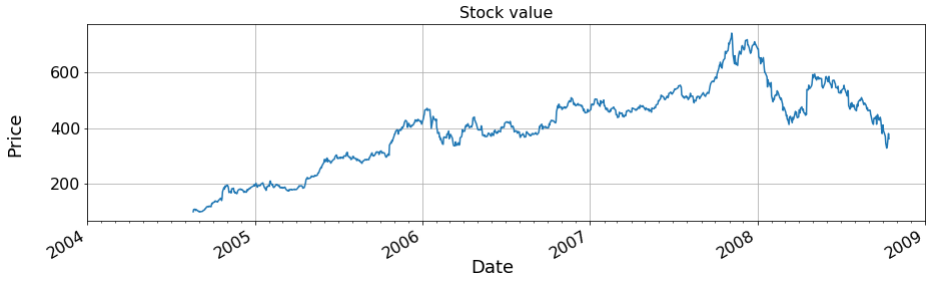

Figure 2.3: Sample time series showing the prices of a sample stock over five years

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, univariate and multivariate 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 has some important properties which will be defined briefly here. They are significant in 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 2.3 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 2.4 shows a seasonal time series. It is the monthly home sales index for 20 major US cities between the years 2000 and 2019.

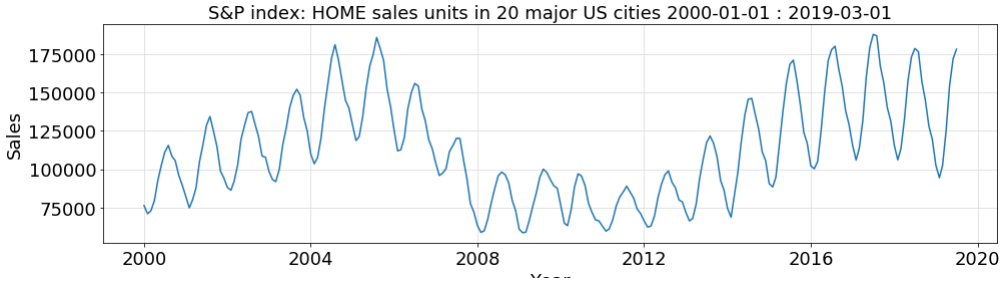

Figure 2.4: Sample time series showing the prices of a sample stock over five years

2.4.3 Cyclic

A cyclic time series is influenced by time factors where there period is not fixed and the duration is above a year, e.g., a decade. The time series in Figure 2.4 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 [12]:

Definition 2.4. *X_t is a stationary time series, if for all s, 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 different 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* [13]. 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 probabilistic looked as the likelihood of being an anomaly which is called: *anomaly score*. Thus formally, anomaly detection can be defined as a function ϕ:

$$\begin{aligned} \phi : \mathbb{R}^N &\to \mathbb{R} \\ \phi(x) &\mapsto \gamma \end{aligned} \tag{2.1}$$

where γ is the anomaly score and $x \in X \subseteq \mathbb{R}^N$ and where X is the dataset.
To convert the continuous value γ into a binary label – normal vs. anomaly – a threshold δ 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:

$$\begin{aligned} &\phi_{binary} : \mathbb{R}^N \to \{normal, anomaly\} \\ &\phi_{binary}(x) \mapsto \begin{cases} anomaly & \text{if } \phi_{score}(x) > \delta \\ normal & \text{otherwise} \end{cases} \end{aligned} \tag{2.2}$$

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 like the stock prices while multivariate contain more than one. Instances of multivariate time series are images or time series observed by several sensors.

3. Labeled or unlabeled data: A dataset is labeled if a label 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 also possible that the dataset is completely labeled but only consists of normal points. Then, it will 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 aids finding contextual anomalies.

In this thesis, we focus on univariate and multivariate 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 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 in °C of an engine recorded every 10 minutes:

$$30, 31, 33, 32, 34, 35, 35, 85, 87, 88, 89, 89$$

If these points were regarded as independent points, most methods will not identify any anomalous behavior, but detect two equal 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 [3] 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. Therefore, we will give an overview while highlighting their differences.

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 for making prediction. To make a prediction on the test data, usually a sliding window is used. A sliding window is a subsequence of a dataset, which is fed as the input to the model enabling it to predict the following timestamp. Formally, let w be the width of the sliding window, and suppose we want to predict x_i, and ψ be the learned forecasting model. To forecast x_i the following function and input data is used:

$$\hat{x}_i = \psi((x_{i-w}, ..., x_{i-1})) \tag{2.3}$$

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

$$e_i = d(x_i, \hat{x}_i) \tag{2.4}$$

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 δ, 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. 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 dwell on this issue in chapter 3.

2.5.3 Anomaly detection on multivariate time series

Multivariate Anomalies are much more complicated than univariate ones. As the dimension of the data increases, the necessity of a larger arsenal of data points extends. This makes the computation more cumbersome.
Anomaly detection based on time series prediction on multivariate time series is not as easy as in univariate time series. The rising number of random variables in multivariate time series increases the likelihood of prediction errors of the forecasting method. For univariate time series, mostly the euclidean distance is used to compute the deviation between the dependent variables. Thus, the euclidean distance between the prediction and the real value is proportional to the anomaly score in univariate time series. This is not often the case in multivariate time series. In multivariate time series two sort of deviations could result in an anomaly:

- Anomaly due to deviation of one univariate time series: There are some time series data where deviation in each of the univariate parts of the multivariate time series produces an anomaly. For example, a multivariate time series for network intrusion detection, which has a univariate time series detecting the host connection IP, another univariate time series for the port numbers that are called, and one containing the number and kind of packets sent to the server. If we have a large number of ICMPv6 packets sent to the server – whether from one machine source or multiple ones – this could reflect an anomaly like a Ping of death attack. Or if a single machine is sending lots of packages to the server, it could be a denial of service attack. Or if the univariate data shows that many different ports are called in a small time interval, then that might be a port scan attack. Hence, for such data each univariate time series could be analysed separately and if a deviation in one occurs, it could be detected as an anomaly.

- Anomaly due to deviation of the correlation of several univariates: Sometimes an anomaly occurs due to the relationship of some random variables. Figure 2.5 clarifies the idea:

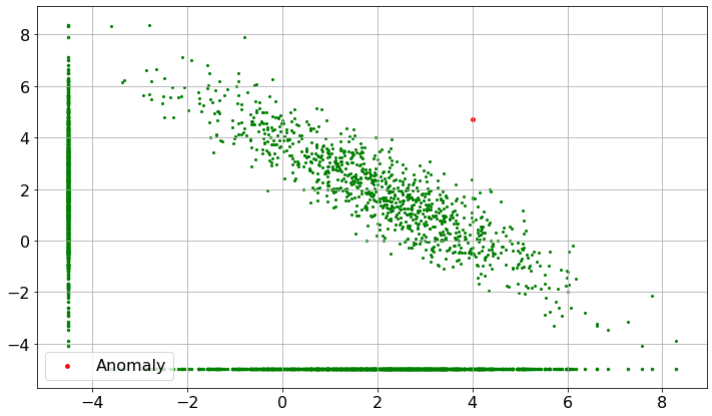

Figure 2.5: Anomaly in multivariate data

Here, you have a bi-variate dataset. We assume that the data is the content of a sliding window with width w of a multivariate time series with $x_i \in \mathbb{R}^2, \forall i \in \{1, ..., w\}$. The red point is an anomaly we want to detect. If we consider each anomaly dimension as a univariate time series, the projected points parallel to the x- and y-axis will be analysed. The abnormal point in the bi-variate time series will be projected in the most dense area on each of the univariate time series. Therefore, it will not be detected!
Figure 2.5 also shows how using euclidean distance can mislead us in detecting deviations in multivariate time series. If we imagine the centroid of the data in the center of the green points, then the euclidean distance between the anomaly to the centroid is less than many normal points that lie in the upper left or lower right of the cloud. Therefore, other similarity functions like Mahalanobis distance are proposed. We will return to this issue in chapter 3

2.5.4 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 δ 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 is existing. 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 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 δ. 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 thesis we will focus on supervised anomaly detection methods and use labeled datasets in our experiments to evaluate them.

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

Munir et al. [14] categorize outlier detection methods in probabilistic models, statistical models, linear models, proximity based models, and outlier detection in high dimensions while referencing on Aggarwal's book [3].
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 [15]. 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 chapter 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 [16]. 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 [17]. Thus, the machine learning methods rely on data modelling. Breiman [16] regards these two approaches as two different cultures recommending to use the machine learning approach. There is an ongoing debate about which of these methods performs better. In this thesis, 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 chapter, the different 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.
As there is a huge difference between these approaches when used for univariate time series on the one hand and using them for multivariate time series on the other hand, we further divide each section to explain the corresponding methods.

3.1 Anomaly detection using statistical approaches

As statistical approaches, we have selected some famous regressive models like 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 of 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. This section is divided into approaches targeting univariate time series and multivariate ones.

3.1.1 Univariate Approaches

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:

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + c + \epsilon_t \tag{3.1}$$

   The AR model in Equation 3.1 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 [18].

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 $\{\epsilon_t, \epsilon_{t-1}, ..., \epsilon_{t-q}\}$:

$$X_t = \sum_{i=1}^{q} a_i \cdot \epsilon_{t-i} + \mu + \epsilon_t \tag{3.2}$$

The MA model in Equation 3.2 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 [19].
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:

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + \epsilon_t \tag{3.3}$$

$\{X_T\}$ 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 [3].

c) **Box-Jenkins Method**[20]: The Box-Jenkins method which was introduced by George Box and Gwilym Jenkins proposes a 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 could 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:

$$X'_i = X_i - X_{i-1}, \forall i \in \{1, ..., T\} \tag{3.4}$$

The effect of differencing is best shown through plotting some data. Figure 3.1 shows the stock data from 2.3 for the years 2005 until 2008. The data shows clearly a positive trend. Therefore, we do not have stationary data here. However, Figure 3.2 shows the daily changes of the same stock over the three years and the data is stationary now:

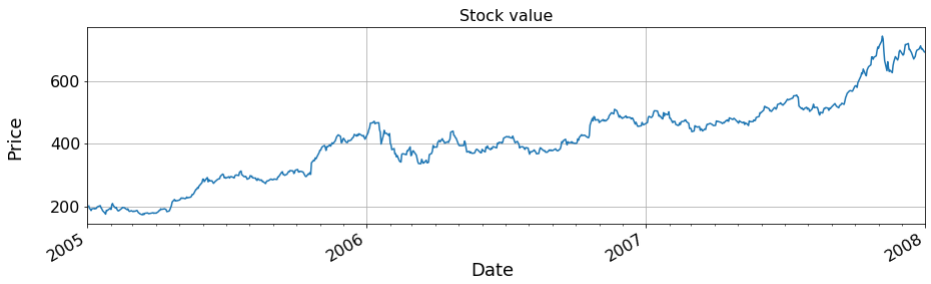

Figure 3.1: Stock value from 2005 until 2008

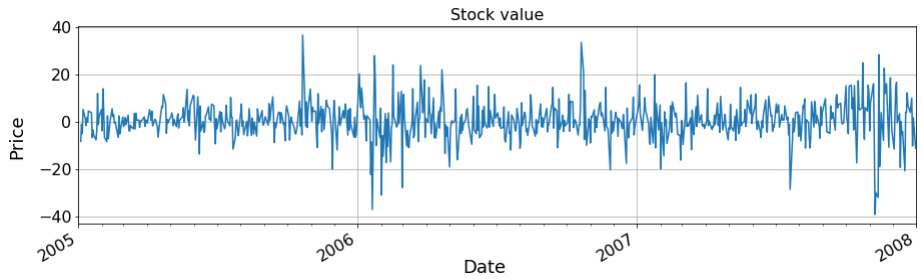

Figure 3.2: Value changes of the stock value from 2005 until 2008

Here, Figure 3.2 plots the differences between the consecutive data points of Figure 3.1.
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:

$$X'_t = X_t - X_{t-n} \text{ where n is the duration of the season} \tag{3.5}$$

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)** [21]
   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 higher weights to the latest observations:

$$\begin{aligned} X_{t+1} =& \alpha X_t + \alpha(1-\alpha)X_{t-1} + \alpha(1-\alpha)^2 X_{t-2} + ... + \alpha(1-\alpha)^N X_{t-N} \\ & \text{where } \alpha \in [0,1] \end{aligned} \tag{3.6}$$

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. This is listed in the following table:

α	0,2	0,4	0,5	0,6	0,8
X_T	0,16000	0,24000	0,25000	0,24000	0,16000
X_{T-1}	0,12800	0,14400	0,12500	0,09600	0,03200
X_{T-2}	0,10240	0,08640	0,06250	0,03840	0,00640
X_{T-3}	0,08192	0,05184	0,03125	0,01536	0,00128
X_{T-4}	0,06554	0,03110	0,01563	0,00614	0,00026
X_{T-5}	0,05243	0,01866	0,00781	0,00246	0,00005
X_{T-6}	0,04194	0,01120	0,00391	0,00098	0,00001

Figure 3.3: SAS Coefficients for X_t to X_{t-6} for different α values

6. **Double and Triple Exponential Smoothing (ES)**[22]
   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 the Triple Extension 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)** [23]
   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:

$$X_t = \frac{\sum_{j=1}^{2k} w_{t-j} X_{t-j}}{\sum_{j=1}^{2k} X_{t-j}} \tag{3.7}$$

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:

$$X_t = \frac{\sum_{j=1}^{k} w_{t-j} X_{t-j} + \sum_{j=1}^{k} w_{t+j} X_{t+j}}{\sum_{j=1}^{k} X_{t-j} + \sum_{j=1}^{k} X_{t+j}} \tag{3.8}$$

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

$$PCI = X_t \pm t_{\alpha,2k-1} \cdot s\sqrt{1 + \frac{1}{2k}} \tag{3.9}$$

Here, the factor $t_{\alpha,2k-1}$ is the p-th percentile of a Students 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 ETS methods. However, it was included in the evaluation of this thesis, because it uses a k-Nearest Neighbor (k-NN) attempt and is a newer approach that was published in 2014.

3.1.2 Multivariate Approaches

1. **Average of anomaly scores of multiple univariate time series**[18]
   One of the simplest approaches used for anomaly detection on multivariate time series is used to compute the anomaly score on each individual univariate time series and take the average of this score as the total anomaly score of the whole multivariate time series. It is also possible to take the maximum and minimum anomaly score. We will evaluate all of them.
   To compute the anomaly score for each univariate time series, some of the univariate anomaly detection methods explained so far can be used. For instance, Varun Chandola [18] uses the k-NN method and Windows based approach with One-Class SVM to assign an anomaly score to each individual univariate time series. In this thesis, we will evaluate Autoregressive Model (AR), Moving Average(MA), Extreme Gradient boosting (XGBoost) and One-Class Support Vector Machine (OC-SVM). XGBoost and OC-SVM will be explained in Section 3.2.1.
   One of the most important shortcomings of this approach is that it ignores the relationship between the random variables. We explained this issue in section 2.5. Often an anomaly in multivariate time series affects all components in such a way that they are not detectable by univariate anomaly detection methods because univariate methods fail to combine information about the outlier among the component series [24].
   Despite this shortcoming, we will evaluate this anomaly detection method to have a persistent model to compare other approaches with.

