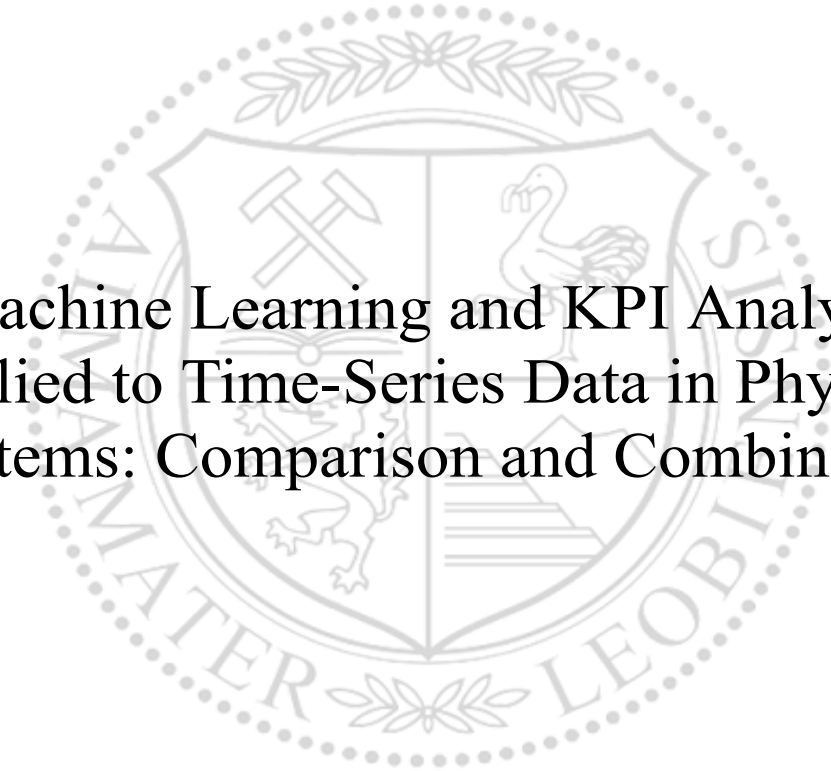
Chair of Automation

Master's Thesis



Machine Learning and KPI Analysis applied to Time-Series Data in Physical Systems: Comparison and Combination

Maria Elisabeth Haider, BSc

May 2021

MONTANUNIVERSITÄT LEOBEN
www.unileoben.ac.at

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich die Richtlinien des Senats der Montanuniversität Leoben zu "Gute wissenschaftliche Praxis" gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 26.05.2021

Unterschrift Verfasser/in
Maria Elisabeth Haider

Acknowledgements

First of all, I would like to thank Professor Paul O'Leary for his great support in supervising this thesis. I am deeply grateful for not only what he has taught me in the context of my studies, but also for his inspiring excursions on other topics. He established an encouraging and motivating environment to work in, especially in these tiring times.

In addition, I would like to express my gratitude towards the people of the Chair of Automation. Everyone was very welcoming, supportive and happy to help, which made working at the chair an enriching experience.

Furthermore, I would like to thank my family for their love and financial support throughout my studies. I also want to thank my sister and friends for proof-reading my thesis, their advice and their emotional support. Without them, I would not be where I am today and I am very thankful for having them in my life.

Kurzfassung

Diese Arbeit untersucht die Kombination des maschinellen Lernens und der statistischen Datenanalyse. Das Ziel ist die Konzeption einer nicht überwachten Erkennung von Ausreißern in Echtzeit-Zeitreihendaten, die von Sensoren und Aktoren stammen. Es wird untersucht, ob und wann, die zusätzliche Anwendung von maschinellem Lernen klassischen Methoden vorzuziehen ist und wo seine Stärken und Grenzen liegen. Um eine gründliche und genaue Identifizierung von Anomalien zu ermöglichen, wird ein hybrider Ansatz untersucht. Seine Komponenten basieren auf früheren Arbeiten des Lehrstuhls für Automatisierungstechnik an der Montanuniversität Leoben, welche die Segmentierung verschiedener Phasen des industriellen Prozesses sowie die Definition von Schlüsselindikatoren (KPIs) durch Einbringen von physikalischem Vorwissen und Erfahrung umfassen. Das hybride Modell enthält die Berechnung des Interquartilbereichs zur Definition von Ausreißern sowie das Trainieren und Testen eines maschinellen Lern-Algorithmus mit optimierten Hyperparametern. Der Ansatz wurde auf Maschinendaten angewandt, die vom Rüttelstopfverfahren zum Verankern von Pfählen im Fundament von Gebäuden stammen.

Abstract

This thesis examines the combination of deep learning and statistical data analysis for the unsupervised detection of outliers in unlabelled real-time time-series data originating from sensors and actuators. It is investigated if and when the additional application of machine learning is preferable to classical methods, where its strengths and shortcomings lie. A hybrid approach is introduced to enable an exhaustive and precise identification of anomalies. Its components are based on previous work performed by the Chair of Automation at the University of Leoben, which includes the segmentation of different phases in the industrial process and the definition of key point indicators (KPIs) by accessing physical knowledge and experience. The hybrid model contains the calculation of the interquartile range to define outliers, as well as the training and testing of a hyperparameter-optimized deep learning algorithm. The approach was applied to datasets from machinery used in the construction of anchoring piles for the foundations of buildings.

Contents

Chapter 1
Introduction

This thesis investigates how the combination of traditional statistical data analysis and state-of-the-art deep learning algorithms improves the unsupervised detection of anomalies in unlabeled time-series data acquired from real physcial systems.

Due to increasing advances in hardware and software technology for data acquisation and organization the ongoing collection of data is standard in almost all industrial fields. The goal is to gain a better understanding of the current state or generate future state predictions in order to optimize processes. However, the sheer accumulation of data does not offer any potential in improving the production process unless it is analyzed and meaningful information is extracted. This can be done by various different data analytics methods.
By prospecting for anomalous behaviour in the data, inefficencies and errors in the process can be identified, which contributes to a more efficient use of energy and ressources and helps to decrease possible health and safety risks for workers as well as customers. For instance, undetected outliers in the construction of anchoring piles for the foundation of a building can result in a dangerous loss of stability of the building. There are many different definitions for the concept of anomalies and outliers, which both refer to the same entity in this work. A widely used definition for the term is given in [1], where an outlier is described as ”an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism”. Hence, an outlier is an observation that does not follow the expected behaviour of the system.

Outlier detection is a challenging task. The results are highly dependent on the available data, the type of outlier and the used detection method, which should eventually deliver correct, exhaustive and interpretable results.
Furthermore, sensor data in manufacturing is collected over time and thus features a temporal component. As a result, the sequence of the observations is of importance. This is especially relevant for the application of machine learning algorithms, as not all models factor sequence.
In addition, due to several sensors observing the same process, the data is multivariate and contains multiple, possibly correlated, dimensions. It is required that the used method is able to find depen-

dencies and correlations within the different features of the data and to operate with reasonable time and computational effort [2].

In literature the term hybrid model often refers to deep learning models that combine different network architectures, as it is the case in [3] and [4]. In [5], the combination of an artificial neural network (ANN) and a finite state machine is described as a hybrid machine learning approach. Based on [6], in this thesis the term hybrid models is used for models that combine statistical and deep learning techniques in order to improve the overall performance of the model. Machine learning is expected to close a gap in the identification of anomalies that are not statistically noticeable. The task of anomaly detection has been sudied extensively in the context of time-series data. In [7], anomalies in time-series data are detected by using a long short-term memory based variational autoencoder (LSTM-VAE). Although an unsupervised approach, the network was trained and the model parameters optimized with data that contains no anomalies. The outliers are detected by summing up the prediction errors to define a score function. The threshold on the score function is either defined with the help of a validation set containing normal data and outliers and calculating the f1-score or by a given percentile of the distribution of the score function when a validation set is used only containing normal samples. In [8], a LSTM-VAE is used to detect outliers in the context of robotics. It detects an anomaly when the log-likelihood of an observation given the expected distribution is lower than a state-based threshold. The training and validation datasets are non-anomalous.
Both, [7] and [8], have non-anomalous training and validation datasets available. This, however, is not the case for data acquired from a real physical system. Anomalous data affects both, traditional analysis ([9]) as well as the training of a neural network. Manually labelling the data does not ensure that the training set is entirely free of outliers and is very time-consuming.

The main contribution of this work is to investigate the differences between machine learning and traditional statistics for outlier detection and to show how both complement each other in this context.

1.1 Motivation

When shown a data plot, humans are able to intuitively recognize the underlying structure of the data points and detect outliers solely by looking at the patterns. This is also the case for anomalies that are not discovered by statistical methods. Where this intuition derives from is a scientific field on its own and is not going to be further investigated in this thesis. However, it is a valid idea to take a look at how nature does it when it comes to the search for new approaches for anomaly

detection in order to close the detection gap of traditional statistical methods.

Modern science and philosophy presume that the human brain works as a network containing the neurons acting as processing units. According to this theory, Artificial Neural Networks try to approximate the complex synaptic computations that are conducted by the brain's network. [10] Consequently, the application of machine learning for the detection of outliers is expected to deliver new and different results when compared to traditional methods. Furthermore, the combination of both methods could achieve better results and lead to a new differentiation between types and heaviness of outliers, which is a topic not explicitely covered by current literature.

1.2 Goals

The combination of traditional descriptive statistics of KPI with deep neural networks for outlier detection shall be evaluated. This is executed by building and training a hybrid machine learning model that includes statistical methods as a preprocessing step as well as for the evaluation of the network's output and comparing the results to those obtained by the sole statistical apprach. The primary goal is to determine if and when machine learning is preferable to classical methods. In addition, the anomaly detection on multivariate time-series data shall be improved by combining both methods and taking statistical results as well as outliers detected by machine learning into account. The following shall be realised:

1. An overview of the strengths and shortcomings of neural networks in the context of outlier detection in time-series data in contrast to classical statistics.

2. A hybrid deep learning model based on the previous work of [11] and [12] to automatically detect anomalies in multivariate time series data with a preferably high accuracy.

3. An evaluation of the differences of outliers detected by deep learning compared to statstically identified outliers.

1.3 Organization

This work is structured in the following manner: Part 2 presents common statistical outlier detection methods. Part 3 provides a fundamental overview about machine learning and common algorithms involved in learning while Part 4 builds up the theoretical background behind the neural network used in the hybrid model. Part 5 reflects on the practical part of building a hybrid anomaly

detection model. Part 6 interprets the results of said model and proposes possible directions for future work.

Chapter 2
Statistical Data Analysis for Outlier Detection

This chapter gives an overview of different statistical analysis methods for the detection of outliers. According to [13], there are two types of outputs for an outlier detection algorithm:

1. Outlier Score: An outlier score quantifies the level of ”outlierness” of each data point. It can also rank data points in the order of their outlier tendency. It does not yield a specific itemisation of those data points that are considered outliers.
2. Binary Labels: Binary labels specifically indicate whether or not a data point is an outlier. Usually thresholds are applied on outlier scores in order to attain a binary labelled output. This type of output contains less information, but is typically the final result needed for anomaly detection in practical applications.

The detection of outliers via machine learning algorithms delivers an outlier score. In order to clearly define whether or not a data point is an outlier, binary labels are necessary. Thus, it is essential to apply statistical data analysis on the output of the machine learning model.

2.1 Input Data Types

According to [13], most works on outlier detection focus on multidimensional numerical data and assume that the single data records are independent of one another. This means, that the order of the data points is not relevant. Examples for data types with discrete, unordered values include categorical, text or attribute data. Common outlier detection methods for text and categorical data are clustering and proximity-based methods as well as probabilistic models. When different data values in a dataset are linked temporally or spacially, the data values are dependent on one another [13].

Temporally dependent data is also called time-series data. In [14], a time series is defined as a set of observations collected sequentially in time. Each observation x_t is recorded at a specific time t. According to [15], a time series is called discrete-time time series, if the set T_0 of times at which observations are made is a discrete set. This is the case when observations are made at fixed time

intervals. If observations are recorded continuously over a time interval the time series is called a continuous-time time series.

A time series containing a finite amount of successive observations can be regarded as a sample from an infinite population of such time series generated by a stochastic process. The underlying process can be stationary or nonstationary. In a stationary process the mean and the variance of the time-series do not change over time. In a nonstationary process different segments of the process yield different values for mean and variance [16] [14].

According to [2], time-series data can be distinguished between univariate and multivariate time-series data. A univariate time series has one dimension that is dependent of the time value. A multivariate time series contains several dimensions. One dimension of a multivariate time series is a univariate time series and each variable is not only dependent on its past but also on the other time-dependent variables. In time-series data the expected values of data points are influenced by their dependencies. [13].

2.2 Types of Outliers in Time-Series Data

Outliers are unusual observations, that significantly differ from the other data points and affect the analysis of the data. In this work outliers in the system's behaviour are detected via outliers in the data. There are numerous ways to define and differentiate between types of outliers. The following descriptions are drawn from [14], [2], [17] and [13]:

1. Additive Outliers: An additive outlier is a measurment error caused by external factors, like a machine breakdown or human error for instance. It does not affect the underlying process.
2. Innovative Outliers: An innovative outlier is caused by a change in the process or system and thus indicates that the process is affected.
3. Level Shifts: Level shifts imply a permanent change in the process mean, resulting in a change from a stationary to a non-stationary process.
4. Point Outliers: A point outlier behaves different in a specific time instant when compared either to other values in the time series or to its neighboring points. In the first case it would represent a global outlier, while in the latter it would be described as a local outlier.
5. Subsequence Outliers: Also subsequence outliers can be either local or global and refer to consecutive points in time whose joint behaviour is unusual. The individual points in a subsequence outlier are not necessarily point outliers. Either one or more dimensions can be affected, thus there are univariate and multivariate subsequence outliers.
6. Outlier Time Series: In a multivariate time series an entire dimension, which is a time series on its own, can be an outlier.

2.3 Descriptive Statistics

This section introduces univariate descriptive statistics for outlier detection. Statistical approaches for outlier detection assume a distribution or probability model to fit a given dataset. They can be divided into parametric methods and non-parametric methods. Parametric methods estimate the parameters of the distribution model from the given data. Non-parametric methods do not assume any distribution characteristics. This work investigates only parametric methods and leaves non-parametric methods aside.

In descriptive statistics characteristic values of past data are calculated in order to specify the properties of the dataset. Univariate statistics only considers a single variable, also if the data has multiple dimensions. Descriptive statistics can, among other things, be divided into measures of central tendency or location statistics and measures of variation. Two of the most commonly used measures of tendency, namely arithmetic mean and median, as well as variance as a measure of variation are shortly explained in the following. Then three broadly used statistical thresholding methods are presented: standard deviation, interquartile range method and median absolute deviation[18][2][17].

Arithmetic Mean

The arithmetic mean or sample mean is the most broadly known measure for the central location of a dataset. For observations $x_1, x_2, ..., x_n$ the arithmetic mean of samples $\bar{x}$ is defined as

$$\bar{x} = \frac{1}{n}(x_1 + x_2 + ... + x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2.1}$$

The arithmetic mean for grouped data is

$$\bar{x} = \frac{1}{n}(f_1 x_1 + f_2 x_2 + ... + f_n x_n) = \frac{1}{n}\sum_{i=1}^{n} f_i x_i, \tag{2.2}$$

whereat x_i is the center of the i-th class and f_i is the absolute frequency of the i-th class, n is the number of observations and k is the number of classes. For a finite population of N elements, the arithmetic mean of the population μ is calculated as

$$\mu = \frac{1}{n}\sum_{i=1}^{N} x_i. \tag{2.3}$$

The mean of the population μ is usually unknown and must be estimated using the sample mean and hypothesis testing [18].

Median

The median of a set of observations is the middle element of the ordered dataset. In case of an uneven dataset, the median is the $\frac{n+1}{2}$-th element or the object value ranked in the middle of the maximum and the minimum object values. If the dataset is even, it is the mean of the $\frac{n}{2}$-th and the $\left(\frac{n+1}{2}+1\right)$-th element or the average of the two central objects values. In contrast to the arithmetic mean the median is not affected by extremely large or extremely small object values [18] [9].

Variance

The variance is a measure of spread. The sample variance s^2 is the difference between the sample and the arithmetic mean $\bar{x}$ divided by $(n-1)$:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{N}(x_i - \bar{x})^2. \tag{2.4}$$

The population or process variance of all N elements of a population is calculated as

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu^2)}{N}. \tag{2.5}$$

The sample and the population variance are similar for large samples.

2.3.1 Standard Deviation

The standard deviation (SD) thresholding technique assumes that the distribution of the data is normal and that the outlier level is 0.13%. The abbreviation *SD* can be either the sample standard deviation s or the population standard deviation σ. Both are defined as the (positive) square root of the sample or population variance, respectively:

$$s = \sqrt{s^2} \tag{2.6}$$

$$\sigma = \sqrt{\sigma^2} \tag{2.7}$$

The minimum and maximum thresholds T_{min} and T_{max} are calculated as

$$T_{min} = \sigma - a\,SD \tag{2.8}$$

$$T_{max} = \sigma + a\,SD \tag{2.9}$$

whereat a is a manually set control parameter. A smaller value results in more objects marked as outliers. As the amount of outliers is expected to be small, $a = 3$ is the most common choice.