2. **Anomaly detection by projection**

One of the main approaches to detect anomalies in multivariate time series is to project a multivariate time series to univariate time series and then detect the anomaly by evaluating the deviation of a point or set of points.

Chandola [18] proposes a window based technique reducing a multivariate time series into a univariate time series by exploring the change in the correlation structure of the time series using subspace monitoring. Cheng et al. [25] uses a kernel matrix which contains the similarity information between the multiple time series. Peña and Prieto [5] project the data onto the direction that maximizes or minimizes the kurtosis coefficients where then they identify the outliers . Gupta et al. [26] extended this approach by adding the directions of minimal and maximal kurtosis coefficients in addition to orthogonal directions. Then, they use an iterative approach to clean the model of outliers where they can finally estimate the outlier effects and model parameters . In our evaluation, we will take Chandola's approach based on correlation of the subspaces of the multivariate time series: *Subspace Monitoring for Multivariate Time Series*

A multivariate time series can be converted into a univariate one, if the univariate time series consists of temporal points containing the difference between two successive multivariate time series windows. The original concept is based on the method proposed by Jordan et al. [27].

Let $X_t \in \mathbb{R}^{T \times d}$ be a multivariate time series consisting of d univariate time series with length T. Also let w be the length of a window, so that $W_t = x_t, x_{t-1}, ..., x_{t-w}$. Then, W_t and W_{t-1} are two successive windows. Consider, V_t and V_{t-1} as the subspace of the principle components of W_t and W_{t-1} capturing $\alpha\%$ of the total variance. Then, the change $\delta_{t-1,t}$ is the maximum change between the two spaces V_t and V_{t-1}. This can be computed by solving the eigenvalue problem of the following matrix:

$$V_t^T V_{t-1} V_{t-1}^T V_t \tag{3.10}$$

The eigenvalues of the Equation 3.10 are as follows:

$$\lambda_1 \geq \lambda_2 \geq :. \geq \lambda_{p+q} \tag{3.11}$$

where p and q are the number of basis vectors of V_t and V_{t-1} respectively. It can be shown that the following equations apply:

$$cos\theta_1 = \lambda_1, ...cos\theta_{p+q} = \lambda_{p+q} \tag{3.12}$$

According to Manabu Kanu et al. [28], the following statement is true:

The change of subspace, when an m-dimensional subspace F1 changes to F2, is defined as the maximum distance between an arbitrary unit vector x in F2 and the subspace F1.

Hence, the change between the two subspace V_t and V_{t-1} can be expressed as:

$$\sigma_{t-1,t} = \sqrt{1 - \lambda_{min}} \tag{3.13}$$

The proof of Equation 3.10 can be seen in [18]. Consequently, the multivariate time series $\{X_T\}$ is converted into a univariate time series $\{\sigma_{T-w+1}\}$. After that, a univariate anomaly detection method like One-class SVM can be used to detect the anomalies. In this thesis, we will use the AR-Model, MA-Model, OC-SVM and XGBoost as univariate approaches to evaluate this algorithm. OC-SVM and XGBoost will be explained in Section 3.2.

One advantage of this approach is that it is not necessary to first fit the VARMA model.

There are also approaches which can be used to estimate the similarities between two spaces. Gupta et al. [25] computes the average cosine of the angle θ between the basis vectors of the two spaces V_t and V_{t-1}:

$$sim(V_t, V_{t-1}) = \frac{V_t^T V_{t-1} V_{t-1}^T V_t}{k} \tag{3.14}$$

where k is the number of principle components in the two spaces. In their approach, the two subspaces have an equal number of principle components. The Equation 3.14 can be rewritten as:

$$sim(V_t, V_{t-1}) = \frac{1}{k} \sum_{j=1}^{k} \sum_{i=1}^{k} cos^2 \theta_{ij} \tag{3.15}$$

which can be rewritten as

$$sim(V_t, V_{t-1}) = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} \lambda_{V_{t,i}} \lambda_{V_{t-1,j}} cos^2 \theta_{ij}}{\sum_{i=1}^{k} \lambda_{V_{t,i}} \lambda_{V t-1,j}} \tag{3.16}$$

where $\lambda_{V_{t,i}}$ is the eigenvalue of the i^{th} principle component of V_t.
In this thesis, the proposed method of Chandola will be used.

3. **Time series Outlier Detection using Prediction Confidence Interval (PCI)** [23]
Yu et al. recommend to use the same univariate approach for multivariate time series. Therefore, each univariate time series of the multivariate time series is analyzed separately using the PCI method explained before. Thus having a d-dimensional time series, d are each used analyzing a univariate time series. This results in d anomaly scores for each timestamp $x_i \in \mathbb{R}^d$. The mean of this anomaly scores is the anomaly score of the multivariate timestamp.

4. **Vector based Autoregressive Model (VAR)**
Vector based Autoregressive model is a AR-Model for multivariate time series. Multivariate time series of a VAR model are based on the assumption that each timestamp is dependent on the previous timestamps and the values of other variables. Thus let $X_t = (x_{1,t}, x_{2,t}, ..., x_{d,t})$ be a timestamp of a multivariate time series such that $X_t \in \mathbb{R}^d$. Hence, X_t is a $(d \times 1)$ vector. Then, a p-th order VAR model X_t is equal to:

$$X_t = c + \Theta_1 X_{t-1} + \Theta_2 X_{t-2} + ... + \Theta_p X_{t-p} + \epsilon_t \tag{3.17}$$

where Θ_i is a $d \times d$ matrix. Like the AR-Model for univariate time series, ϵ_t is white noise.
To detect anomalies on a multivariate time series, first a p-th order VAR model is fitted on the given time series. After that, the deviation ϵ is used to determine if a timestamp X_t is an anomaly or not. Leigh et al. [29] use the euclidean distance to compute the amount of deviation between the observed timestamp X_t and the predicted one $\hat{X}_t$. As the euclidean distance is not always the best option for all kind of data, we also use the Mahalanobis distance to include the relationship between the variables of the error values.

5. **Vector Autoregression Moving-Average (VARMA)**
VARMA is a multivariate ARMA process. Equal to the univariate ARMA process, a VARMA process is defined by the p-th order vector AR-Model(VAR) and a q-th order vector MA-Model(VMA). Thus, let X_t be a (p, q)-VARMA model, then X_t is rewritten as:

$$\begin{aligned} X_t &= \Theta_1 X_{t-1} + \Theta_2 X_{t-2} + ... + \Theta_p X_{t-p} + \Phi_1 \epsilon_{t-1} + \Phi_2 \epsilon_{t-2} + ... + \Phi_q \epsilon_{t-q} + \epsilon_t \\ &= \sum_{l=1}^{p} \Theta_l X_{t-l} + \sum_{m=1}^{q} \Phi_m \epsilon_{t-m} + \epsilon_t \end{aligned} \tag{3.18}$$

After fitting the VARMA-Model, similar to the VAR approach, we will use the error value ϵ_t to assign an anomaly score to each timestamp. Also here, we will evaluate the euclidean and mahalanobis distance.

3.2 Anomaly detection using classical machine learning approaches

Machine learning algorithms try to detect anomalies in the time series dataset 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 [16]. Many researchers argue that a theoretical foundation of a model can be neglected, if the method performs effectively in practise [30]. In such context, in this section several univariate and multivariate anomaly detection methods, using classical machine learning algorithms, are introduced. Later, the performance of these algorithms are compared to the statistical approaches introduced so far and the deep learning approaches in Section 3.3.

3.2.1 Univariate Approaches

1. **K-Means Clustering – Subsequence Time-Series Clustering (STSC)**
   One of the clustering algorithms for anomaly detection is using K-Means clustering [31]. This method is also called *Subsequence time-series Clustering (STSC)* [32]. To use K-Means as an anomaly detection method for time series data, sliding windows approach is used [33] [34]. This implies that given a time series $\{X_T\} = (x_1, x_2, ..., x_T)$ and window length w and a slide length γ, the time series $\{X_T\}$ results in a set of sub sequences $\mathcal{S}$:

   $$\mathcal{S} = \{(x_0, x_1, ..., x_w), (x_{0+\gamma}, x_{1+\gamma}, ..., x_{w+\gamma}), ..., (x_{T-w}, x_{T-w+1}, ..., x_T)\} \tag{3.19}$$

   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 [35]. 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 $\mathcal{E}$:

   $$\mathcal{E} = (e_0, e_1, ..., e_{|\mathcal{S}|}) \tag{3.20}$$

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

   $$e_i = \min_{\forall c \in \mathcal{C}}(d(s_i - c)) \tag{3.21}$$

   where d is 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 δ 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 [36].
   Note: Lin et al. [36] 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 run of 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 [36]. 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 approved 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 centroid are sinus waves, which are totally different to the instances in the CBF dataset. Several authors tried to analyse this behavior mathematically [32, 37, 38] 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 [39, 40]. But the problems remain generally unsolved[41].
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) [42]. But in comparison to other clustering methods like STSC or CBLOF[43], 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. *BoarderPoint*(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 Core-Points 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. [44] have used DBSCAN for anomaly detection on 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, the 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 prominent clustering algorithm is Local Outlier Factor (LOF). In contrast to DBSCAN, it is not based on density but finding the nearest neighbors (K-NN)[45] while also focusing on local outliers. LOF was initially designed to detect anomalies on spatial data [46]. But Breuning et al.[47] extended the approach to use it also for time series data.
Let D be a dataset and $x \in \mathcal{D}$. To calculate the LOF value of a data point x, the following steps has 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:

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

Breuning et al. 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:

$$A(w, t) \subseteq P(\{X_T\}) = \{(x_i, ..., x_{i+w}) | i \in \{1, ..., t - w\}, t \leq |\{X_T\}|\} \tag{3.22}$$

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

$$B(w, t) \subseteq \{X_T\} = \{x_{t-\frac{w}{2}}, ..., x_{t+\frac{w}{2}}\} \tag{3.23}$$

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 δ 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 points:

- 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 ensemble strategy to compute k[48].
- Concatenating the time series into a vector and computing the distance to other vectors removes the ordered information of a time series. Here a temporal data is converted into a 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 Bruening et al. 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 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 that the complexity is $O(n^2)$ as compared to DBSCAN where the complexity is $O(nlogn)$.

4. **Isolation Forest**
   One of the machine learning approaches to detect anomalies in time series is isolation forest using sliding window. Isolation forest, also known as iForest, was introduced by Lui et al. [49]. 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 [50]. Figure 3.4 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:

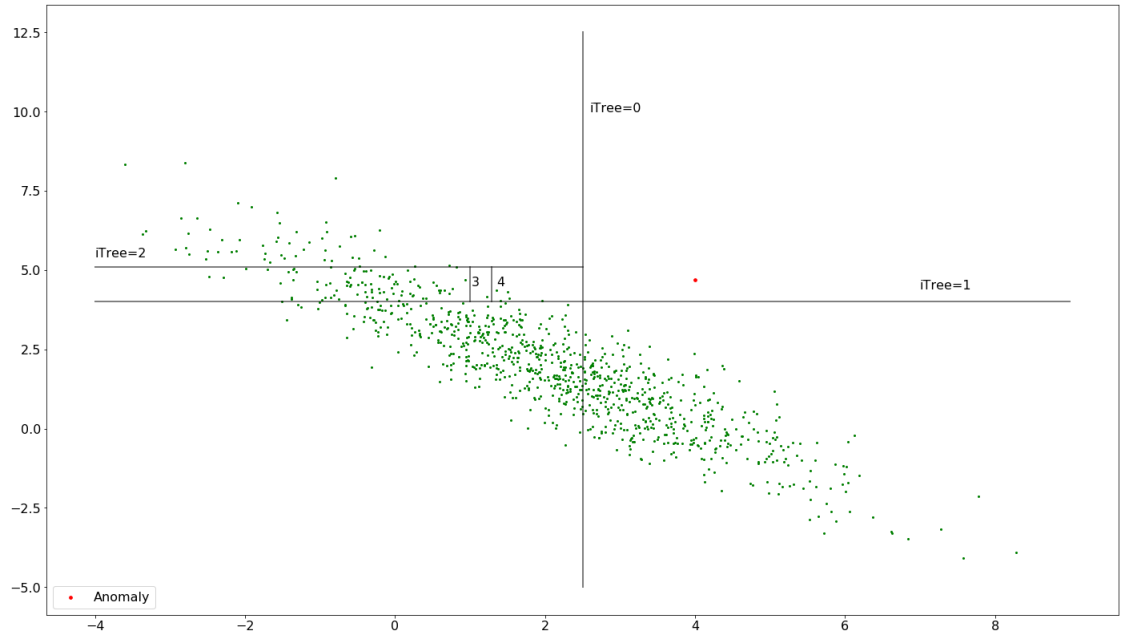

Figure 3.4: Isolation Forest: The anomaly is isolated by the random generated tree after two partitions

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 and w the window length. Then:

$$\begin{aligned} W :=& (W_1, W_2, ..., W_p) \\ =& ((x_1, ..., x_w), (x_2, ..., x_{w+1}), ..., (x_p, ..., x_{w+p-1})) \end{aligned} \tag{3.24}$$

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 test set later.
Ding et al. [51] 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:

$$\begin{aligned} S(x, w) &= 2^{-\frac{E(h(x))}{c(w)}} \\ E(h(x)) &= \frac{1}{L} \sum_{i=1}^{L} h_i(x) \end{aligned} \tag{3.25}$$

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 et al. [51] have shown in their experiment results that fixed sliding window 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 the higher number of iTrees will increase the computation time [50]. For w as the window length and L 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 like 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 Vladimir N. Vapnik and Alexey Ya. Chervonenkis 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)[52].

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 et al. [53] propose to unfold the time series into a phase space using a time-delay embedding process [54]. Zhang et al. recommended to create windows with length w of the time series dataset, so that for a given time series $\{X_T\} = (x_1, x_2, ..., x_T)$, the dataset is first converted into [55][56]:

$$\begin{aligned} W :=& (W_1, W_2, ..., W_p) \\ =& ((x_1, ..., x_w), (x_2, ..., x_{w+1}), ..., (x_p, ..., x_{w+p-1})) \end{aligned} \tag{3.26}$$

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

$$\begin{aligned} & f : \mathbb{R} \to \mathbb{R}^2 \\ & f(X_t) \mapsto \begin{cases} [X_t X_t] & \text{for t=1} \\ [X_t X_{t-1}] & \text{else} \end{cases} \end{aligned} \tag{3.27}$$

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 techniques which also gained high performance in the Kaggle and KDDCup competitions is Extreme Gradient boosting (XGBoost). One of the main advantages of XGBoost is its scalabality [57]. XGBoost is derived from the Tree boosting algorithm using the second order method introduced by Friedman et al. [58].
Let D be a dataset of n example with m dimension: $D = \{(x_i, y_i) | x_i \in \mathbb{R}^m, y \in \mathbb{R}, i \in \{1, ..., n\}\}$. Then, tree boosting uses a sequential sequence of tree models to make a prediction $\hat{y}$ for x_i:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \text{ where } f_k \in \mathbb{F} \tag{3.28}$$

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

$$\begin{aligned} & \mathcal{L}(\phi) = \textstyle\sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \\ & \text{where } \Omega(f) = \gamma T + \tfrac{1}{2}\lambda ||w||^2 \end{aligned} \tag{3.29}$$

where T is the number of leaves in each tree and w the leaf weights.
The loss function in Equation 3.29 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.

$$\mathcal{L}^{(t)} \approx \sum_i [l(\hat{y_i}^{t-1}, y_i) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{3.30}$$

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 [57].