Thus, an object is declared an outlier if its value is higher or lower than the mean $\pm$ three times SD. Problems involved with this technique are, that the data has to be normally distributed, that outliers strongly affect the mean and that the data can also contain far more outliers than 0.13% [9].

2.3.2 Interquartile Range Method

The interquartile range method (IQR) calculates the difference between the values in 25% and 75% of the data set. The former is denoted as Q_{25}, while the latter is called Q_{75}. The thresholds are calculated as

$$IQR = Q_{75} - Q_{25} \tag{2.10}$$

$$T_{min} = Q_{25} - c\ IQR \tag{2.11}$$

$$T_{max} = Q_{75} + c\ IQR \tag{2.12}$$

where c is a value usually set to 1.5. When Q_{75} is located within the outliers, this method is also affected by them [9].

2.3.3 Median Absolute Deviation

The median absolute deviation (MAD) is a more robust method, as the median M is not as heavily affected by outliers as the mean. The MAD is calculated as follows [9].

$$MAD = bM(|X - M(X)|) \tag{2.13}$$

$$T_{min} = M(X) - a\ MAD \tag{2.14}$$

$$T_{max} = M(X) + a\ MAD \tag{2.15}$$

Usually data points that are more than three scaled MAD distant from the mean are considered as outliers and should be detected. Thus the factor a is set to 3. The factor b is defined as

$$b = \frac{-1}{(\sqrt{2}\ \varepsilon\ (\frac{3}{2}))} = 1.4826 \tag{2.16}$$

whereat ε is the inverse complementary error function [19].

2.3.4 Limitations in Statistical Thresholding

The advantage of statistical outlier detection methods is that they are mathematically justified and deliver interpretable results [17]. However, especially parametric methods are subject to some limitations. According to [9], outliers in the data affect statistical methods, yet most time-series data obtained in a real physical process contain outliers. As a consequence, anomaly thresholds tend to be set up either too high or too low and outliers are missed. Standard deviation and interquartile range are both based on the mean, which is heavily influenced by anomalies in the data. The median absolute deviation is a more robust threshold, because the median is quite independent of outliers. Yet, if the rate of outliers exceeds 50%, also MAD is affected by anomalous data. This is shown in Figure 2.1.

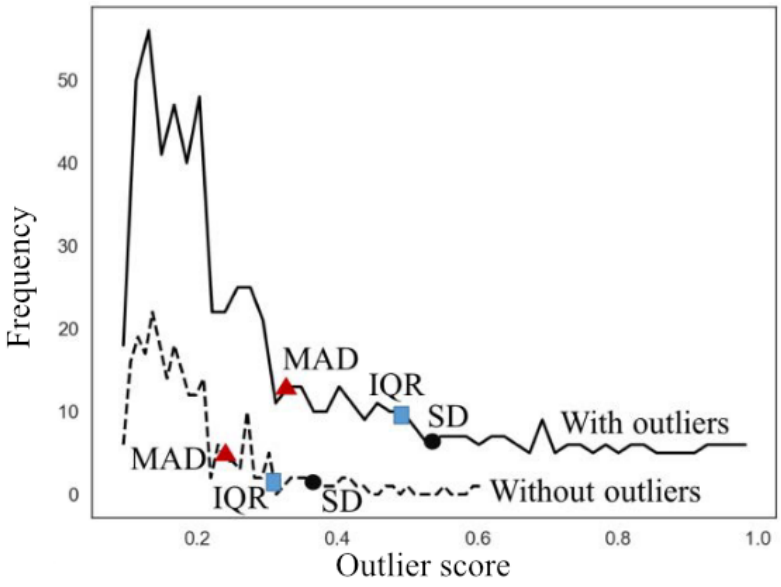


Fig. 2.1: The effect of outliers on three thresholding techniques, MAD, IQR and SD is shown. This figure is taken from [9].

Furthermore, distribution-based methods require prior knowledge about the underlying dataset. As the results are largely dependent on the distribution chosen to fit the data, their quality is hard to evaluate. A single distribution may not model the entire data because it could derive from multiple distributions. This is another limitation to statistical outlier detection methods: they apply to an univariate feature space. Thus, they are typically not applied to multivariate datasets and if they are, they treat each dimension as independent of the others. Basic statistical outlier detection techniques are incapable of finding those correlations between the different dimensions of multivariate data, that could be crucial in detecting certain anomalies [17].

Chapter 3
Machine Learning

In machine learning a computer program or algorithm statisically estimates the functions that describe the underlying structures, regularities and correlations hidden in the data. Consequently, it can compute a model of the system and make predictions, generate new data or find anomalies based on this model.
While the term Machine Learning became popular rather recently, it actually is quite an old scientific field, rooted in statistics, computer science and other disciplines that process data, develop data-adaptive models and make predicitons.
The process of learning from data is called training. The algorithm can not learn anything that is not given in the data. Once the machine learning model is implemented it can complete a task much faster than humans, especially when confronted with a huge amount of data. There are two major different ways of training a machine learning model, supervised and unsupervised methods. Furthermore, hybrid learning combines several different machine learning models or a machine learning model and a statistical model in order to complete a task [20][21][22].

3.1 Gradient-Based Optimization

The learning process of a machine learning algorithm is basically an optimization problem. The goal is to minimize a cost or loss function. One of the most broadly used methods for the iterative minimization of a differentiable cost function is the method of gradient descent. There are several different gradient descent based optimization algorithms. In the following the fundamentals for each gradient-based optimization method is briefly explained. In principle, each gradient descent method involves three steps:

1. Search Direction: Calculation of the derivative to decide on a direction. The derivative of a multi dimensional function is called the gradient.
2. Step size: The step size can be either constant or changing with each iteration and controls how ”far” the current point moves into the search direction before the next derivative is calculated.

3. Convergence Step: Reaching the minimum of the function is called convergence. Until convergence the first two steps are iterated and updated.

May the cost function be $J(\boldsymbol{\theta}), \boldsymbol{\theta} \in \mathbb{R}$. The gradient descent method starts from an initial estimate $\boldsymbol{\theta}^{(0)}$ and generates a sequence $\boldsymbol{\theta}^{(i)}, i = 1, 2, \ldots$ such that

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \mu_i \Delta\boldsymbol{\theta}^{(i)}, \tag{3.1}$$

where $i > 0$ and $\mu_i > 0$. The sequence μ_i is the step size or learning rate at the ith iteration. The values of μ_i are either constant or change at each iteration. The vector $\Delta\boldsymbol{\theta}^{(i)}$ is the update or search direction. The choice of $\Delta\boldsymbol{\theta}^{(i)}$ is made that

$$J(\boldsymbol{\theta}^{(i)}) < J(\boldsymbol{\theta}^{(i-1)}). \tag{3.2}$$

The steepest descent direction coincides with the negative gradient descent direction. The corresponding update recursion is the gradient descent scheme

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \mu_i \nabla J(\boldsymbol{\theta}^{(i-1)}). \tag{3.3}$$

Because of the assumption that $J(\boldsymbol{\theta})$ is differentiable, the contours of the function must be smooth and accept a unique tanget plane at any point. Thus, gradient descent is limited to continuous spaces. In Figure 3.1 an example for a cost function in a two-dimensional parameter space is given [20][22].

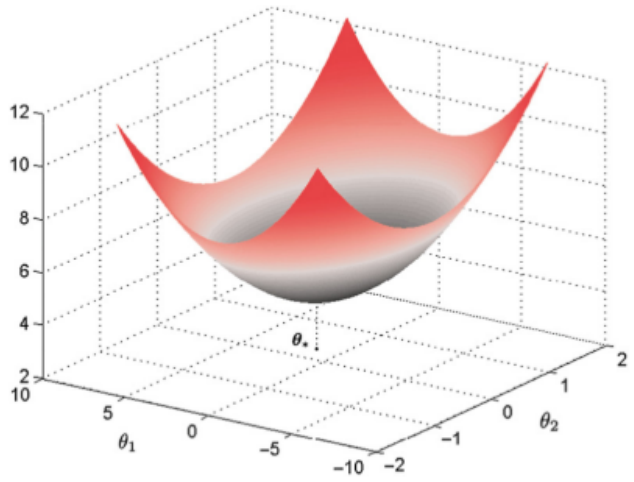


Fig. 3.1: Example of a cost function in a two-dimensional parameter space. At θ_0 both parameters of the cost function reach their minimum. The initial estimate $\theta^{(0)}$ is somewhere on the surface and approximated to θ_0 via gradient descent. This graph is taken from [20].

3.2 Supervised Learning

In supervised learning methods the available data is already labeled and provided as examples that deliver known correspondences between input and expected outcome. Both, input and output data are in vector form, for instance $(\boldsymbol{y}_n, \boldsymbol{x}_n)$ with $n = 1,2,...,N$. The variables in $\boldsymbol{x}_n$ are the input variables or features, while the variables in $\boldsymbol{y}_n$ are known as the output or label variables. In the learning process a function is estimated to predict the value of the respective output, if the input value is given. The classification and the regression tasks are the basis of supervised learning. The main difference is that the output of regression is a probability, while the output of classification is a class identity [20][23][22].

3.2.1 Regression

Linear regression has long been used in statistics to tackle quantitive problems. The output variable is not discrete, but provides probabilities as values in an interval. The regression task is a function fitting problem and can be used to model the dependence of the output or regression target y on the input or features x. In addition, it can predict the output y of a new sample x, which is outside of the training set. This is depicted in Figure 3.2. The task is to find a function f, whose graph

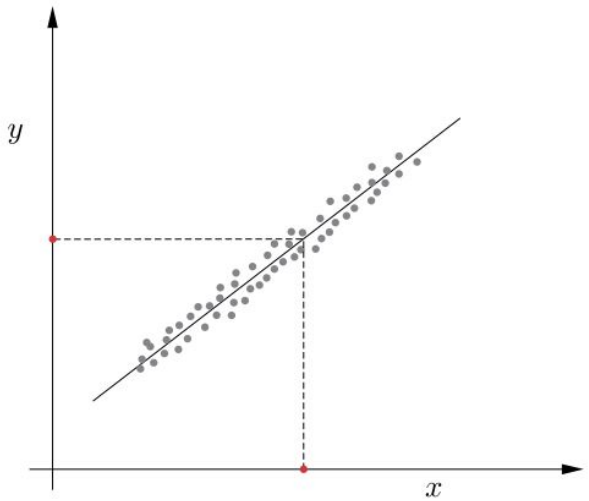


Fig. 3.2: In linear regression the coefficients or weights of the function f are designed to fit the given training data set, depicted by the gray points. Once the function fitting task has been completed the output value $\hat{y} = f(x)$, which is the red point on the y-axis can be predicted when an input outside of the training data set (red point on the x-axis) is given. In this simple example, which is taken from [20], the graph is a straight line.

fits a given set of training samples $(y_n, x_n), y_n \in \mathbb{R}, x_n \in \mathbb{R}^l, n = 1,2,...,N$. As the name already

reveals, in linear regression the learned relationships between input and output of the training data set are linear. Therefore, they can be written as in Equation 3.4. Due to the linearity, interpretating the results is relatively easy, which is why linear regression and similar models are widespread in different academic fields.

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \varepsilon \tag{3.4}$$

While the coefficients β_j of the linear equation are the learned feature weights, the weighted sum of the p features represents the predicted output of an instance. The first weight β_0 is not multiplied with a feature and called the intercept. The error between the predicted and the actual output, calculated as the difference of the two values, is denoted by ε. It is assumed, that the errors follow a Gaussian distribution, thus errors are made in both negative and positive directions and there are rather many small and just few large errors.

To improve the computed relationship between input and output as well as the prediction of the output the error function first has to be defined and then minimized. The goal is to estimate the optimal weights. Various methods exist for doing so, usually the least squared error method is used. It minimizes the squared differences between the estimated and the actual outcome and is given in equation 3.5.

$$\hat{\beta} = arg \min_{\beta_0,...,\beta_p} \sum_{i=1}^{n} \left(y^{(i)} - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j^{(i)} \right) \right)^2 \tag{3.5}$$

The estimated weights come with confidence intervals, for instance of 95%. This would mean that, given the linear regression model is correct for the data, the confidence interval includes the true weights in 95 out of 100 cases. [20][21][23]

3.2.2 Classification

Classification aims at designing a pattern for one class out of a set of possible classes, whereat the number of classes is known. A very broadly used approach for classificaiton are support vector machines, abbreviated to SVM. In Figure 3.3 an exemplary linear classifier is depicted, which separates the training data into two classes [20].

As they are described by a linear function $\boldsymbol{w}^\top \boldsymbol{x} + b$, support vector machines are somehow similar to logistic regression. However, instead of delivering probabilities as an output, the support vector machine provides a class identity. A linear kernel SVM segregates two classes of points by a margin, which is a region bounded by two parallel hyperplanes. No point should be in the margin region between the seperating hyperplanes. If this was the case, it would not be clearly classified. Those data points that are lying on the margin boundaries are called support vectors, while those lying on either side of the hyperplanes belong to one of the two classes. Certain points can be falsely classified, as can be seen in the red point in figure 3.3. The two classes are marked with $+1$

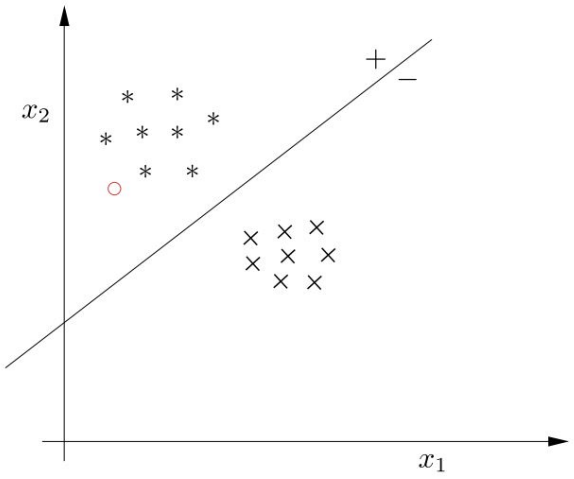


Fig. 3.3: A linear classifier divides the training data into two classes. The straight line is a linear function. On its positive side lie the points of one class and on its negative side the points of the other. The red point is an example for a point of an unknown class which is classified in the same class as the star-shaped points, because it is lying on the positive side of the graph. This figure is taken from [20].

and -1 labels. The vector w is determined such that the prediction states a positive class is present when $\boldsymbol{w}^\top \boldsymbol{x} + b$ is positive and a negative class is present when $\boldsymbol{w}^\top \boldsymbol{x} + b$ is negative [22][23].

3.3 Unsupervised Learning

In contrast to supervised learning, where input and output variables are provided in the training data, in unsupervised learning only the input variables are known. Furthermore, no label information is available, which usually is the case for most data in machine learning, as labeling is a costly process and has to be done manually. Consequently, unsupervised learning extracts information from a sample without requiring human labor. The goal is to find the underlying struture of the given training data set.
There are different types of applications for unsupervised learning, for instance clustering and dimensionality reduction. Other possible applications for unsupervised learning are probability distribution estimation and data generation [20][23][22].

3.3.1 Clustering

One of the most important unsupervised learning methods is clustering. It aims to find a group struture within the points in a data set. The required algorithm is based on the concept of similiar-

ity. This means it is assumed that patterns that belong to one cluster are more similar than patterns from a different cluster. There are various clustering algorithms, for instance k-means clustering, density-based clustering and matrix factorization, which is conducted by singular value decomposition (SVD). Principal component analysis (PCA), which is explained in Section 3.3.2, can be used as well [20][23][22].

k-means Clustering

K-means clustering is a simple representation learning algorithm, that divides the training set into k different clusters of examples that are near to each other. The algorithm provides a k-dimensional vector $\boldsymbol{h}$ respresenting an input $\boldsymbol{x}$. The entry $\boldsymbol{h_i} = 1$, if $\boldsymbol{x}$ belongs to cluster i, while all other entries of $\boldsymbol{h}$ are zero. The vector $\boldsymbol{h}$ is an example for a sparse representation, because most of its entries are zero. The k-means algorithm first initializes k different centroids $\{\boldsymbol{\mu}^{(1)}, ..., \boldsymbol{\mu}^{(k)}\}$ to different values. Then the training examples are assigned to the cluster i, where i is the index of the nearest centroid $\boldsymbol{\mu}^{(i)}$. Each centroid $\boldsymbol{\mu}^{(1)}$ is updated to the mean of all training examples $\boldsymbol{x}^{(j)}$ assigned to cluster i. It alternates between these two steps until convergence [22].