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 δ to detect anomalies.

3.2.2 Multivariate Approaches

1. **Local Outlier Factor (LOF)**
   Anomaly detection using LOF on univariate time series was explained in Section 3.2.1. Oehmcke et al.[47] used the same method for multivariate time series. Their experiments were done on marine time series data where 16 sensor values were observed.
   Thus, to compute the anomaly score, each element $x_i \in \{X_T\}$ is a multidimensional vector:

   $$x_i \in \mathbb{R}^d \text{ where } d > 1 \tag{3.31}$$

   Hence, $B(w,t)$ is redefined as following:

   $$B(w,t) = (x_1^{t-\frac{w}{2}}, ..., x_d^{t-\frac{w}{2}}, ..., x_1^{t+\frac{w}{2}}, ..., x_d^{t+\frac{w}{2}})^T \tag{3.32}$$

   In this context, the anomaly score is computed as described in Section 3.2.1. Oehmcke et al. have used the Euclidean distance for their experiments.
   The critical points of this method is the use of Euclidean distance. Using the Euclidean distance as a similarity function for outlier detection could ignore the relationship between variables in multidimensional space. Therefore, the performance of the method is highly dependent of the distribution of the specific dataset.

2. **Density-Based SpatialClustering of Applications with Noise (DBSCAN)**
   DBSCAN has been used to detect anomalies in multivariate time series. Elsner et al. [59] use DBSCAN to detect anomalies in Enterprise Applications containing multivariate time series. The recommended algorithm consists of the following steps:

   a) Select a subset of the time series, i.e., for a sliding window with width w, select $X_w := (x_i, ..., x_{i+w}) \in \{X_T\}$.

   b) On each timestamp $x_i \in \{X_w\}$ where $x_i \in \mathbb{R}^N$ train a multivariate DBSCAN model. This results in several clusters in n-dimensional space.

   c) Select the biggest cluster: The cluster with the most points. This reveals the centroid of the cluster: C_{normal}.

   d) Calculate the distance of each observation $x_i \in X_w$ to the centroid C_{normal}. This distance reflects the anomaly score τ.

   Elsener et al. recommend to use $\epsilon = 1$ where ϵ is the distance and $\mu = 10$ where μ is the minimum number of points each normal cluster has to have.
   Finally, we have to declare a threshold δ, so that:

   $$x_i == Anomaly, \forall x_i \in X_w \Leftrightarrow \tau_{x_i} > \delta \tag{3.33}$$

   This is done by using the training part of the time series dataset.

3. **Isolation Forest**
Isolation forest can also be used to detect anomalies on multivariate time series. Diang et al. [51] also used Isolation forest for anomaly detection on multivariate time series. The concept is the same as used for univariate time series. Let $\{X_T\}$ be the multivariate time series such $x_i \in \{X_T\}$ and $x_i \in \mathbb{R}^N$. A sliding window with width w is defined, such that:

$$\{X_W\} := (W_1, W_2, ..., W_p) = ((x_1, ..., x_w), ..., (x_p, .., x_T)) \tag{3.34}$$

An element of $x_j \in \{X_W\}$ is the concatenation of all dimensions N of the points in the sliding window:

$$x_j = (x_{1,1}, ..., x_{1,N}, ..., x_{w,1}, ..., x_{w,N}) \tag{3.35}$$

This will be the input of the isolation forest as described in the univariate part. The isolation forest will detect the anomaly score τ of x_j:

$$\tau_{x_j} = 1 \frac{1}{w} \sum_{i}^{w} \sum_{j}^{N} h(x_{i,j}) \tag{3.36}$$

where h(x) denotes the length of the i-th Tree.
Again a threshold δ can be computed using the training part of the time series dataset.

4. **One-Class Support Vector Machine (OC-SVM)**
Lamrini et al. [60] have used OC-SVM for anomaly detection on multivariate time series. Thus, for $\{X_T\}$ to be a multivariate time series, they use a sliding window with width w such that the time series data $\{X_T\}$ is split in:

$$\{X_W\} := (W_1, W_2, ..., W_p) = ((x_1, ..., x_w), ..., (x_p, .., x_T)) \tag{3.37}$$

Thus, an element $x_j \in \{X_W\}$ consists of w timestamps where each timestamp has N dimenions. This data is further processed: Each dimension of each sequence x_j is characterized by seven feautures:

a) Minimum of the sequence

b) Maximum of the sequence

c) Mean of the sequence

d) Median of the sequence

e) Standard deviation of the sequence

f) Squared error computed between the sequence and the linear fitting

g) Number of the average crossings

The features are normalized and used as input for the OC-SVM. This approach converts a multivariate time series to a univariate one which is then used as input for OC-SVM. The rest of the algorithm is equal to the approach described in the univariate section.

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 researches have tried to evaluate the performance of neural networks compared to the classical approaches like ARIMA. Sharda et al. [61] compared 101 time series using forward neural networks and the ARIMA model. Thang et al. [62] also compared neural networks with ARIMA models focusing on 16 time series with different complexities. Using neural network for time series forecasting paved the way for using neural network to detect anomalies in univariate and multivariate time series. In this section, we selected the most prominent approaches used in recent years.

3.3.1 Univariate Approaches

1. **Multiple Layer Perceptron (MLP)**
   The most fundamental artificial neural network architecture (ANN) is the Multilayer Perceptron (MLP) [63] which is a fully-connected feed-forward neural network. According to Hyndman et al. [12] 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:
   $$\text{NNAR}(p, 0) \Leftrightarrow ARIMA(p, 0, 0) \tag{3.38}$$
   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 3.5):

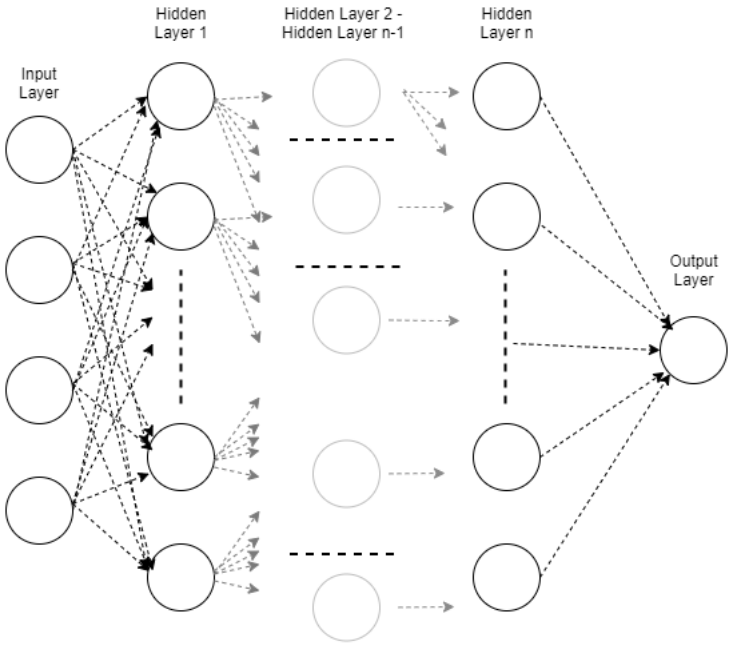

Figure 3.5: Multilayer Perceptron (MLP)

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 [64]. Haselsteiner et al. [65] 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 [66].
In this thesis, 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 $\{X_T\}$ be a time series, $x_i \in \{X_T\}$, w the window length, and f the function of the MLP, then:

$$\hat{x}_{t+1} = f(x_{t-w}, ..., x_t), \forall t \in \{w, ..., T\} \tag{3.39}$$

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:

$$(\hat{x}_{t+1}, ..., \hat{x}_{t+p_w}) = f(x_{t-w}, ..., x_t) \tag{3.40}$$

p_w is the number of timestamps the MLP predicts which is called the *prediction window* or *Forecasting Horizon*[14].
The prediction of the MLP is then used to detect anomalies. Let δ be the anomaly threshold, then x_{i+1} is marked as an anomaly, if:

$$f(x_{i-w}, ..., x_i) - x_{i+1} > \delta \tag{3.41}$$

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. 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 [67].

2. **Convolution Neural Networks (CNN)**
Another artificial neural network approach, which is used for anomaly detection in time series data is deep convolution neural networks (CNN). CNNs are mainly used in computer vision for tasks like object detection, classification and segmentation [68, 69, 70]. In contrast to MLP, where layers are fully connected, a CNN uses convolution layers that are partially connected reducing the amount of parameters enabling them to go deeper and train faster. CNN, in contrast to MLP, focus on local patterns in the data. In addition to the convolution layers, CNNs also use pooling layers as regularizer to avoid overfitting. One of the pooling operations, which achieved the best results [71] is the maximum pooling [72].
In the recent years, there have been increasing interests in using CNNs for time series analysis. Munir et al.[14] used 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.[73] 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 a bit different than 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 3.6 shows the architecture used in DeepAnT:

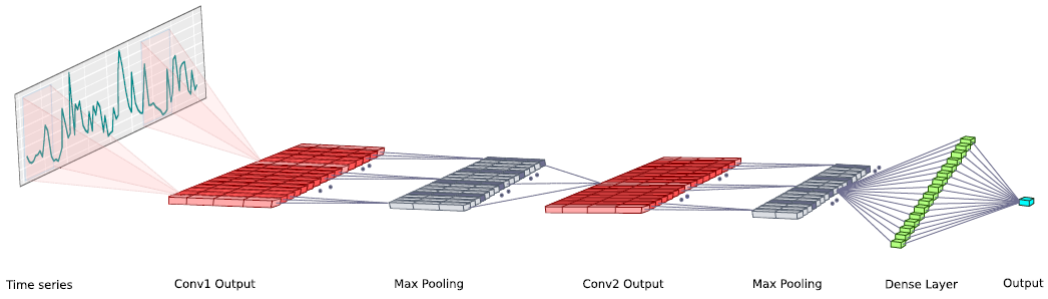

Figure 3.6: DeepAnT architecture for time series prediction [14]

The first layer after the input layer is 1-dimensional convolution layer followed by a max-pooling layer. As figure 3.6 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 CNN in particular is the architecture of the model. The amount of convolution and max-pooling layers differ in the architectures used in this thesis based on the dataset. We will analyse

this in chapter 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, the rectified linear units (ReLU) is used [74]. Sergey Ioffe et al. suggested another regularization technique called Batch Normalization [75]. 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 δ be the anomaly threshold, f the function implemented by the CNN model, then x_{i+1} is marked as an anomaly, if:

$$f(x_{i-w}, ..., x_i) - x_{i+1} > \delta \tag{3.42}$$

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 was Residual Neural Network (ResNet). ResNet introduced a new artifact called residual blocks developed by He et al. [76]. Residual blocks add the output of a convolution block with the input of it using a skip connector. A convolution block consists of several convolution layers, activation layers and regularization artifacts like 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:

$$y = \phi(x_i) + x_i \tag{3.43}$$

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

$$y = \psi(\phi(x_i) + x_i) \tag{3.44}$$

Figure 3.7 shows a residual block that is used in this thesis:

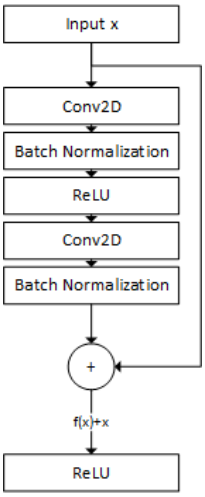

Figure 3.7: Residual Block used for time series consisting of convolution block of two convolution layers with one ReLU and two Batch Normalization layers

Residual Blocks were used to avoid the vanishing gradient problem, which occurs often in deeper CNNs.
Wang et al.[77] used 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 3.7, 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 chapter 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 Aäron van den Oord et al. [78] 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, which was constructed to create audio waveforms, is a probabilistic model that tries to approximate the joint probability of a waveform $x = \{x_1, x_2, ..., x_T\}$:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}) \tag{3.45}$$

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 3.8 shows a CNN with regular convolution layer:

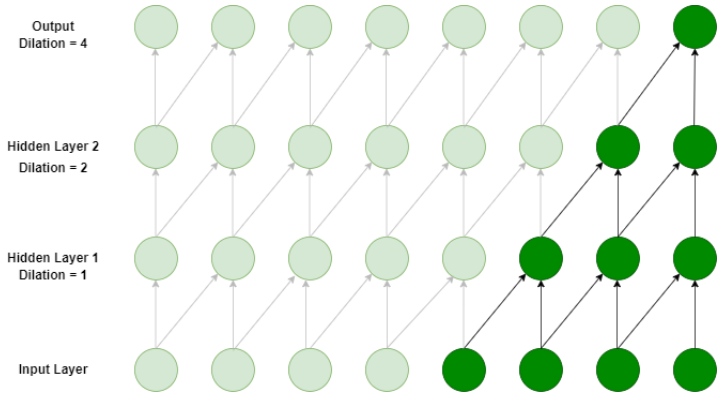

Figure 3.8: CNN with regular convolution layers with filter size of 2

In contrast, the dilated convolution layer performs the convolution operation on a data bigger than the filter size. This is accomplished by skipping some input values using a skip step. Figure 3.9 shows a CNN with delation convolution layers:

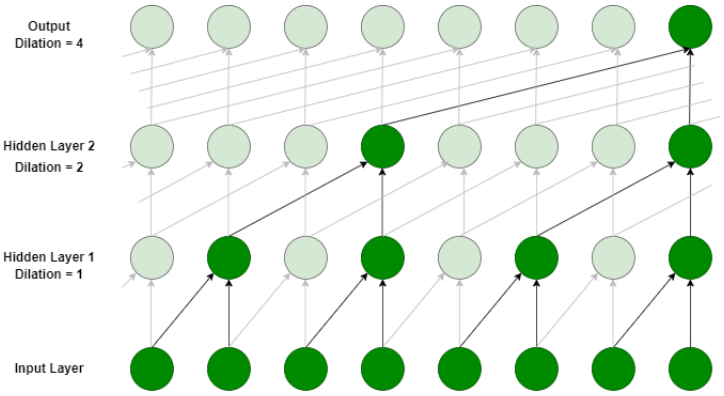

Figure 3.9: CNN with dilated convolution layers with filter size of 2 and changing skip steps (delation) = $\{1, 2, 4\}$

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. [79] used the concept of WaveNet to design a CNN for time series forecasting. Thus, to predict a timestamp $x_t \in \{X_T\}$, a sequence of timestamps with width w is used as the input for the function f which is expressed by the CNN:

$$\hat{x}_t = f((x_{t-w}, ...x_{t-1})) \tag{3.46}$$

The dilation of the convolution layers increases by a factor of 2, which is illustrated in Figure 3.9. Therefore, the window width w can be much bigger than the value used in normal convolution layers. In this thesis, we will extend this approach to also detect anomalies where we use the same approach we used for the MLP and CNN approach:
Let δ be the anomaly threshold, then x_t is marked as an anomaly, if:

$$\hat{x}_t - x_t > \delta \tag{3.47}$$

.