Singular Value Decomposition

Singular Value Decomposition (SVD) is a starting point for dimensionality reduction, which is discussed in section 3.3.2. It provides a way to factorise a matrix into singular vectors and singular values. It enables the discovery of the same kind of information as the eigendecomposition reveals, but is more generally applicable. For instance, if the matrix is not square, the eigendecomposition is not defined and SVD must be used instead.
The eigendecomposition analyses a matrix $\boldsymbol{A}$ to discover a matrix $\boldsymbol{V}$ of eigenvectors and a vector of eigenvalues $\boldsymbol{\lambda}$ which can be written as

$$\boldsymbol{A} = \boldsymbol{V} diag(\boldsymbol{\lambda}) \boldsymbol{V}^{-1}. \tag{3.6}$$

In singular value decomposition the matrix $\boldsymbol{A}$ is written as the product of three matrices

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\top}. \tag{3.7}$$

Supposing that $\boldsymbol{A}$ is an $m \times n$ matrix, then the dimensions of $\boldsymbol{U}$ are $m \times m$, $\boldsymbol{D}$ is defined to be an $m \times n$ matrix and the dimensions of $\boldsymbol{V}$ are $n \times n$. The matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are defined to be orthogonal matrices. $\boldsymbol{D}$ is a diagonal matrix and not necessarily square. The elements along the diagonal of $\boldsymbol{D}$ are the singular values of the matrix $\boldsymbol{A}$. The columns of $\boldsymbol{U}$ are called the left-singular vectors of $\boldsymbol{A}$ and are the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^{\top}$. The columns of $\boldsymbol{V}$ are known as the right-singular vectors and the eigenvetors of $\boldsymbol{A}^{\top}\boldsymbol{A}$. The nonzero singular values of $\boldsymbol{A}$ are the square roots of the eigenvalues

of $\boldsymbol{A}^\top \boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{A}^\top$. A very useful feature of SVD is, that it can be used to partially generalise matrix inversion to nonsquare matrices [22].

3.3.2 Dimensionality Reduction

Unsupervised learning can be used for dimensionality reduction, which is important in machine learning to compress representations or reduce computational effort, which is especially of importance in the context of big data processing and analysis. Although a data set is high-dimensional the true dimensionality, called intrinsic dimensionality can be of a much lower value, for instance because a number of components is zero. The underlying assumption for dimensionality reduction is that a system or process is driven by a relatively small number of latent (not directly observed) variables, which generate the data. The objective is to learn the latent structure. In any dimensionality reduction technique some original information is going to be lost. Methods proposed for dimensionality reduction are autoencoders, which are going to be explained in Section 4.3.4, principal component analysis (PCA) and canonical correlation analysis (CCA) [20][23][22].

Principal Component Analysis

Principal Component Analysis, abbreviated to PCA, can be used for two tasks: compression and representation of data. The representation of data learned by PCA has a lower dimension as the input and its elements have no linear correlation with each other. To achieve a statistically independent representation, the learning algorithm must remove the nonlinear relationships between the variables. In PCA a linear and orthogonal transformation of the data that projects an input $\boldsymbol{x}$ to a representation $\boldsymbol{z}$, as depicted in Figure 3.4 is learned [20][22].

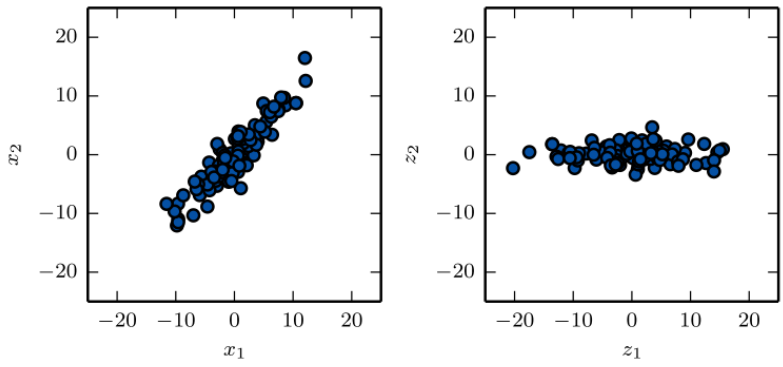


Fig. 3.4: The linear projection learned by PCA aligns the direction of greatest variance with the axes of the new space. On the left site the original data $\boldsymbol{x}$ is shown. On the right side the transformed data $\boldsymbol{z} = \boldsymbol{x}^\top \boldsymbol{W}$ varies most around the z_1-axis and the direction of the second-most variance is along z_2. This depiction is taken from [22].

Canonical Correlation Analysis

While PCA focuses on a single data set, Canonical Correlation Analysis (CCA) can process two data sets jointly. The goal of CCA is to find a pair of linear transformations for each set of variables, respectively, such that the resulting transformed variables are maximally correlated [20].

3.4 Semi-supervised Learning

Semi-supervised learning lies somewhere inbetween supervised and unsupervised learning. The data is insufficiently labeled. This means some information is provided, but not for all samples. Thus, both unlabeled examples from $P(x)$ as well as labeled examples from $P(x, y)$ are used for the estimation of $P(y|x)$ or the prediction of y based on a given x. The data can be divided into two parts: the points for which labels are provided and the points where the labels are unknown. For instance, this can be realized by using principal components analysis as a preprocessing step before a classifier is applied [20][22][24].

3.5 Hybrid Learning

While in [3] hybrid learning is defined as the combination of clustering and classification, or rather unsupervised and supervised learning methods, this work defines hybrid learning based on [6] as the combination of two models which differ in their transparacy and the interpretability of the results they deliver. One model, the data-driven model, is trained by a machine learning algorithm and thus behaves like a black-box. The other, knowledge-based model is transparent and thus behaves like a white-box. It delivers more interpretable results. Hybrid modelling is a way of incorporating knowledge about a task into the computation of a solution to receive more accurate results. There are different types of hybrid modelling, parallel hybrid and serial hybrid [6].

3.5.1 Parallel Hybrid

In a parallel hybrid model the input data is fed into the two models independently. Both models are delivering a result which is eventually combined to one final one. There are two categories of parallel hybrids. In the first category the black-box estimates the residual ε of a white-box, which can be added to the result $\hat{Y}_{WB}$ of the white-box as a correction. This type of a parallel hybrid is very similar to a serial hybrid, which is going to be explained in Section 3.5.2.
In the second category of parallel hybrids both models are standalone parts. This can make so-

phisticated ways of combining the different results necessary. The weighting of the results can be computed in a seperate black-box [6].

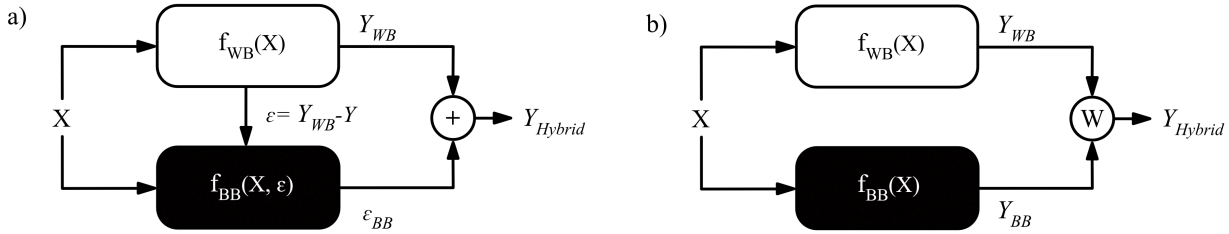


Fig. 3.5: The two categories of parallel hybrid models are *a)* used for error correction and *b)* weighting two standalone predictions.

3.5.2 Serial Hybrid

In a serial hybrid the two models are arranged in series. Serial hybrids differ in the sequential arrangement of each approach.
If the knowledge-based model is the preceding one, certain input-output relations are enforced by passing on a coefficient to the data-driven model. Consequently, certain variable relationships are not changed by the data-driven model. Successive data-driven models can also be applied to compute the error between the output of the knowledge-based model and the ground truth to subsequently perform an error correction. This is similar to the first category in Section 3.5.1. In the case of the data-driven model preceding the goal is to compute particular coefficients for the knowledge-based model that are otherwise not known a priori [6].

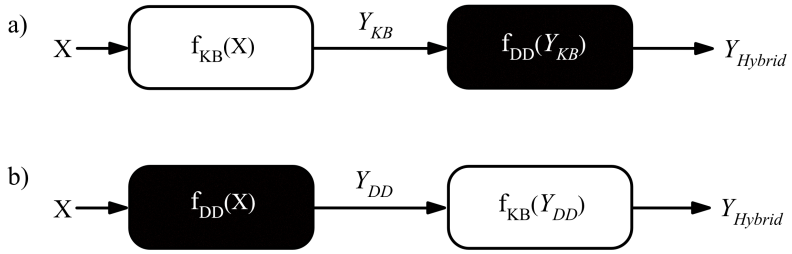


Fig. 3.6: The two different ways of serial hybrid models are *a)* the ones that feed the data into a knowledge-based model first and *b)* the ones that feed data into the data-driven model.

3.6 Challenges of Machine Learning

A central challenge in machine learning is that the algorithm must be able to perform well on new and previously unobserved inputs. This ability is called generalization. In order to judge the systems ability to generalize, the training and the test dataset have to be completely different portions of the dataset.
The goal during training a machine learning model is to reduce its error measure on the training set, the training error. This is a simple optimization problem. However, the generalization error, which is the expected value of the error of a new input, is expected to be minimized as well. Assuming that the training and the test set are identically distributed, the generalization error, also called test error, can be estimated by measuring the models performance on a test set of examples. The expected training set error is equal or greater than the test set error. The two main tasks of a machine learning algorithm are

1. minimizing the training error and
2. minimizing the gap between the training and the test error.