5. **Long Short Term Memory (LSTM) network**
Another ANN, which is designed for sequence data is the LSTM network. LSTM network belongs 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:

$$y_t = \phi(x_t^T \cdot w_x + b) \tag{3.48}$$

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

$$y_t = \phi(x_t^T \cdot w_x + y_{t-1}^T \cdot w_y + b) \tag{3.49}$$

A neuron in a simple RNN is computed like 3.49. Sepp Hochreiter et al. [80] developed a new version of recurrent cells called Long Short Term Memory (LSTM). Figure 3.10 shows the structure of an LSTM cell.

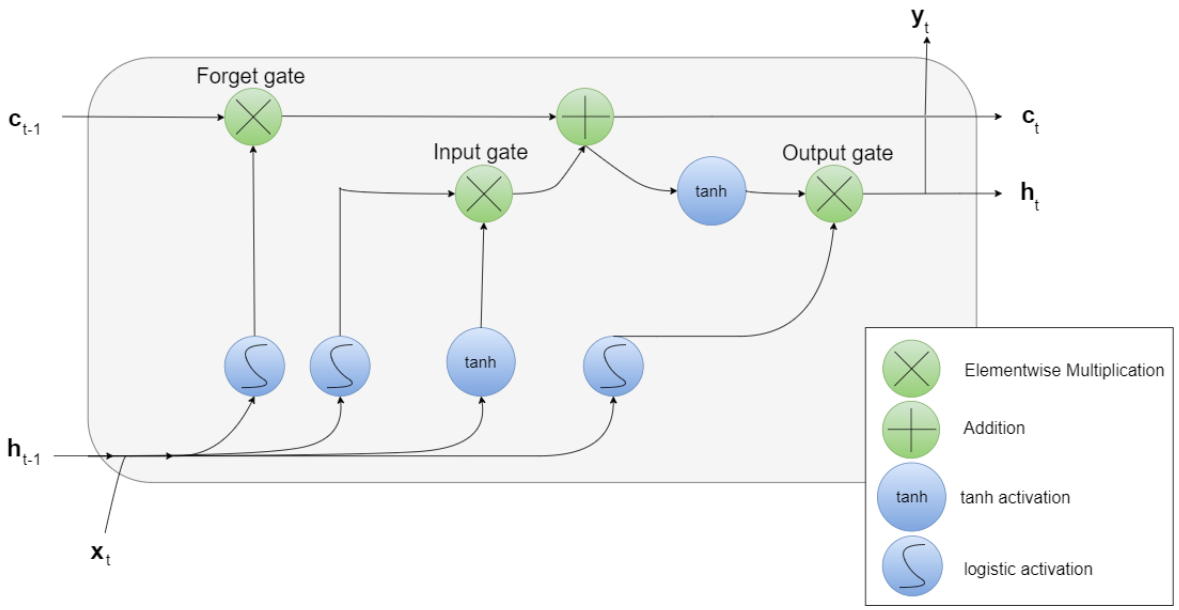

Figure 3.10: LSTM Cell

In contrast to 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 long term memory. The three gates regulate how much of 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 elementwise 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 architecutures 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 timestamps, which accelerates the learning process.
Chauhan et al. [81] use an 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.

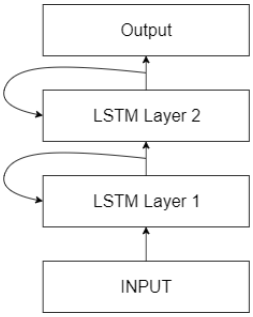

Figure 3.11: Stacked LSTM: The original LSTM model consisted of just one LSTM layer. Stacked LSTM has multiple LSTM layer

Malhotra et al. [82] use a stacked LSTM (Figure 3.11) 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:

$$(\hat{x}_{i+1}, ..., \hat{x}_{i+l}) = f(x_i) \tag{3.50}$$

$$\forall i \in \{1, ..., t\}, l < i < t - l, \text{ each } x_i \text{ is predicted l-times} \tag{3.51}$$

Then, the error value of each prediction is computed:

$$e^{(i)} = (e_1, ..., e_l) = (f(x_{i-l})_l - x_{i+1}, f(x_{i-l+1})_{l-1} - x_{i+1}, ..., f(x_i)_1 - x_{i+1}) \tag{3.52}$$

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^t a likelihood p^i. Finally given a anomaly threshold δ:

$$x_i \text{ is an anomaly} \longrightarrow P(e^{(i)})_{e^{(i)} \sim \mathcal{N}(\mu, \Sigma)} > \delta \tag{3.53}$$

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

6. **Gated recurrent unit (GRU)**
In 2014, Cho et al. [83] 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:

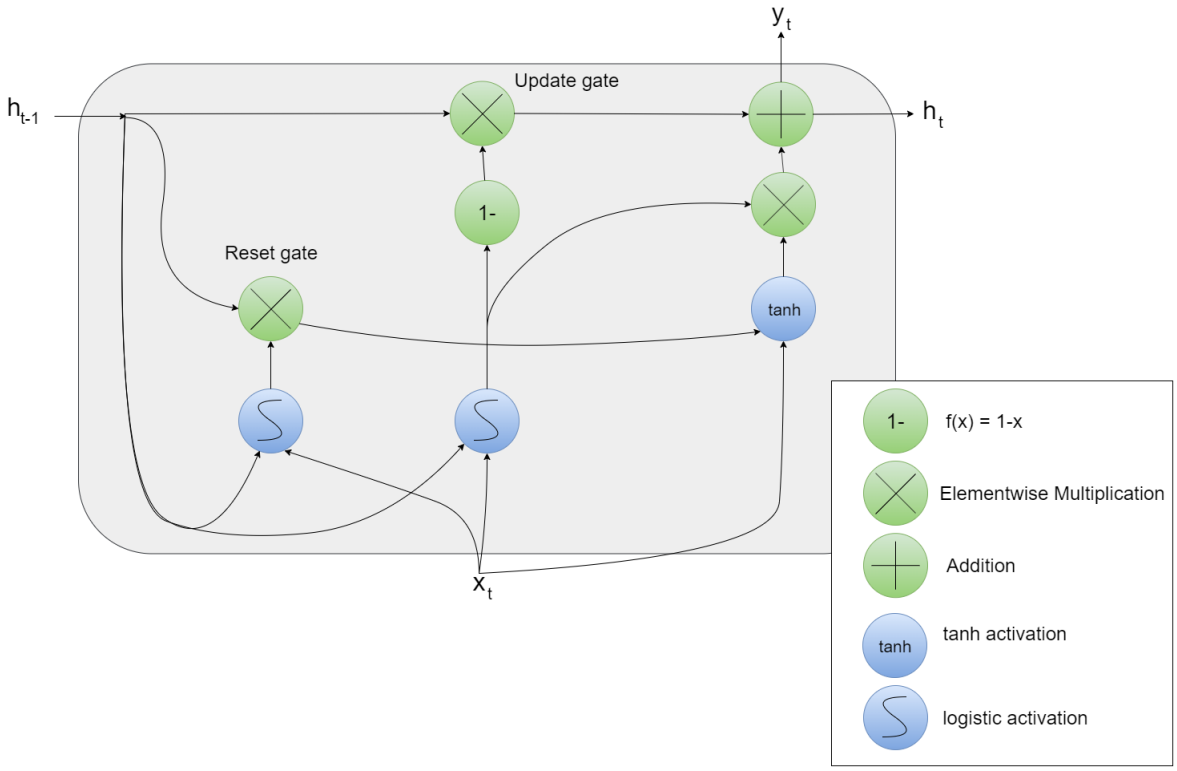

Figure 3.12: Gated recurrent unit (GRU)

Researches have shown that mostly GRU performs as well as LSTM although requiring less computation due to its simplified structure [84].
GRU has also been used for anomaly detection on time series data. Wu et al. [85] 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 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 original space will show significant difference in some data points, which represent the anomalous data instances making the autoencoder appropriate for anomaly detection.
Autoencoders belong to the feed-forward neural networks, which is 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:

$$\hat{X} = \psi(\phi(X)) \tag{3.54}$$

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

$$\min_{\theta_\psi,\theta_\phi} \left\| X - \hat{X} \right\|_2 = \min_{\theta_\psi,\theta_\phi} \|X - \psi(\phi(X))\|_2 \tag{3.55}$$

where θ_ϕ and θ_ψ are the weights of the decoding and encoding part. Figure 3.13 shows the concept of an autoencoder graphically:

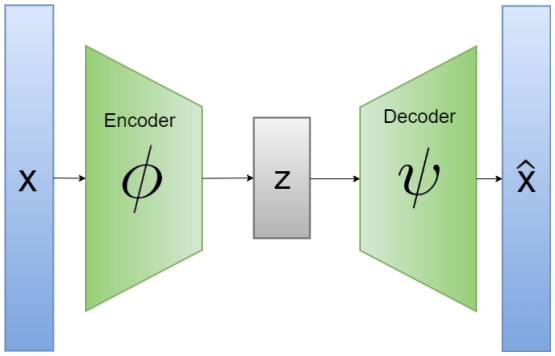

Figure 3.13: Autoencoder: The encoding layers ϕ reduce the dimension of x to z and the decoding layers ψ projects the data to original dimension resulting in $\hat{x}$

There have been different approaches using autoencoder to detect anomalies on spatial data. Chong Zhou et al. [86] use Robust autoencoders to detect anomalies on images. Baur et al. [87] use deep autoencoders to detect anomalies on 2D brain MR images.
Mayu Sakurada et al. [88] use an autoencoder to detect anomalies on time series and compare it with linear and kernel PCA. They implemented 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 thesis, we will implement the normal autoencoder to detect anomalies on time series. Thus, let $\{X_T\}$ 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 $(x_i, x_{i+1}, ..., x_{i+w}) \in \{X_T\}$. Then the autoencoder will compute the following:

$$(\hat{x}_i, \hat{x}_{i+1}, ..., \hat{x}_{i+w}) = \psi(\phi((x_i, x_{i+1}, ..., x_{i+w}))) \tag{3.56}$$

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

$$\min_{\theta_\psi,\theta_\phi} \|(x_i, x_{i+1}, ..., x_{i+w}) - (\hat{x}_i, \hat{x}_{i+1}, ..., \hat{x}_{i+w})\|_2 \,, \forall i \in \{i, ..., T - w\} \tag{3.57}$$

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:

$$\hat{x}_j = f_{\psi\phi}(x_j) = \psi(\phi(x_j)) \tag{3.58}$$

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

$$e_j = \|x_j - \hat{x}_j\| \tag{3.59}$$

Last but not least, having an anomaly threshold δ, 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.

3.3.2 Multivariate Approaches

1. **Multi-Layer Perceptron (MLP)**
   Detecting anomalies on multivariate time series is done in two steps:

   a) Use an MLP for multivariate forecasting

   b) Compute the deviation for the forecast to the real value to determine anomalous timestamps using an appropriate distance function.

   To use an appropriate MLP architecture for multivariate time series forecasting, different approaches exist. Let $\{X_T\}$ be a multivariate time series where each $x_i \in \{X_T\}$ is a d-dimensional vector: $(x_{i,1}, x_{i,2}, ..., x_{i,d})$. Charkraborty et al. [89] evaluated three different approaches:

   a) Use a separate MLP for each time series, i.e., construct d different MLP where x_j is the input to the j-th MLP for each $j \in [1, d]$. Thus, each MLP is trained on a separate univariate time series. That means that $\forall j \in [1, ..., d]$ a separate neural network is trained on the time series $(x_{1,j}, x_{2,j}, ..., x_{T,j})$. To improve the forecast results, a sliding window with width w will be used. Therefore, the input layer of each MLP would have w neurons.
   It is clear that this approach is possible, if d is small. Charkraborty et al. used a trivariate time series and therefore had to create just three networks. But there are often multivariate time series where $d > 30$. In such cases, it is not efficient to create more than 30 distinct MLPs.

   b) The second approach is to create d different MLP, but the input would be a multivariate time series consisting of all dimensions. Thus, let $x_i \in \{X_T\}$, to predict $x_{i,j}$ the j-th MLP will be used where the input would be $(x_{i-w,1}, ..., x_{i-w,d}, x_{i-w+1,1}, ..., x_{i-w+1,d}, ..., x_{i-1,1}, ..., x_{i-1,d})$. Let $\mathcal{M}_j$ be the j-th MLP. Then it is optimized using the following objective function:

   $$\min_{\theta_j} \|\mathcal{M}_j((x_{i-w,1}, ..., x_{i-w,d}, x_{i-w+1,1}, ..., x_{i-w+1,d}, ..., x_{i-1,1}, ..., x_{i-1,d})) - (x_{i,j})\| \tag{3.60}$$

   Charkraborty et al. received the best results with this architecture.
   It is still clear that this architecture is also not appropriate for high values of d.

   c) The last option is to use just one MLP. Therefore, all dimensions will be concatenated in one vector. That means to predict x_i, the input of the MLP would be:

   $$(x_{i-w,1}, ..., x_{i-w,d}, x_{i-w+1,1}, ..., x_{i-w+1,d}, ..., x_{i-1,1}, ..., x_{i-1,d})$$

   and the output would be $(x_{i,1}, ..., x_{i,d})$ using the following objective function:

   $$\min_{\theta} \|\mathcal{M}((x_{i-w,1}, ..., x_{i-w,d}, x_{i-w+1,1}, ..., x_{i-w+1,d}, ..., x_{i-1,1}, ..., x_{i-1,d})) - (x_{i,1}, ..., x_{i,d})\| \tag{3.61}$$

   Charkraborty et al. could not achieve good results with this approach assuming that using the same weights θ for all dimensions could affect the accuracy. On the other hand, this approach is suitable when the dimension value d is high.

In our evaluation, we will switch between model 2 and 3 based on the amount of dimensions of the time series.
Finally, we have to detect the anomalies based on the deviation of the prediction to the actual value. Therefore, we compute for each $x_i \in \{X_T\}$ the deviation:

$$e_i = \|\mathcal{M}((x_{i-w}, ..., x_{i-1})) - x_i\| \tag{3.62}$$

Then, we can compute the probability distribution of the $e_i \forall i \in 1, ..., T$. Using the computed probability distribution, we can determine an appropriate δ using the τ-Percentile. Then, let $x_i \in \{X_{TEST}\}$ be a timestamp in the test dataset, we will mark it as an anomaly if and only if $\|\mathcal{M}((x_{i-w}, ..., x_{i-1})) - x_i\| > \delta$.

2. **Convolution Neural Network (CNN)**
CNNs are also used to detect anomalies on multivariate time series. Zheng et al. [73] use a specific CNN architecture to classify multivariate time series. Therefore, the multivariate time series with d dimension is split into d univariate time series. This d univariate time series are fed into d different convolution blocks that extract separate features for each dimension. The result of each channel is then combined at the end. Then, an MLP is placed to classify the time series as normal or abnormal. The drawback for such an architecture is that the MLP is designed for classification of a balanced dataset. But time series datasets are not balanced, which makes the optimization of the MLP very difficult.
Munir et al. [14] does not use different convolution layer, but one CNN for all inputs. Here again a sliding window with width w is used. Thus to predict $x_i \in \{X_T\}$, the data sequence $(x_{i-w}, ..., x_{i-1}) = (x_{i-w,1}, ..., x_{i-w,d}, ..., x_{i-1,1}, ..., x_{i-1,d})$ is used as an input requiring $w \cdot d$ input neurons where d is the dimension of a timestamp. The authors used two convolution blocks with 32 filters with max pooling layer and ReLU activation function. At the end, a full connected layer is using the filtered data to make a multidimensional prediction. As loss function, the Mean Absolute Error (MAE) function is used

$$MAE = \frac{1}{2}\sum_{j=1}^{n} |y - \hat{y}| \tag{3.63}$$

To determine whether x_i is an anomaly, the euclidean distance between the prediction to the real value is computed:

$$e_i = \sqrt{(y_i - \hat{y}_i)^2} \tag{3.64}$$

This value is used as an anomaly score. Using the training data, a threshold δ can be computed, analyzing the distribution of e_i,$\forall i \in [1, T]$.
In this thesis, we implement an architecture that is very similar to the DeepAnT approach of Munir et al. The differences are other hyperparameters, which will be mentioned in chapter 5.

3. **Residual Neural Network (ResNet)**
To detect anomaly on multivariate time series with ResNet, we use exactly the same method as the CNN approach. The only difference is that instead of convolution blocks, residual blocks are used in the ResNet approach.

4. **WaveNet**
We will also evaluate anomaly detection on multivariate time series using the WaveNet architecture. Here, we use a CNN architecture using dilation, which increases by factor of 2. The detection method is equal to the other CNN approaches. Further details about the hyperparameters are mentioned in chapter 5.

5. **Autoencoder**
Detecting anomalies with autoencoders on multivariate time series is very similar to the univariate approach. Mayu Sakurada [88] uses their autoencoder approach on multivariate time series data, which consist of several spacecraft telemetry datasets. In contrast to the LSTM methods, they only use one timestamp as input.
Thus, let $\{X_T\} \in \mathbb{R}^{TxD}$ be the dataset of time series with T timestamps where each timestamp $x_i \in \{X_T\}$ has D dimensions, then the input of the autoencoder would be x_i where $i \in [1, T]$. Hence, no sliding window will be used.
As the previous timestamps are important, we will use an extended version of this approach. Therefore, a sliding window with width w will be defined. The input for the encoder f would be:

$$f((x_i, x_{i+1}, ..., x_{i+w})) = f((x_{i,1}, x_{i,2}, ...x_{i,D}, x_{i+1,1}, ..., x_{i+1,D}, ...x_{i+w,1}, ..., x_{i+w,D})), \forall i \in [1, ..., T-w] \tag{3.65}$$

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

$$\min_{\theta_f} \|(x_i, x_{i+1}, ..., x_{i+w}) - f((x_i, x_{i+1}, ..., x_{i+w}))\|_2, \forall i \in \{i, ..., T-w\} \tag{3.66}$$

The training set is used to get the optimal weights θ_f. Instead of using the probability distribution of the error value e_i as in the LSTM approach, we decided to attach an SVM to classify the outcome of the encoder as anomaly or normal. Thus, a timestamp $x_i \in \{X_T\}$ is classified as normal or abnormal as follows:

$$SVM(\|(x_i, x_{i+1}, ..., x_{i+w}) - f((x_i, x_{i+1}, ..., x_{i+w}))\|), \forall i \in \{i, ..., T-w\} \tag{3.67}$$

where the SVM is trained on the training data of $\{X_T\}$ where each timestamp x_i is labeled as normal or abnormal. A sequence of timestamps in a sliding window in the training set is labeled as follows:

$$(x_i, ..., x_{i+w}) == abnormal \Leftrightarrow \exists j \in [i, i+w] : x_j == abnormal \tag{3.68}$$

Hence, a sequence is abnormal if an abnormal data point is present. This is mostly used in online anomaly detection where it is necessary to detect the anomaly Δ time before it happens [90].
Thus, this method is a combination of an autoencoder and SVM. It is also possible to compute the probability distribution of the errors in the training set and to classify a sequence $(x_i, ..., x_{i+p})$ as anomaly if $p(f(x_i, ..., x_w)) > \delta$ for a pre-computed δ.
We will evaluate both methods.

6. **Long Short Term Memory (LSTM) network**
LSTM networks are also used to detect anomalies in multivariate time series. The approach is very similar to the method used for univariate time series. Malhotra et al. [82] used the same approach for multivariate time series. Let $\{X_T\} \in \mathbb{R}^{T \times D}$ be a multivariate time series with T timestamps $(x_1, ..., x_T)$ and each timestamp x_i is a vector with d-dimension: $(x_i^1, x_i^2, ..., x_i^D)$. Here an LSTM model is used, which predicts the next l timestamps of x_i. Thus, let f be the appropriate function of the LSTM model, then:

$$f : \mathbb{R}^{D \cdot l} \rightarrow \mathbb{R}^{D \cdot l} \tag{3.69}$$

The prediction window has a length l and therefore:

$$\forall i \in \{1, ..., t\}, d \in D, l < i < t - l, \text{ each } x_i^d \text{ is predicted l-times} \tag{3.70}$$

The error for a multivariate timestamp x_i is computed as follows:

$$e^{(i)} = (e_{11}^{(i)}, ..., e_{d1}^{(i)}, ..., e_{1l}^{(i)}, ..., e_{dl}^{(i)}) = (f(x_{i-l})_l - x_{i+1}, f(x_{i-l+1})_{l-1} - x_{i+1}, ..., f(x_i)_1 - x_{i+1}) \tag{3.71}$$

To detect the anomalies, the same approach used for univariate time series, will be carried out: The error Vector $e^i \in \mathbb{R}^{l \cdot d}$ is fit to a multivariate Guassian distribution $\mathcal{N}(\mu, \Sigma)$ and MLE is used to estimate the parameters μ and Σ to compute the likelihood $P(e^{(i)})$. Finally, for a anomaly threshold δ, $e^{(i)}$ is an anomaly, if and only if $P(e^{(i)}) > \delta$.

7. **Gated recurrent unit (GRU)**:
   GRU models can be used like LSTM to detect anomalies in multivariate time series. The approach is equal except that the LSTM cells will be replaces by GRU cells as we did for univariate time series. This enables us to check the LSTM model and GRU model against each other.

4 Approach

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 different areas, there have 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 [91]. Afterwards, there have been attempts to compare the best performing approaches. Makridaki et al. [15] published an article where they compared the best performing statistical forecasting methods of M-3 Competition with different machine learning approaches [92]. Their paper focuses on univariate time-series models.
There have also been attempts to compare different anomaly detection methods. Goldstein et al. evaluated different unsupervised anomaly detection methods on multivariate datasets [48]. They selected 19 different unsupervised anomaly detection algorithms and evaluated them on 10 different datasets. However, the algorithms are not designed for timeseries data and the datasets are mostly not containing time series, too. There have also been different studies comparing ML methods but for a specific sort of data. For instance, Almaguer-Angeles et al. [93] compare 22 ML algorithms detecting anomalies on IoT-Datasets. There are also papers where the authors compare their anomaly detection approach with different approaches. Charkraborty [89] compare their neural network approach with ARMA model. Furthermore, there are also researches, which compare anomaly detection methods, but only evaluate on a single dataset. Lazarevic et al. compare different clustering techniques like Mining Outliers Using Distance to the k-thNearest Neighbor, Nearest Neighbor approach, Mahalanobis-distance Based Outlier Detection, LOF approach and SVM, but using just one dataset. Munir et al. [90] compared their proposed approach, FuseAD, with other state-of-the-art anomaly detection methods like LOF, iForest, OC-SVM, PCA, Twitter anomaly detection (TwitterAD)[94], and DeepAnT[14] 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 classicial machine learning approaches and neural networks detecting anomalies in univariate and multivariate 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 Makridaki et al. [92] which compares univariate forecasting methods using statistical approaches and ML methods.
In this thesis, we use different univariate time series datasets to compare the univariate approaches and different multivariate time series datasets for the multivariate anomaly approaches that have been introduced in chapter 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 and multivariate 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 timeseries requirements.
Therefore, several univariate and multivariate 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 Univariate Datasets

UD1 – Real Yahoo Services Network traffic

This dataset, which is published by Yahoo [95], 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 in average each time series consists of 1420 timestamps where 1.9% are anomalies.

UD2 – Synthetic Yahoo Services Network traffic

Yahoo made also another dataset [95] 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. In average, each time series consists of 0.3% anomalies.

UD3 – Synthetic Yahoo Services with Seasonality

This dataset also consists of 100 synthetic univariate time series [95] 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. In average, the anomalous rate of each time series is about 0.3%.

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. In average, 0.5% of the dataset is anomalous.

NYCT – NYC Taxi Dataset

This is 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.

4.2.2 Multivariate Datasets

NASA – NASA Shuttle dataset [96]

The NASA Shuttle dataset consists of a multivariate time series data one containing 49097 timestamps. The data consists of 1.89% anomalous data. Originally it was multiclass dataset consisting of seven different categories. It was transformed to an anomaly dataset, labeling class 1 as normal and all other classes as anomaly. As a multivariate time series each timestamp consists of nine numerical features. This dataset is a benchmark dataset used in more than 20 different papers.

SMTP – SMTP Dataset [96]

The dataset is a subset of the KDD Cup 1999 data, an annual Data Mining and Knowledge Discovery competition. This dataset only contains the data where the service is equal to SMTP, filtering out other services like http or ftp. The dataset contains 95156 records where about 0.03% are anomalous. It is a bi-variate dataset containing two continuous features.

SD – Synthetic dataset

This dataset is multivariate time series based on the synthetic dataset of Chandola [18]:
First, we created four univariate first autoregressive time series x_1, x_2, x_3, x_4 where we use 0.1 as coefficient for the AR model where each of them contain 3000 timestamps. Additionally, we create a vector z with the length 100 and all values equal to zero except $z_{300} = ... = z_{320} = 1, z_{600} = ... = z_{610} = -0.7, z_{1300} = ...z_{1320} = 2, z_{2100} = ... = z_{2150} = -1.5$. Additionally, the following matrix M is constructed:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

Then, the normal multivariate time series data is generated as follows:

$$N = [x_1, x_2, x_3, x_4, z] * M \tag{4.2}$$

Therefore, the multivariate time series contains about 3000 timestamps and each timestamp has five dimensions. The anomalous portion of the time series is about 3%.

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 $\mathcal{D}$:

$$\hat{x} = \frac{x - \mu}{\sigma}, \forall x \in \mathcal{D} \tag{4.3}$$

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

$$x \in [0, 1], \forall x \in \mathcal{D} \tag{4.4}$$

Normalization are sensitive to outliers. Thus, it would be inappropriate to normalize our datasets, which contain outliers.
Standardization in Equation 4.3 helps many ML methods to converge faster. Especially, in multidimensional data, i.e., multivariate time series where features have different scales, standardization speeds up the training process. Shanker et al. [97] have shown that especially in smaller datasets, the neural network yields better results when the data is standardized. Guo-Rui Ji et al. [98] mention in their paper that standardizing time series data is necessary for the SVM approach. Also other approaches which are evaluated in this thesis 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 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, Wilpen Gorr [99] believes that they are able to detect non-linear trends and seasonality in the time series dataset. Sharda et al. [61] showed empirically that neural networks are able to detect automatically the seasonality in univariate time series by evaluating about 101 different time series. They argue that neural networks will behave similar on multivariate time series. Although there have also been other works achieving similar results [100, 62].
There have also been studies achieving contradicting results. Faraway et al. [101] compared models using detrended and deseasonalized time series to networks using the origin data. They showed empirically, that the model using detrended and deseasonalized data achieved better results. Zhang et al. [102] made an empirical study comparing the performance of neural networks using detrending and deseasonalizing performed 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 will 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:

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.5}$$

Munir et al. [14] used this metric to evaluate their anomaly detection methods on different time series. Also Maya et al. [103] also used 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:

$$TPR = \frac{TP}{P} \tag{4.6}$$

and FPR is:

$$FPR = \frac{FP}{N} \tag{4.7}$$

To compute the ROC-Curve, we use different δ 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 [88, 104, 105, 106] also used the ROC-Curve and AUC-Value to compare different anomaly methods. For instance, Goldstein et al. [48] used this measure to compare different machine learning anomaly detection methods.
In a nutshell, AUC-value will be the main measure used in the experiments of this thesis in chapter 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. [103] computed the time used for training and inference separately. Goldstein et al. [48] computed the time the unsupervised anomaly detection methods needed to analyse a dataset.
The method to measure the time performance in this thesis is different to the approaches used in the mentioned papers. This is based on the fact, that those papers compare similar detection methods. For instance [48] just compares machine learning approaches especially clustering and density approaches. Also [103] compares deep learning methods making it possible to measure the training and inference phase separately. This is not possible when comparing deep learning method 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 quickly. This is completely different in clustering methods like 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% is 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 chapter 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, as described before a validation set is used. This prevents overfitting in 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:

$$\begin{aligned} \hat{x}_t =& f_{NaiveModel}((x_{t-w-1}, ..., x_{t-1})) \\ where\ & f_{NaiveModel} : \mathbb{R}^w \to \mathbb{R} \\ & f_{NaiveModel}((x_{t-w-1}, ..., x_{t-1})) \mapsto x_{t-1} \end{aligned} \tag{5.1}$$

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 naive model. Therefore, this thesis suggests the following metric to evaluate the forecasting performance of the prediction model:

$$\text{NMM}(MSE_{Targetmodel}, MSE_{NaiveModel}) = \frac{MSE_{Targetmodel}}{MSE_{NaiveModel}} \tag{5.2}$$

where NMM stands for *Naive Model Metric*. Empirically, we have noticed that a model usually has a higher AUC-Value when 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

Software

The entire algorithms have been implemented in python. For the statistical approaches mainly the module **Statsmodels** [107] was used while some custom algorithms like PCI was implemented from scratch. The classical machine learning approches have been mostly implemented using **Scikit-learn (Sklearn)** [108] and for the deep learning approaches the **tensorflow** [109] and **keras** library [110] have been used.