These two factors represent the two central challenges in machine learning: underfitting and overfitting. When the model is not able to obtain a sufficiently low error value on the training set underfitting occurs. Overfitting is the result of a too large gap between the training and the test error.
A model's ability to fit a wide range of functions is called its capacity. By altering the capacity of a model, the model's likeliness to underfit or overfit can be controlled. Low capacity may lead to underfitting and high capacity to overfitting. Machine learning algorithms will perform best when the capacity is appropiate to the complexity of the task. In Figure 3.7 the relationship between the training error, the generalization error, the capacity, underfitting and overfitting is shown [22].

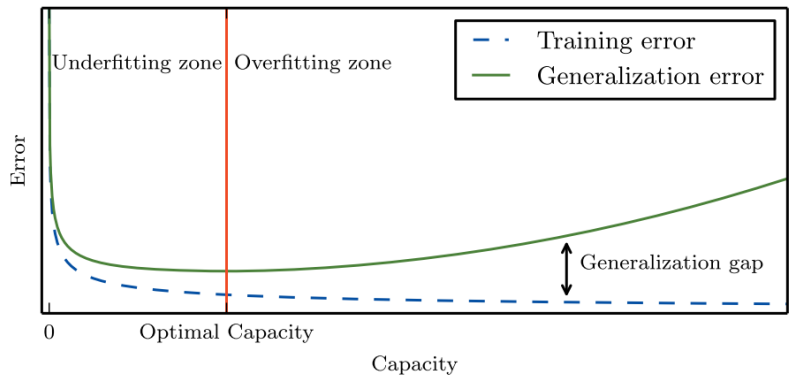


Fig. 3.7: As depicted in this graph from [22], the optimal capacity is the border between underfitting and overfitting. A low capacity leads to underfitting and a high training and test error. If the capacity increases, the generalization gap, which is the difference between the training error and the test error, increases as well. In an overfitted model the generalization error is large, while the training error slightly decreases. This results in a large generalization gap.

Chapter 4
Artificial Neural Networks

Neural networks are learning machines that comprise of a large number of neurons that are connected in a layered fashion. In the learning process synaptic weights are adjusted in order to minimize a preselected cost function [20].
The following chapter reflects on the origin of artificial neural networks, explains backpropagation and hyperparameters as well as investigates different network architectures.

4.1 The Perceptron

Very often in history, scientists have been inspired by nature. Thus, it is not surprising that artificial intelligence, of which machine learning is a part of, is inspired by the study of the human (or in general mammals) brain. This central organ consists of 60 to 100 billions of basic building elements, known as neurons. Those neurons are connected with each other by 50 to 100 trillions so-called synapses [20].
In the 1950s Frank Rosenblatt developed an artificial neuron called perceptron, which is a learning machine based on the concept of neuronal models. It consists of a single neuron that is able to learn from labeled training data and can either be used for classification or regression tasks. In addition, he developed the perceptron algorithm for the respective training. This was the first step towards machine learning as it is known today.
Similar to the biological neuron, information in an artificial neural network is weighted before it is sent to the next cell. Furthermore, the weighted sum passes through an activation function, which the output is proportional to. There are several activation functions, such as the sigmoid or logistic function. In a multilayer network the output would serve as an input signal for the neurons of the next layer [20] [25] [26].

According to [26], the output of an artificial neuron can be described by equation 4.1

$$\hat{y} = \phi\left(\sum_{i=1}^{n} \boldsymbol{\theta_i} \boldsymbol{x_i} + b\right) \tag{4.1}$$

whereat ϕ is the activation function, $\boldsymbol{\theta}$ is the weights vector, $\boldsymbol{x}$ is the input vector and b is the bias. In 4.1 an artificial neuron is depicted. The weights are found by minimizing the distance between the computed output value $\hat{y}_i$ and the real observed output y_i [26].

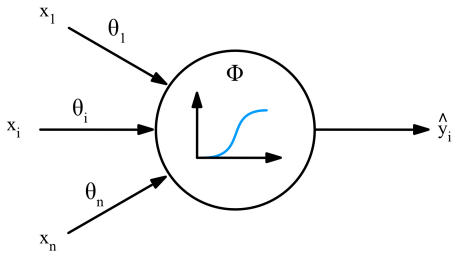


Fig. 4.1: An artificial neuron or perceptron, based on a depiction from [26]

4.2 Deep Feedforward Networks

As explained in 4.1 the human brain not only consists of a single neuronal cell but of billions of interconnected neurons. Likewise, a multilayer perceptron (MLP) contains several neurons, which are interconnected and hierachically structured in a layer-wise fashion. As a result, they form a so-called neural network.

MLPs, also described as deep feedforward networks, are quintessential in the field of deep learning models. Their goal is to approximate a function f^*. For instance, it defines a mapping $y = f(x;\theta)$ for a classifier $y = f^*(x)$ that maps an input x to a category y. By learning the best value for the parameters θ the deep neural network provides the best fitting approximation for the function. The application of deep neural networks started only in the 2000s, especially around 2010, when computer power became affordable enough to allow the use of more complex models on large amounts of data.

In contrast to a perceptron, which describes a single neuron, a multilayer perceptron is a neural network that consists of at least three layers: an input layer, at least one hidden layer and an output layer. While the input of the input layer and the output of the output layer are known, the behaviour of the other layers is not directly specified by the training data. The learning algorithm must decide how to use those layers between input and output layer to best implement an approximation of the target function. Because the desired output of these layers is not shown by the training data they are called hidden layers.
The hidden layer of a neural network is typically vector valued and its dimensionality determines

the width of the model. A layer can be thought of as consisting of many units, each representing a vector-to-scalar function, that act in parallel. Each one of these units resembles a neuron, which receives inputs from several other units and computes its own output.

Neural networks are called networks, because they compose together different functions. Chain structures are the most commonly used structures. If three exemplary functions $f(1)$, $f(2)$ and $f(3)$ in a network that approximates a function $f(x)$ in order to find the best representation for x were to be connected in a chain, the structure would have the form $f(x) = f(3)(f(2)(f(1)(x)))$. The first layer would be $f(1)$, the second $f(2)$, and so on. The length of this chain gives the depth of the model. This terminology formed the term "deep learning". If the network contains more than one hidden layer, as in the example above, it is referred to as a deep neural network.

In deep feedforward networks the information flows forward through the network. This is also described as forward-propagation. If a feed-forward neural network includes feedback connections, in which the outputs of the model are fed back into itself, it is called a recurrent neural network (see Section 4.3.2). Furthermore, a feedforward network is called a fully connected network if the nodes of one layer are fully connected to those of the preceding layer.

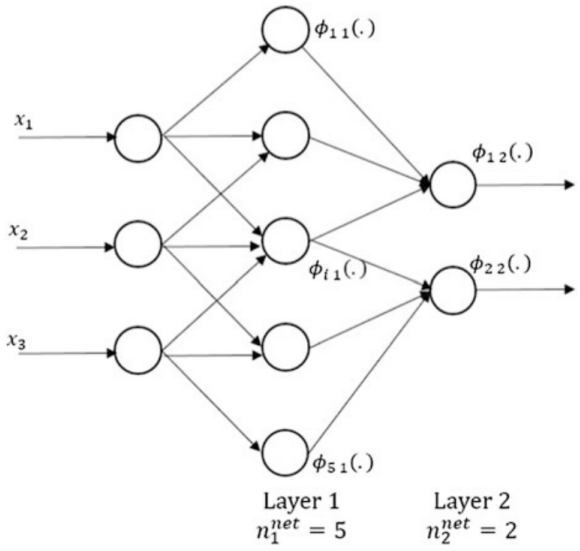


Fig. 4.2: Feed-forward neural network with one hidden layer. This figure is taken from [26].

Neuroscientific observations inspired the idea of using many layers of vector-values representation and the choice of functions used to compute those representations. However, nerual network research now is guided by various mathematical and engineering disciplines. The goal is not to perfectly model the brain anymore, but to approximate functions, that are designed to achieve statistical generalisation. In order to train a deep neural network, the architecture of the network has

to be determined. This includes the selection of the amount of layers in the network, how these layers should be connected and how many units a layer should contain.
Each unit in a layer is determined by the weights that are assigned to its connections with other units as well as a bias term. During the training of a multilayer neural network those synaptic weights are adjusted to minimize a preselected cost function. The multilayer structure of deep neural networks complicates the computation of the gradients, which are involved in the optimization process. The following Section 4.2.1 introduces the back-propagation algorithm, which enables neural networks to efficiently compute the gradients from complicated functions [20][22][3][26].

4.2.1 Backpropagation

Backpropagation is a training algorithm and an efficient approach for fitting neural networks. It is impossible to train a deep learning network without it. In this algorithm the result of the cost function flows backwards through the network to compute the gradient descent. The goal is to minimize the error between the target value and the output of the network. For instance the mean squared error can be used as a cost function.
While the term is often misunderstood to mean the whole learning algorithm for a multilayer neural network, just like feed-forward propagation does, back-propagation actually only refers to the method of computing the gradients in a simple and inexpensive way. Consequently, backpropagation is not the opposite of feed-forward propagation, but a feed-forward network can be trained by back-propagation [22][26].
The weights for all the units of the network define a large hypothesis space. The backpropagation algorithm implements a gradient descent search in this space of possible network weights. It iteratively reduces the error between the target output values and the computed network outputs. Considering networks with multiple output units rather than single units, the value E that shall be minimized can be defined according to [27] as

$$E(\boldsymbol{\omega}) = \frac{1}{2} \sum_{j \in D} \sum_{i \in outputs} (y_{ij} - \hat{y}_{ij})^2 \tag{4.2}$$

where $outputs$ represents all output units of the network. y_{ij} and $\hat{y}_{ij}$ are the target output and the computed output, respectively. The error surface of multilayer networks can have multiple local minima. As a result, the gradient descent can converge towards a local minimum and get trapped in it. Consequently, it would stop converging towards the global minimum error. However, every weight adds one dimension to the search space. Thus, networks with a large number of weights correspond to very high dimensional spaces and error surfaces. Due to this, the gradient descent is unlikely to get stuck in a local minimum, as more dimensions provide more ”escape routes” for the gradient descent to fall away from the local minimum. As a consequence, backpropagation is an

effective function approximation method in practice and has been found to produce great results in various real-world applications [27].

4.2.2 Hyperparameters

Hyperparameters describe settings that control the behavior of the neural network algorithm. Usually the values of the hyperparameters are not computed by the learning algorithm itself, but set from the start. However, there is the possibility of implementing a learning procedure that learns the best hyperparameters for another learning algorithm [22]. In [12] the hyperparameters are optimized with genetic algorithms. This chapter explains some of the most important hyperparameters.

1. Learning Rate
   The learning rate of a neural network affects the training of a network. According to [28] the learning rate or step size is the most important hyperparameter, accounting for more than two thirds of the variance. Its optimal value is dependent on the dataset. A small value leads to smooth learning curves, but slow convergence. A higher value results in faster convergence, but the learning curve tends to be oscillatory. A solution for this are time-varying step sizes. In the search space for each data set there is an area with good learning rates of which the performance does not differ much. The learning rate has a bigger impact on the variance than the hidden layer size, which suggests that the learning rate can be tuned on a small network with less effort and then used to train a larger one [28][20].

2. Epochs
   Epochs are the number of training iterations over the whole dataset. Thus, the number of epochs also affects the training of a neural network. An epoch is completed once all samples have run through the neural network. After a successive finite number of epochs the algorithm is guaranteed to converge. In batch training, the weights of the network are updated after one epoch, as the batch size equals the training set size. The opposite is sequential training, where each training vector is sequentially passed trough the algorithm and the weights are updated after each sample [22][20].

3. Batch size
   The batch size affects the data set used for training the network. In batch training the batch size equals the training data set size, whereat in the minibatch scheme the batch contains less samples than the epoch. As a result, the weights are not updated after every epoch, but after every minibatch. In the context of deep learning with large data training sets, minibatch schemes are favored for various applications. The choice of the minibatch size depends not

only on the application and the dataset but also on the available computing power. Minibatch and batch schemes have an averaging effect on the computed gradients [20].

4. Hidden Layer Size
   The larger the network, the more time is required to train it. Larger networks work better, but with diminishing returns [28]. The number of hidden layers determine the depth of a machine learning model and the dimensionality of the layers its width. Thus, the hidden layer size affects the architecture of a neural network.

4.3 Deep Learning Architectures

The architecture of a neural network depends on its purpose and as a consequence numerous different architectures exist. The following chapter will focus on those structures that are applied in this work and investigate the motivation behind their usage. Futhermore, the difference between convolutional neural networks and recurrent neural networks, which are the precessor of long short term memory neural networks, is discussed [26].

4.3.1 Convolutional Neural Networks

The multilayer networks introduced in Section 4.2, require that feature vectors are used as an input for the network. Convolutional networks however, integrate the feature generation phase into the training of the neural network. As a result, the features from the data are learned together with the networks parameters. The name ”convolutional” derives from the convolutions performed by the first layers of the network instead of inner products. CNNs are often used for image classification, as they work well for processing data that has a known grid-like topology [20][22].
The convolution operation is visualized in 4.3 and can be seen in equation 4.3, whereat w is the kernel, x is the input and $s(t)$ is the output or so-called feature map:

$$s(t) = (x * w)(t). \tag{4.3}$$

Usually the input is a multidimensional array of data, while the kernel is a multidimensional array of parameters that are learned in the process. In traditional neural network layers every output unit interacts with every input unit via matrix multiplication. In contrast, convolutional networks have sparse interactions or sparse weights, which is obtained by a kernel that is smaller than the input. Thus, fewer parameters need to be stored reducing the memory requirements and improving the statistical efficiency. This enables the network to describe complicated interactions between many variables with simple building blocks that describe sparse interactions [22].

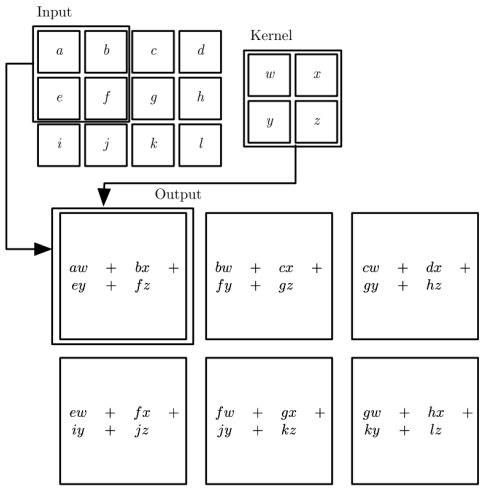


Fig. 4.3: An example of 2-D convolution taken from [22]. The arrows indicate how the upper-left element of the output tensor is computed when the kernel is applied to the corresponding upper-left region of the input tensor.

4.3.2 Recurrent Neural Networks

Recurrent neural networks, short RNNs, are networks that are especially good at processing sequential data. This means the input vectors are not independent of each other and the order of the input data is of importance. RNNs can scale to very long sequences as well as process sequences of variable length.

A siginificant idea behind recurrent networks as well as convolutional networks is the sharing of parameters across different parts of the model. Therefore, a recurrent neural network can share the same weights across several time steps. This enables the network to generalize to sequence lengths not seen during training as well as share statistical strength across different sequence lengths and different positions in time.

Through the introduction of states the network is able to take previous time steps into account and learn time-dependent patterns. A state vector $\boldsymbol{h_n}$ encodes the past history up to the current time n. Every state vector is influenced by the state $\boldsymbol{h}_{n-1}$ as is shown in equation 4.4 and together with the input vector $\boldsymbol{x_n}$ computes the output vector $\boldsymbol{y_n}$.

The symbol of $\boldsymbol{h}$ shows that it is a vector of hidden variables or hidden layers. It represents the memory of the model. Furthermore, a set of unknown parameter matrices and vectors, namely U, W, V, $\boldsymbol{b}$ and $\boldsymbol{c}$ has to be learned during training [20] [22].

$$\boldsymbol{h}_n = f(U\boldsymbol{x}_n + W\boldsymbol{h}_{n-1} + \boldsymbol{b}) \tag{4.4}$$

$$\boldsymbol{y_n} = g(V\boldsymbol{h}_n + \boldsymbol{c}) \tag{4.5}$$

The network has to keep information for as long as possible in order to learn time-dependant patterns over time. The gradient of the error is computed to minimize the cost function. In networks

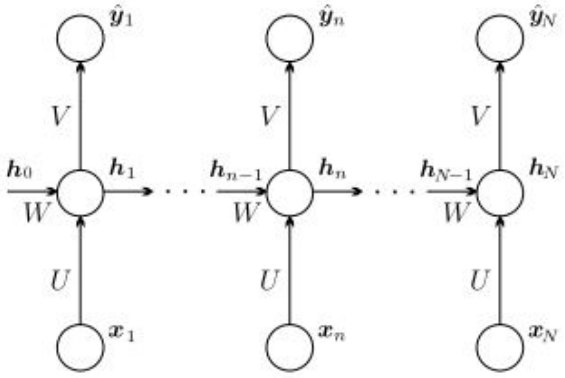


Fig. 4.4: This figure is taken from [20] and shows the process of a RNN over time .

with numerous hidden layers the repeated multiplication of very small gradients close to zero can lead to vanishing gradients. This stops the network from learning. Due to the vanishing gradient problem information can not be kept over longer sequences. A solution for this probem are LSTMs, which are explained in the next section 4.3.3. The learning process of the network also stops if very big gradients are multiplicated resulting in exploding gradients [29].