Hardware

The training and testing of the models were performed on the following hardware:

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

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

5.2.2 Hyperparameter Tuning

General Hyperparameters

Some of the hyperparemters are general and dependent on the dataset, unless another value is listed explicitly for a specific approach. These are the following:

Dataset	Hyperparameter	Value
Yahoo Services Network traffic	Sliding window width w	30
All datasets	Ratio of training set	0.3
	Ratio of test set	0.7
	Ratio of validation set	0.1 of test set

Statistical approaches

The hyperparameters of the statistical approaches are listed based on the specified algorithm:

Univariate 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 brute force grid optimizer to find the best values using MLE
ES	α, β, γ	Use brute force grid optimizer to find the best values using MLE
PCI	k	30
	α	98.5

The following are the parameters for the multivariate statistical approaches, which achieved the best results on the different multivariate datasets

Multivariate Statistical Approaches		
Model	Hyperparameter	Value
Average AR-Model (Avg_{AR})	maximal lag	$12 \cdot (\frac{\lvert X_{Train} \rvert}{100})^{\frac{1}{4}}$
	Fitting method	Conditional maximum likelihood using OLS
	Aggregation functions	$max, min, mean$
Average MA-Model (Avg_{MA})	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
	Aggregation functions	$max, min, mean$
VAR-Model	maximal lag	$12 \cdot (\frac{\lvert X_{Train} \rvert}{100})^{\frac{1}{4}}$
	Fitting method	Conditional maximum likelihood using OLS
VARMA-Model	maximal lag	$12 \cdot (\frac{\lvert X_{Train} \rvert}{100})^{\frac{1}{4}}$
	(p, q)	(4,2)
	Fitting method	Maximizing Conditional Sum of Squares likelihood
	Maximal iteration	100 and 1000
PCI	k	100
	α	98.5
Projection with AR-Model (Proj_{AR})	maximal lag	$12 \cdot (\frac{\lvert X_{Train} \rvert}{100})^{\frac{1}{4}}$
Projection with MA-Model (Proj_{MA})	maximal lag	$12 \cdot (\frac{\lvert X_{Train} \rvert}{100})^{\frac{1}{4}}$
Projection with OVSVM-Model (Proj_{OVSVM})	Upper bound of outliers	0.4
	Sliding window w	10
Projection with XGBoost (Proj_{XGB})	Sliding window w	10

Machine Learning approaches

The hyperparameters of the machine learning approaches used for univariate time series are listed based on the specified algorithm:

Univariate 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

In the following table the hyperparameters of the machine learning approaches for the multivariate time series are listed. If the parameter values differ from the datasets they are listed separately.

Multivariate Machine Learning Approaches		
Model	Hyperparameter	Value
Average One-Class SVM (Avg_{XGB})	kernel	Radial basis function kernel(RBF)
	Upper bound of outliers	0.5
	Aggregation functions	$max, min, mean$
Average XGBoosting (Avg_{XGB})	Max Tree depth	3
	Learning rate	0.1
	Number of estimators	1000
	Loss function	MSE
	Sliding window w	100
	Aggregation functions	$max, min, mean$
Multivariate One-Class SVM (OC-SVM)	kernel	Radial basis function kernel(RBF)
	Upper bound of outliers	0.9
	sliding window w	350
Multivariate XGBoosting (XGB)	Max Tree depth	3
	Learning rate	0.1
	Number of estimators	1000
	Loss function	MSE
	Sliding window w	108
LOF	k	6
	distance function	Minkowski distance
DBSCAN	ϵ	0.3
	μ	30
	distance function	Euclidean distance
	sliding window w	100
Isolation Forest (iForest)	Number of iTrees	20
	Contamination value	Find automatically
	sliding window w	100

Deep Learning approaches

The hyperparameters of the deep learning approaches for univariate timeseries are listed based on the specified algorithm:

Univariate 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 3.7 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

In the following table, the hyperparameters of the multivariate deep learning methods are listed:

Multivariate Deep Learning Approaches		
Model	Hyperparameter	Value
MLP	Number of hidden layers	2
	Neurons in each hidden layer	100, 300, 400, 200
	Activation function	ReLU
	Optimizer, Loss	Adam, MSE
	Batch size,Epochs	32, 30
CNN	Architecture	3 Convolution Blocks with Max-Pooling and ReLU function, then one Dense layer with 18 neurons and ReLU
	Sliding window w	30
	Filters	8,8,8 with kernel size 2
	Optimizer, Loss	Adam, MSE
	Batch size, Epochs	32, 20
CNN+Batch Normalization (CNN_B)	Architecture	2 Convolution Blocks with Batch-Normalization with ReLU then one Dense layer with 18 neurons and ReLU
	Sliding window w	30
	Filters	8,8,8 with kernel size 2
	Optimizer, Loss	Adam, MSE
	Batch size, Epochs	32,20
CNN+Residual Blocks (CNN Residual)	Architecture	1 Convolution Block with One Residual Block 3.7 then one Dense Layer with 18 neurons and ReLU
	Sliding window w	180
	Filters	3,3,3 with kernel size 1
	Optimizer, Loss	Adam, MSE
	Batch size, Epochs	32, 60
WaveNet	Architecture	4 Convolution Blocks with Max-Pool of size 2 and ReLU function, with Dilation rate 1,2,4,8 then one Dense layer with 18 neurons and ReLU
	Sliding window w	30
	Filters	8,8,8,8 with kernel size 2
	Optimizer	Adam, MSE
	Batch size, Epochs	32, 20
LSTM	Architecture	2 stateful LSTM Layer
	Sliding window w	30
	Filters	8,8
	Optimizers, Loss	Adam, MSE
	Batch size, Epochs	32, 3
GRU	Architecture	2 stateful LSTM Layer
	Sliding window w	30
	Filters	8,8
	Optimizers, Loss	Adam, MSE
	Batch size, Epochs	32, 3
Autoencoder	Architecture	2 Encoding Layers (5,3), 2 Decoding Layers (3,5)
	Activation functions	ReLU for Decoding and Encoding, linear for output
	Optimizer, Loss	Adam, MSE
	Batch size, Epochs	32, 3

6 Results

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

6.1 Results of Univariate Approaches

6.1.1 AUC-Values

The following table 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

The best results are highlighted in the table. 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 6.2:

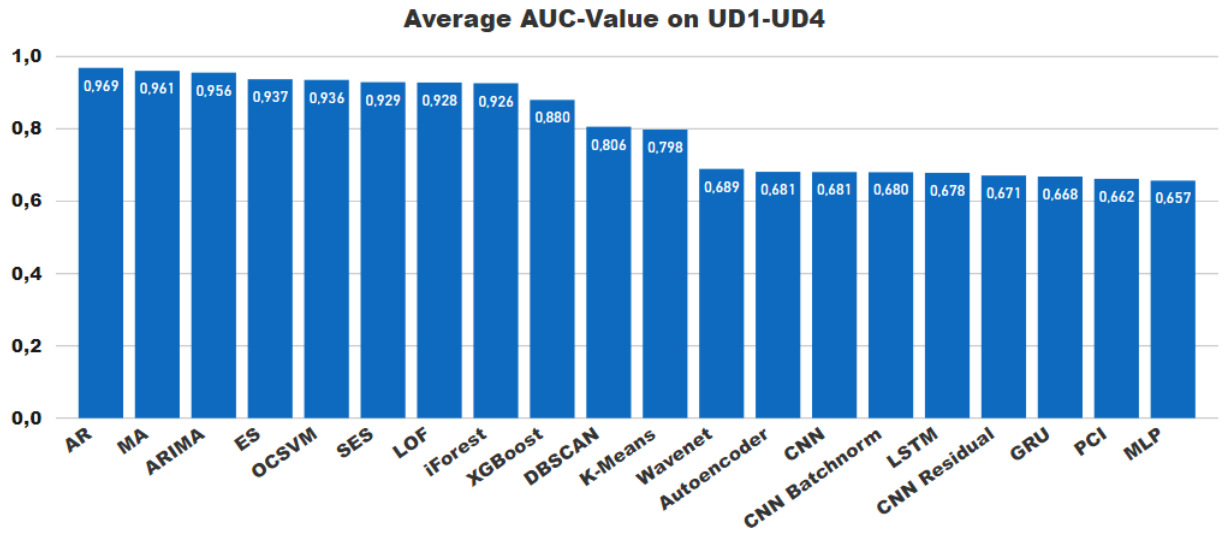

Figure 6.1: Average AUC-Value computed on UD1-UD4

This figure 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 shows how the results:

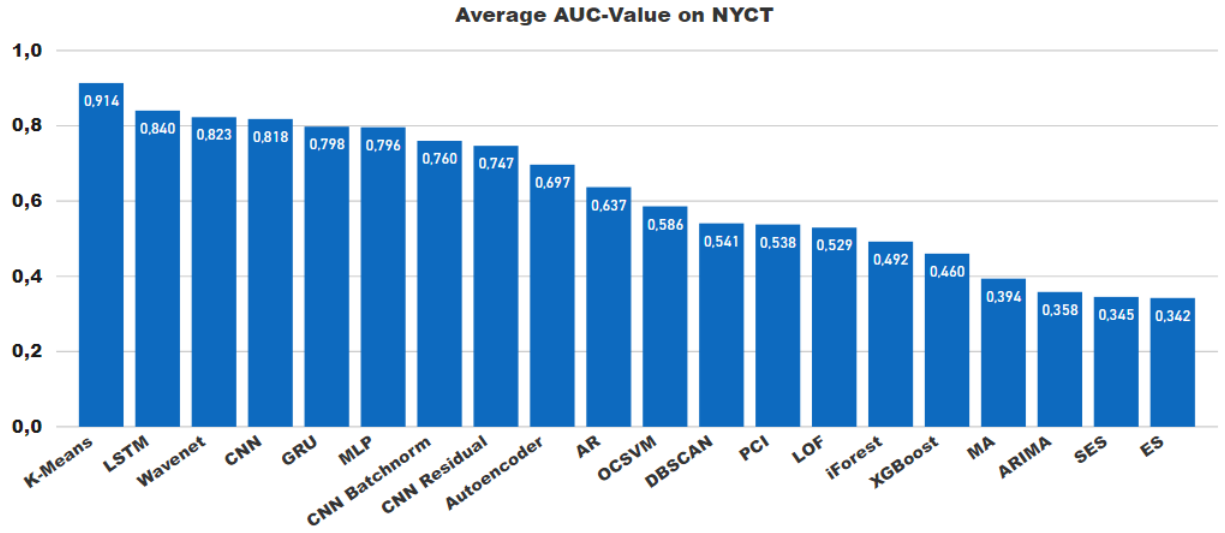

Figure 6.2: Average AUC-Value computed on UD1-UD4

The main reason to this poor performance of the statistical approach is the fact that the NYC timeseries 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.

6.1.2 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 timeseries	**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 timeseries	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 timeseries	20.15	29.49	35.38	100.92	47.0	147.36	135.57	25.65

The results 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 the following table, 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

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.

6.1.3 Computation Time vs AUC-Value

An interesting relation is the accuracy a method can achieve compared to the computation time it requires. Figure 6.3 displays how the different univariate 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.

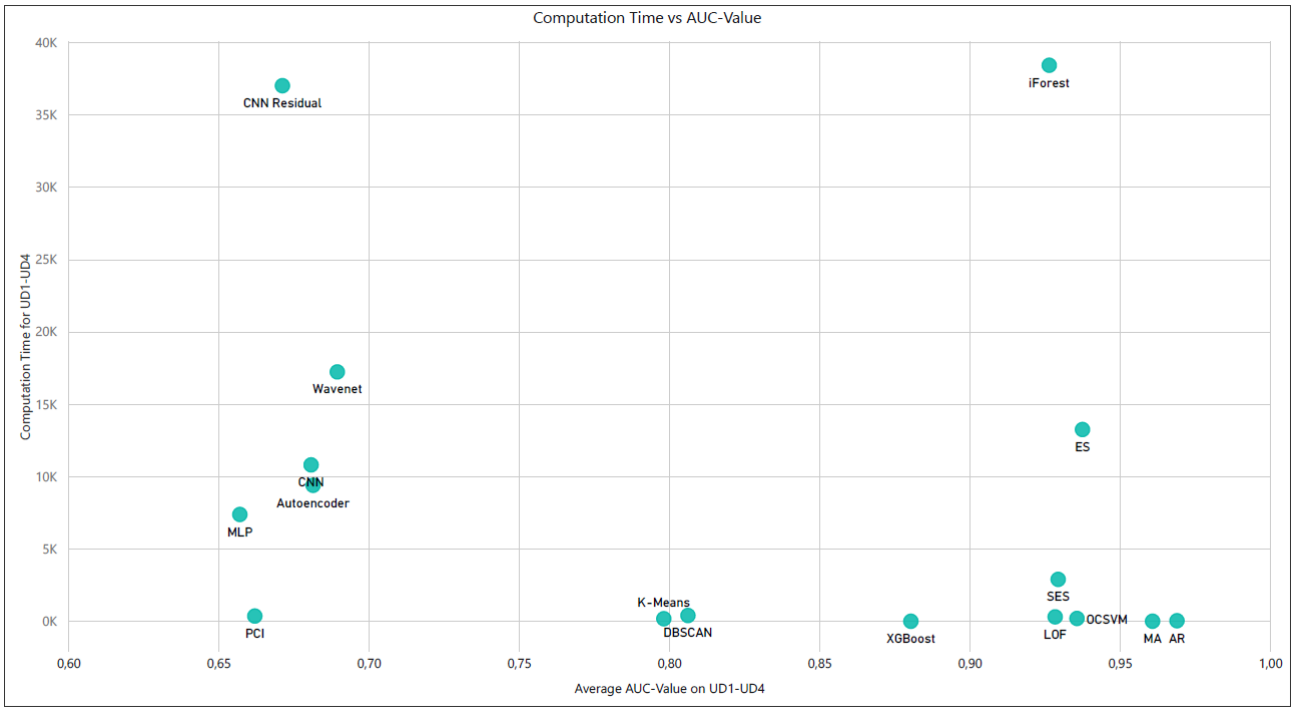

Figure 6.3: Average AUC-Value vs computation time

6.2 Results of Multivariate Approaches

6.2.1 AUC-Values

In the following table, the AUC values of each algorithm on the multivariate data sets are listed. For each dataset the algorithm with the highest AUC-Value is highlighted:

Dataset	Avg_{AR}	Avg_{MA}	Avg_{XGB}	Avg_{OCSVM}	VAR	VARMA	PCI
SD	0.5785	0.5047	0.593	0.6163	0.5632	0.7337	0.5883
SMTP	0.6055	0.8203	**0.9058**	0.7326	0.8669	0.8292	0.7996
NASA	0.91	0.7907	0.8323	**0.9212**	0.4632	0.5781	0.4866
Dataset	Proj_{AR}	Proj_{MA}	Proj_{XGB}	Proj_{OCSVM}			
SD	0.9542	**0.9611**	0.9557	0.9524			
SMTP	0.4432	0.5872	0.4178	0.811			
NASA	0.4706	0.4714	0.4658	0.4684			
Dataset	OCSVM	XGB	LOF	DBSCAN	iForest		
SD	0.796	0.7368	0.6897	0.5069	0.6354		
SMTP	0.6496	0.7263	0.6747	0.6929	0.7201		
NASA	0.885	0.4738	0.8158	0.4633	0.8814		
Dataset	MLP	CNN	CNN B	CNN Residual	Wavenet	LSTM	GRU	Autoencoder
SD	0.85	0.922	0.8943	0.9269	0.9336	0.9364	0.9299	0.8926
SMTP	0.5878	0.6110	0.7102	0.6447	0.646	0.4413	0.5294	0.9030
NASA	0.4727	0.4689	0.4719	0.4732	0.4722	0.4703	0.4708	0.4526

As stated in Chapter 5, we used the minimize, maximize and mean function for the statistical projection methods, i.e., Proj_{AR}, Proj_{MA}, Proj_{XGB} and Proj_{OCSVM}, where most of them achieved the best results with the minimize function. Figure 6.4 shows the average AUC-Value achieved by each approach on the SD and SMTP datasets. These two datasets have been selected for this figure to have the average value of one real and one synthetic dataset:

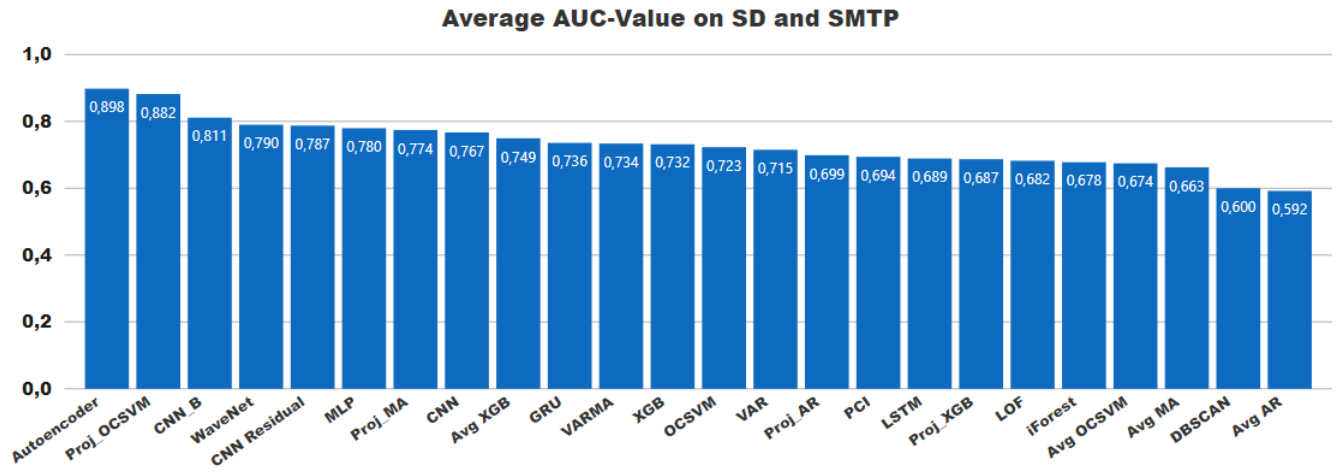

Figure 6.4: AUC-Value on the SD and SMTP datasets

Figure 6.4 shows that the deep learning approaches achieve high AUC-Value for the two datasets. Five of the best six approaches are deep learning approaches.
Another interesting outcome of the evaluation is illustrated in figure 6.5:

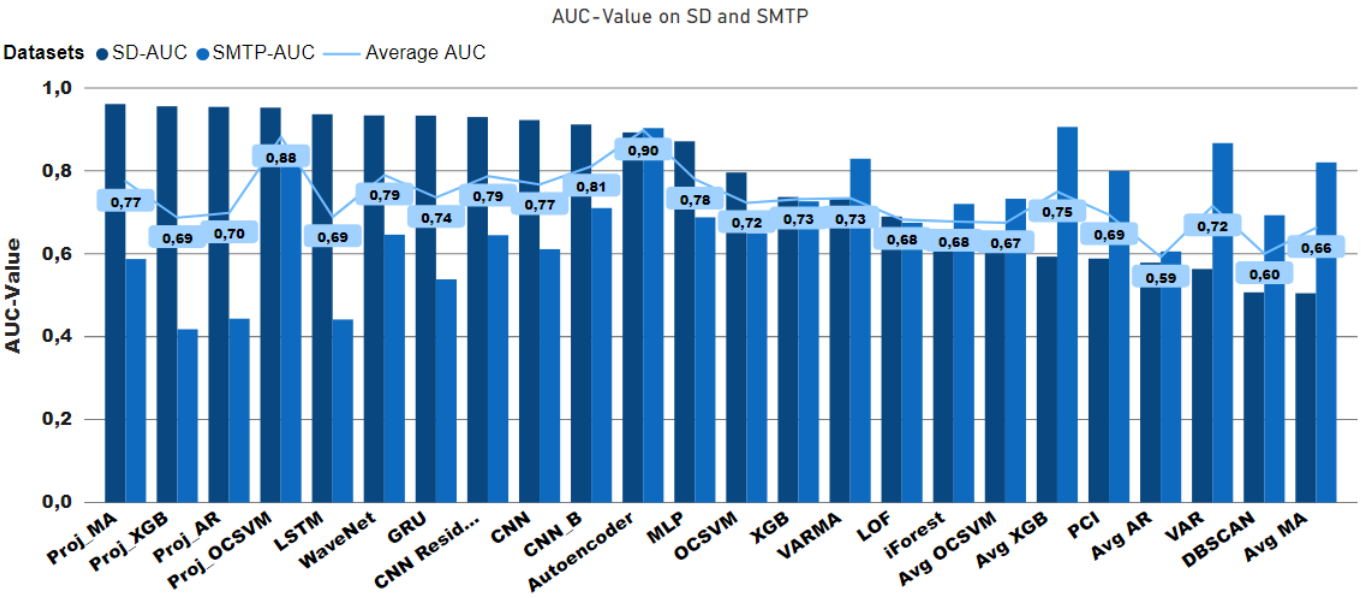

Figure 6.5: Separate AUC-Values of the SD and SMTP datasets and average AUC-Value

Figure 6.5 shows that the statistical methods perform much better on the synthetic data compared to the ML and Deep Learning approaches. This is due to the fact that the dataset is created by several AR-Models. Interestingly, the figure also shows how the AUC values of the statistical models decreases on the SMTP dataset and how the values of the deep learning methods improve. Also ML methods like LOF, AVG_{XGB} or AVG_{OCSVM} achieve high results. The blue line illustrates the average AUC-Value on the two datasets where the autoencoder and the hybrid approach Proj_{OCSVM} consisting of statistical Projection and Machine

learning methods OC-SVM achieve the best results. It also interesting to notice that the GRU model achieves almost the same results as the LSTM model, although GRU is less complicated than LSTM as decribed in Chapter 3.
Another important observation of the experiments was the performance of the different distance functions in anomaly detection methods, which are based on time series forecasting. The results in table 6.1 show that using the mahalanobis distance improves the AUC-Value compared to the euclidean distance:

Dataset	Function	MLP	CNN	CNN B	CNN Residual	WaveNet	LSTM	GRU	Autoencoder
SD	Mahalanobis	0.8712	0.9225	0.9118	0.9297	0.9336	0.9364	0.9332	0.8926
	Euclidean	0.6997	0.7457	0.7545	0.7592	0.7787	0.7589	0.7467	0.8726
SMTP	Mahalanobis	0.6878	0.611	0.7102	0.6447	0.646	0.4413	0.5382	0.903
	Euclidean	0.6076	0.5941	0.5494	0.5067	0.5178	0.404	0.4981	0.8438
NASA	Mahalanobis	0.4727	0.4689	0.4719	0.4732	0.4722	0.4703	0.47089	0.4526
	Euclidean	0.47635	0.476	0.4747	0.4749	0.4756	0.4751	0.4751	0.4513

Table 6.1: Mahalanobis vs. Euclidean distance

Almost in all approaches with all datasets, the mahalanobis distance achieves better results than the euclidean distance where the difference between the AUC-Values varies on the different datasets.

6.2.2 Computation Time

The computation time is measured for all the datasets and is listed for each approach in seconds in the following table:

Dataset	Avg_{AR}	Avg_{MA}	Avg_{XGB}	Avg_{OCSVM}	VAR	VARMA	PCI
SD	1.401	0.593	0.387	10.783	0.198	117.93	0.7721
SMTP	21.91	4.403	**1.338**	103.23	2.429	5377	1.37
NASA	57.412	10.399	3.760	337	**3**	11387	3.454
Dataset	Proj_{AR}	Proj_{MA}	Proj_{XGB}	Proj_{OCSVM}			
SD	0.2534	**0.075**	2.3698	0.5967			
SMTP	15.148	7.111	4.6182	64.5			
NASA	11.682	8.161	8.199	31.756			
Dataset	OCSVM	XGB	LOF	DBSCAN	iForest		
SD	17	13	9	3	231		
SMTP	588	76	72	100	7524		
NASA	431	261	53	71	4000		
Dataset	MLP	CNN	CNN B	CNN Residual	Wavenet	LSTM	GRU	Autoencoder
SD	11	11	11	21	11	164	133	34
SMTP	151	117	151	578	150	4673	3874	70
NASA	88	58	84	314	69	3001	1844	32

Similar to the univariate approaches, the statistical approaches on multivariate time series benefit from higher performance while some machine learning approaches like iForest or deep learning approaches like LSTM suffer from high computation times. The results also show that the variance between the measured computation time values is huge. For instance, iForest takes about 300-times longer than the Proj_{MA} method.

6.2.3 Computation Time vs AUC-Value

As shown in evaluation of the univariate approaches, analysing the AUC-value and computation time separately could misguide us. Therefore, we will analyze the relationship between these two measurements. Figure 6.6 shows the relationship between these two variables based on the inspected approach.

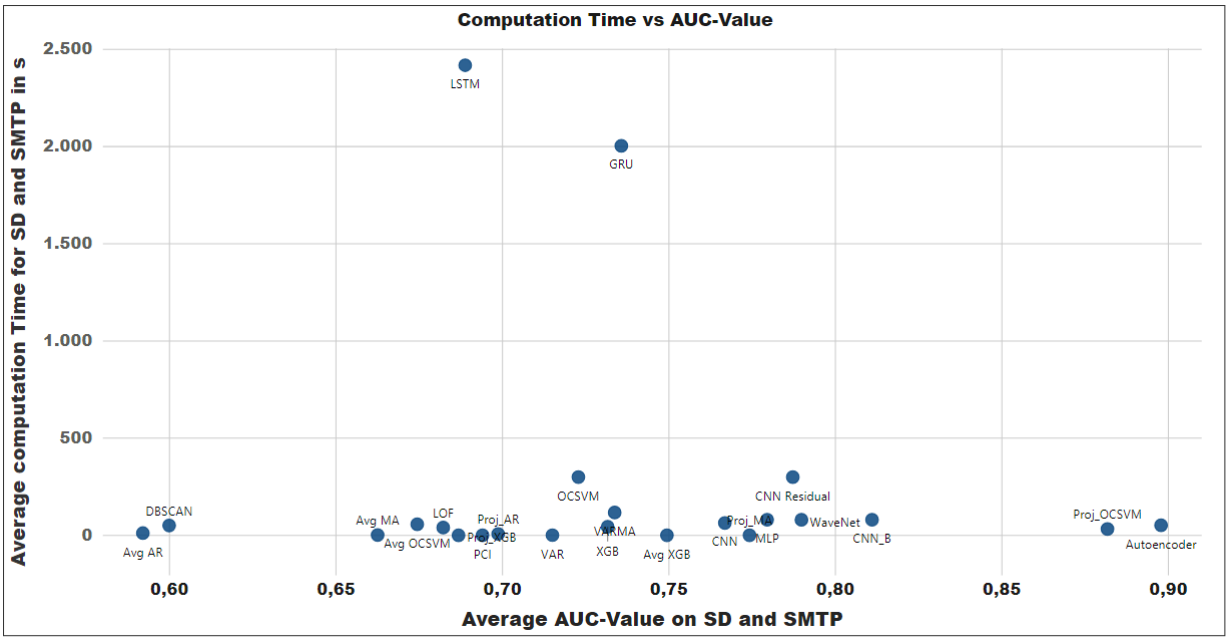

Figure 6.6: Average AUC-Value vs computation time

While for the univariate approaches the statistical methods occupied the bottom right corner, on multivariate approaches another distribution is noticed. Here, deep learning methods like autoencoder achieve high AUC-Value while also having low computation time. Also other deep learning methods like the different variations of CNN-Model achieve high AUC-Values while profiting from low computation times. Figure 6.6 also shows that hybrid approaches like $Proj_{OCSVM}$ gain high AUC-values as well as low computation times.

6.3 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 appraoches, 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 achieved notably higher AUC-values. This finding is important, because a similar comparison of statistical approaches and machine learning approaches for forecasting univariate timeseries concluded that statistic approaches dominated the machine learning and deep learning approaches across in both accuracy and for all forecasting horizons they examined [15]. Hence, these findings show how anomaly detection on univariate time series data differ from exclusive univariate forecasting.
The results of the multivariate time series showed a shift in the achieved AUC values. Compared to the univariate time series where the neural network approaches generally performed poorly, in multivariate time series most of them could obtain high AUC values. Especially on datasets that have not been simulated explicitly by autoregression models like the NASA and SMTP datasets, deep learning approaches like autoencoders performed very well. The results highlighted also the fact that choosing an appropriate distance function can increase significantly the accuracy of deep learning method. During the training process, we also observed that neural networks overfit much faster on time series data than we were used to. While in computer vision tasks, large amount of trainable parameters were used, in the timeseries of the mentioned datasets we achieved the best results by decreasing the amount of layers, amount of neurons, kernel sizes and kernel numbers. We also observed that the number of epochs should be kept low, otherwise the model overfits resulting in low AUC values. The explicit hyperparameter values have been listed in section 5.2.2. It also showed up that for multivariate timerseries LSTM based approaches didn't outperformed simpler models like CNNs. This is in spite of the fact, that LSTM and GRU models are the most time consuming neural network approaches and need ten times more training time on the small datasets. On the bigger datasets like SMTP and NASA the computation time difference reaches a ration of 30:1.
The final interesting observation was the fact, that hybrid approaches consisting of a statistical and machine learning part like Avg_{OCSVM} achieved promising AUC values while still need a limited computation time. We recommend that these methods should be analyzed further in future works.

7 Conclusion and Future Works

In this thesis for the first time 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 and 24 multivariate anomaly detection methods by using five univariate time series datasets consisting of 368 univariate time series and three multivariate time series datasets to provide a reliable contribution for the research community about the performance of these three classes of anomaly detection methods. Additionally, by analysing both univariate and multivariate approaches, we wanted to evaluate if statistical, machine learning and deep learning methods behave differently on univariate and multivariate timeseries.

The experiments showed that the statistical approaches perform best on univariate time series, especially in 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. On the other hand, deep learning approaches showed promising performance on multivariate time series. 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 results have also shown that hybrid approaches consisting of statistical and machine learning components can achieve satisfactory outcomes. Therefore, these methods are the subject of future research. Additionally, one related field, which is also of current interest, is Online Anomaly Detection. This thesis focused on anomaly detection on static data, but can be extended to evaluate streaming data.

In this thesis, 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 thesis by evaluating the approaches on different domain-specific datasets.

Finally, several attempts can be made to extend the results of this thesis. However, to the best of our knowledge, this is the first attempt to provide a broad evaluation of different anomaly detection techniques on timeseries data. We hope that this thesis and the corresponding experiments aid other researchers in selecting an appropriate anomaly detection method when analysing timeseries.

Bibliography

[1] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[2] I. Chang, G. Tiao, and C. Chen, "Estimation of Time Series Parameters in the Presence of Outliers," *Technometrics*, vol. 30, pp. 193–204, 05 1988.

[3] C. C. Aggarwal, *Outlier Analysis*. Springer Publishing Company, Incorporated, 2nd ed., 2016.

[4] C. M. Salgado, C. Azevedo, H. Proença, and S. M. Vieira, *Noise Versus Outliers*, pp. 163–183. Cham: Springer International Publishing, 2016.

[5] N. Günnemann, S. Günnemann, and C. Faloutsos, "Robust Multivariate Autoregression for Anomaly Detection in Dynamic Product Ratings," in *WWW*, 2014.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–72, 2009.

[7] F. E. Grubbs, "Procedures for Detecting Outlying Observations in Samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[8] K. Ord, "Outliers in Statistical Data," *International Journal of Forecasting*, vol. 12, no. 1, pp. 175–176, 1996.

[9] D. Hawkins, *Identification of Outliers*. Monographs on Applied Probability and Statistics, London [u.a.]: Chapman and Hall, 1980.

[10] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A Review of Novelty Detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[11] W. Wei, *Time Series Analysis: Univariate and Multivariate Methods*, vol. 33. 01 1989.

[12] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. OTexts, 2014.

[13] M. Gupta, J. Gao, C. Aggarwal, and J. Han, *Outlier Detection for Temporal Data*. Morgan & Claypool Publishers, 2014.

[14] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019.

[15] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning Forecasting Methods: Concerns and Ways Forward," *PLOS ONE*, vol. 13, pp. 1–26, 03 2018.

[16] L. Breiman, "Statistical Modeling: The Two Cultures," *Statistical Science*, 2001.

[17] G. Papacharalampous, H. Tyralis, and D. Koutsoyiannis, "Comparison of Stochastic and Machine Learning Methods for Multi-Step Ahead Forecasting of Hydrological Processes," *Stochastic Environmental Research and Risk Assessment*, vol. 33, pp. 481–514, Feb 2019.

[18] V. Kumar, A. Banerjee, and V. Chandola, "Anomaly Detection for Symbolic Sequences and Time Series Data," 2009.

[19] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. New York, NY, USA: John Wiley & Sons, Inc., 1987.

[20] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden-Day, Inc., 1990.

[21] R. Brown, *Exponential Smoothing for Predicting Demand*. Little, 1956.

[22] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, "A State Space Framework for Automatic Forecasting Using Exponential Smoothing Methods," *International Journal of Forecasting*, vol. 18, no. 3, pp. 439–454, 2002.

[23] Y. Yu, Y. Zhu, S. Li, and D. Wan, "Time Series Outlier Detection Based on Sliding Window Prediction," 2014.

[24] R. S. Tsay, D. Peña, and A. E. Pankratz, "Outliers in Multivariate Time Series," *Biometrika*, vol. 87, pp. 789–804, 12 2000.

[25] M. Gupta, A. B. Sharma, H. Chen, and G. Jiang, "Context-Aware Time Series Anomaly Detection for Complex Systems," 2013.

[26] P. Galeano, D. Peña, and R. S. Tsay, "Outlier Detection in Multivariate Time Series by Projection Pursuit," 2006.

[27] C. Jordan, "Essai sur la Géométrie à n Dimensions," *Bulletin de la Société Mathématique de France*, vol. 3, pp. 103–174, 1875.

[28] M. Kano, S. Hasebe, I. Hashimoto, and H. Ohno, "A New Multivariate Statistical Process Monitoring Method Using Principal Component Analysis," 2001.

[29] C. Leigh, O. Alsibai, R. J. Hyndman, S. Kandanaarachchi, O. C. King, J. M. McGree, C. Neelamraju, J. Strauss, P. D. Talagala, R. D. R. Turner, K. L. Mengersen, and E. E. Peterson, "A Framework for Automated Anomaly Detection in High Frequency Water-Quality Data from in Situ Sensors.," *The Science of the total environment*, vol. 664, pp. 885–898, 2018.

[30] T. Januschowski, J. Gasthaus, Y. Wang, D. Salinas, V. Flunkert, M. Bohlke-Schneider, and L. Callot, "Criteria for Classifying Forecasting Methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 167 – 177, 2020. M4 Competition.

[31] 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*, (Berkeley, Calif.), pp. 281–297, University of California Press, 1967.

[32] T. Idé, "Why Does Subsequence Time-Series Clustering Produce Sine Waves?," in *Knowledge Discovery in Databases: PKDD 2006* (J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, eds.), (Berlin, Heidelberg), pp. 211–222, Springer Berlin Heidelberg, 2006.

[33] T. Yairi, Y. Kato, and K. Hori, "Fault Detection by Mining Association Rules from House-keeping Data," in *In Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.

[34] E. Friedman and T. Dunning, “Practical Machine Learning: A New Look at Anomaly Detection,” 2014.

[35] P. S. Bradley and U. M. Fayyad, “Refining Initial Points for K-Means Clustering,” in *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML ’98, (San Francisco, CA, USA), pp. 91–99, Morgan Kaufmann Publishers Inc., 1998.

[36] E. Keogh and J. Lin, “Clustering of Time-Series Subsequences is Meaningless: Implications for Previous and Future Research,” *Knowledge and Information Systems*, vol. 8, pp. 154–177, Aug 2005.

[37] R. Fujimaki, S. Hirose, and T. Nakata, *Theoretical Analysis of Subsequence Time-Series Clustering from a Frequency-Analysis Viewpoint*, pp. 506–517.

[38] 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*, pp. 902–907, Dec 2009.

[39] J. R. Chen, “Making Subsequence Time Series Clustering Meaningful,” in *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM ’05, (Washington, DC, USA), pp. 114–121, IEEE Computer Society, 2005.

[40] 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*, pp. 547–556, Dec 2011.

[41] S. J. Zolhavarieh, S. Aghabozorgi, and T. Wah, “A Review of Subsequence Time Series Clustering,” *The Scientific World Journal*, 06 2014.

[42] M. Ester, H.-P. Kriegel, J. Sander, and X. 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, pp. 226–231, AAAI Press, 1996.

[43] Z. He, X. Xu, and S. Deng, “Discovering Cluster-Based Local Outliers,” *Pattern Recognition Letters*, vol. 24, no. 9, pp. 1641–1650, 2003.

[44] M. Çelik, F. Dadaer-Çelik, and A. . Dokuz, “Anomaly Detection in Temperature Data Using DBSCAN algorithm,” in *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pp. 91–95, June 2011.

[45] 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)*, vol. 2, pp. 1–4, Aug 2016.

[46] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying Density-based Local Outliers,” *SIGMOD Rec.*, vol. 29, pp. 93–104, May 2000.

[47] S. Oehmcke, O. Zielinski, and O. Kramer, “Event Detection in Marine Time Series Data,” in *KI*, 2015.

[48] M. Goldstein and S. Uchida, “A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data,” *PLOS ONE*, vol. 11, pp. 1–31, 04 2016.

[49] F. T. Liu, K. M. Ting, and Z. hua Zhou, “Isolation Forest,” in *In ICDM 08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. IEEE Computer Society*, pp. 413–422.

[50] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-Based Anomaly Detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, pp. 3:1–3:39, Mar. 2012.

[51] Z. Ding and M. Fei, "An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data Using Sliding Window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013. 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013.

[52] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support Vector Method for Novelty Detection," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, (Cambridge, MA, USA), pp. 582–588, MIT Press, 1999.

[53] 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.*, vol. 3, pp. 1741–1745 vol.3, July 2003.

[54] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a Time Series," *Phys. Rev. Lett.*, vol. 45, pp. 712–716, Sep 1980.

[55] R. Zhang, S. Zhang, S. Muthuraman, and J. 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, (Stevens Point, Wisconsin, USA), pp. 31–37, World Scientific and Engineering Academy and Society (WSEAS), 2007.

[56] R. Zhang, S. Zhang, Y. Lan, and J. Jiang, "Network Anomaly Detection Using One Class Support Vector Machine," 2008.

[57] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *CoRR*, vol. abs/1603.02754, 2016.

[58] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: a Statistical View of Boosting," *Annals of Statistics*, vol. 28, p. 2000, 1998.

[59] D. Elsner, P. A. Khosroshahi, A. D. MacCormack, and R. Lagerström, "Multivariate Unsupervised Machine Learning for Anomaly Detection in Enterprise Applications," in *HICSS*, 2019.

[60] B. Lamrini, A. Gjini, S. Daudin, F. Armando, P. Pratmarty, and L. Travé-Massuyès, "Anomaly Detection Using Similarity-Based One-Class SVM for Network Traffic Characterization," 08 2018.

[61] R. Sharda and R. B. Patil, "Connectionist Approach to Time Series Prediction: An Empirical Test," *Journal of Intelligent Manufacturing*, vol. 3, pp. 317–323, Oct 1992.

[62] Z. Tang and P. Fishwick, "Feedforward Neural Nets as Models for Time Series Forecasting," *INFORMS Journal on Computing*, vol. 5, pp. 374–385, 11 1993.

[63] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 ed., 2009.

[64] J. Ba and R. Caruana, "Do Deep Nets Really Need to be Deep?," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2654–2662, Curran Associates, Inc., 2014.

[65] E. Haselsteiner and G. Pfurtscheller, "Using Time-Dependent Neural Networks for EEG Classification," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, pp. 457–463, Dec 2000.

[66] A. D. Back and A. C. Tsoi, "FIR and IIR Synapses, a New Neural Network Architecture for Time Series Modeling," *Neural Computation*, vol. 3, pp. 375–385, Sep. 1991.

[67] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-parameter Optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, (USA), pp. 2546–2554, Curran Associates Inc., 2011.

[68] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[69] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[70] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *CoRR*, vol. abs/1807.05511, 2018.

[71] D. Scherer, A. Müller, and S. 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, (Berlin, Heidelberg), pp. 92–101, Springer-Verlag, 2010.

[72] H. Wu and X. Gu, "Max-Pooling Dropout for Regularization of Convolutional Neural Networks," *CoRR*, vol. abs/1512.01400, 2015.

[73] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks," in *Web-Age Information Management* (F. Li, G. Li, S.-w. Hwang, B. Yao, and Z. Zhang, eds.), (Cham), pp. 298–310, Springer International Publishing, 2014.

[74] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[75] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *CoRR*, vol. abs/1502.03167, 2015.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.03385, 2015.

[77] Z. Wang, W. Yan, and T. Oates, "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline," *CoRR*, vol. abs/1611.06455, 2016.

[78] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," 2016.

[79] A. Borovykh, S. M. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," 2017.

[80] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[81] 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)*, pp. 1–7, Oct 2015.

[82] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series," in *ESANN*, 2015.

[83] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," *CoRR*, vol. abs/1406.1078, 2014.

[84] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *CoRR*, vol. abs/1503.04069, 2015.

[85] W. Wu, L. He, and W. Lin, "Local Trend Inconsistency: A Prediction-driven Approach to Unsupervised Anomaly Detection in Multi-seasonal Time Series," *ArXiv*, vol. abs/1908.01146, 2019.

[86] C. Zhou and R. 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, (New York, NY, USA), pp. 665–674, ACM, 2017.

[87] C. Baur, B. Wiestler, S. Albarqouni, and N. Navab, "Deep Autoencoding Models for Unsupervised Anomaly Segmentation in Brain MR Images," *CoRR*, vol. abs/1804.04488, 2018.

[88] M. Sakurada and T. 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, (New York, NY, USA), pp. 4:4–4:11, ACM, 2014.

[89] K. Chakraborty, K. Mehrotra, C. K. Mohan, and S. Ranka, "Forecasting the Behavior of Multivariate Time Series Using Neural Networks," *Neural Networks*, vol. 5, no. 6, pp. 961–970, 1992.

[90] M. Munir, S. A. Siddiqui, M. A. Chattha, A. Dengel, and S. Ahmed, "FuseAD: Unsupervised Anomaly Detection in Streaming Sensors Data by Fusing Statistical and Deep Learning Models," *Sensors - Open Access Journal (sensors)*, vol. 19, pp. 1–15, 5 2019.

[91] S. Makridakis, A. Andersen, R. F. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. L. Winkler, "The Accuracy of Extrapolation (Time Series) Methods: Results of a Forecasting Competition," 1982.

[92] S. Makridakis and M. Hibon, "The M3-Competition: Results, Conclusions and Implications," *International Journal of Forecasting*, vol. 16, no. 4, pp. 451–476, 2000. The M3- Competition.

[93] 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)*, pp. 491–495, April 2019.

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

[95] "Yahoo! Webscope Dataset Ydata-Labeled-Time-Series-Anomalies-v1$_0$."

[96] M. Lichman, "UCI Machine Learning Repository," 2013.

[97] M. Shanker, M. Hu, and M. Hung, "Effect of Data Standardization on Neural Network Training," *Omega*, vol. 24, no. 4, pp. 385–397, 1996.

[98] 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*, vol. 5, pp. 2735–2739, Aug 2007.

[99] W. L. Gorr, "Editorial: Research Prospective on Neural Network Forecasting," *International Journal of Forecasting*, vol. 10, no. 1, pp. 1–4, 1994.

[100] P. H. Franses and G. Draisma, "Recognizing Changing Seasonal Patterns Using Artificial Neural Networks," *Journal of Econometrics*, vol. 81, no. 1, pp. 273–280, 1997.

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

[102] P. Zhang and M. Qi, "Neural Network Forecasting For Seasonal And Trend Time Series," *European Journal of Operational Research*, vol. 160, pp. 501–514, 02 2005.

[103] S. Maya, K. Ueno, and T. Nishikawa, "dLSTM: A New Approach for Anomaly Detection Using Deep Learning with Delayed Prediction," *International Journal of Data Science and Analytics*, vol. 8, pp. 137–164, Sep 2019.

[104] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series," 04 2015.

[105] T. Amarbayasgalan, B. Jargalsaikhan, and K. Ryu, "Unsupervised Novelty Detection Using Deep Autoencoders with Density Based Clustering," *Applied Sciences*, vol. 8, 08 2018.

[106] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," in *In Proceedings of SIAM Conference on Data Mining*, 2003.

[107] S. Seabold and J. Perktold, "Statsmodels: Econometric and Statistical Modeling with Python," in *9th Python in Science Conference*, 2010.

[108] 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*, vol. 12, pp. 2825–2830, 2011.

[109] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. Software available from tensorflow.org.

[110] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Mohammad Braei, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäSS §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, den 29. November 2019

Mohammad Braei