4.3.3 LSTM

LSTMs (long short term memory) belong to the so-called gated recurrent neural networks. Their goal is to prevent both vanishing and exploding gradients caused by long-term dependencies within the network. As a consequence, LSTMs can store information over a longer sequences than RNNs. This is achieved by complementing the hidden variables vector $\boldsymbol{h}$ with a new state vector $\boldsymbol{s}$. Combined they represent the system's memory.
In addition, LSTM networks consist of nonlinear elements known as gates that control the information flow into and out of the memory of the system. Those nonlinear gates are the sigmoid function σ and the hyperbolic tangent function *tanh*. They enable the network to forget information that has already been used and is no more needed. The state vector $\boldsymbol{s_n}$ can be reset, written to and read from according to the forget gate $\boldsymbol{f}$, the input gate $\boldsymbol{i}$ and the output gate $\boldsymbol{o}$ [20][22][29][23][30].

Apart from the input vector $\boldsymbol{x_n}$, the LSTM unit receives $\boldsymbol{s}_{n-1}$ and $\boldsymbol{h}_{n-1}$ for every timestep n and passes $\boldsymbol{s_n}$ and $\boldsymbol{h_n}$ to the next timestep. As it was the case in Chapter 4.3.2 the matrices U, W, V and the bias vector $\boldsymbol{b}$ have to be learned during training. The associated updating equations 4.6 to 4.10 are depicted in Figure 4.5. The state unit $\boldsymbol{s}_n$ has a linear self-loop. Its weight is controlled by the forget gate unit $\boldsymbol{f}$ and set to a value between 0 and 1 via the sigmoid unit:

$$\boldsymbol{f} = \sigma(U^f \boldsymbol{x}_n + W^f \boldsymbol{h}_{n-1} + \boldsymbol{b}^f) \tag{4.6}$$

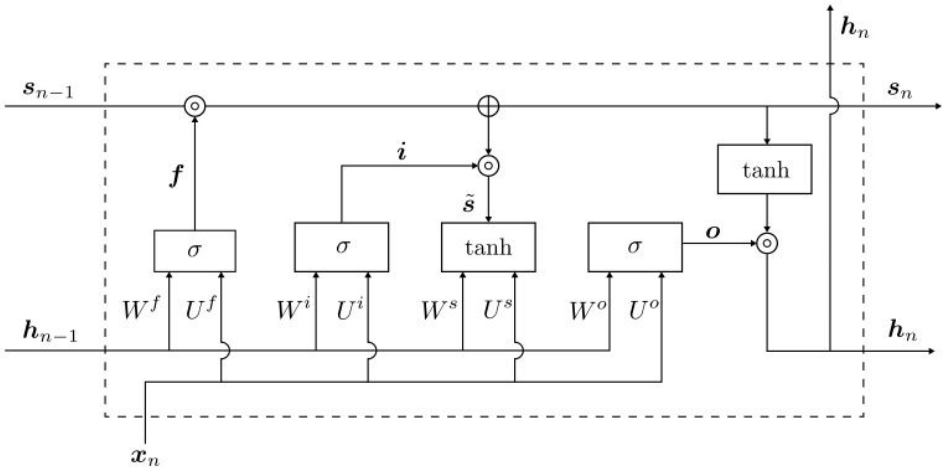


Fig. 4.5: A basic LSTM unit as depicted in [20].

where $\boldsymbol{x_n}$ is the current input vector and $\boldsymbol{h_n}$ is the current hidden layer vector. Thus, the LSTM cell internal state is updated to

$$\boldsymbol{s}_n = \boldsymbol{s}_{n-1} \circ \boldsymbol{f} + \boldsymbol{i} \circ \hat{\boldsymbol{s}} \tag{4.7}$$

The external input gate $\boldsymbol{i}$ is computed similarly to the forget gate, but with its own parameters:

$$\boldsymbol{i} = \sigma(U^i \boldsymbol{x}_n + W^i \boldsymbol{h}_{n-1} + \boldsymbol{b}^i) \tag{4.8}$$

Th output $\boldsymbol{h}_n$ is defined as

$$\boldsymbol{h}_n = \boldsymbol{o} \circ tanh(\boldsymbol{s}_n) \tag{4.9}$$

and can be shut off via the output gate $\boldsymbol{o}$

$$\boldsymbol{o} = \sigma(U^o \boldsymbol{x}_n + W^o \boldsymbol{h}_{n-1} + \boldsymbol{b}^o), \tag{4.10}$$

which also uses a sigmoid function for gating.

$$\hat{\boldsymbol{s}} = tanh(U^s \boldsymbol{x}_n + W^s \boldsymbol{h}_{n-1} + \boldsymbol{b}^s) \tag{4.11}$$

The vectors $\boldsymbol{f}$, $\boldsymbol{i}$, $\hat{\boldsymbol{s}}$ and $\boldsymbol{o}$ update the two states $\boldsymbol{s}$ and $\boldsymbol{h}$ as can be seen in Equation 4.7 and 4.9. The element-wise product, also called Hadamard product, is denoted by $\circ$ while σ denotes the sigmoid function. The first gate $\boldsymbol{f}$ controls what information is passed via the cell state $\boldsymbol{s}_n$ from the previous instant to the next one. The elements in $\boldsymbol{f}$ take values between 0 and 1, depending on the hidden variables and the current input that is received from the previous stage. Thus, the weighting is adjusted to the context. Several different variants of LSTM structures exist [20][30].

4.3.4 Autoencoders

Autoencoders are able to automatically learn features from unlabeled data and compute reconstructions that are close to the original input. As can be seen in Figure 4.6 autoencoders consist of two parts, the encoder and the decoder. The architecture of the decoder is often the mirror image of the architecture of the encoder. In the encoder the number of units are reduced. This dimensionality reduction is expected to help extract those features that represent the data very well [31][20] [32].

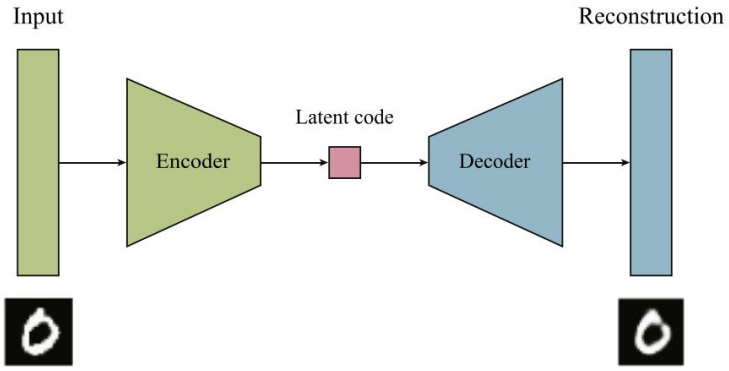


Fig. 4.6: An autoencoder consists of two neural networks, the encoder and the decoder. The input data passes through the encoder and is mapped to the latent space. Then it is transferred to the decoder to generate the output. This depiction is taken from [31].

The encoder maps the input data to an internal latent representation. Latent variables are not part of the dataset but part of the model. This latent represenation is used to produce an output similar to the input data pattern. The single hidden layer feed-forward neural network is defined in terms of a vector function by Equation 4.12, whereat ϕ_e is the activation function and is usually a sigmoid function.

$$\boldsymbol{f} : \boldsymbol{x} \in \mathbb{R}^l \longrightarrow \boldsymbol{h} \in \mathbb{R}^m \tag{4.12}$$

where

$$h_i := f_i(\boldsymbol{x}) = \phi_e(\boldsymbol{\theta}_i^T \boldsymbol{x} + b_i) \qquad i = 1,2,...,m \tag{4.13}$$

The decoder is defined as follows.

$$\boldsymbol{g} : \boldsymbol{h} \in \mathbb{R}^m \longrightarrow \hat{\boldsymbol{x}} \in \mathbb{R}^l \tag{4.14}$$

where

$$\hat{x}_j := g_j(\boldsymbol{h}) = \phi_d(\boldsymbol{\theta'}_i^T \boldsymbol{h} + b'_i) \qquad j = 1,2,...,l \tag{4.15}$$

The activation ϕ_d is either the identity or linear reconstruction or the logistic sigmoid function. During training the following parameters are estimated.

$$\Theta := [\theta_1, ..., \theta_m], b, \Theta' := [\theta'_1, ..., \theta'_l], b' \tag{4.16}$$

The given data x is processed in the encoder to generate the latent representation h. This vector h is then transferred to the decoder, which computes the output vector $\hat{x}$. The reconstruction error $e = x - \hat{x}$ is the difference between the input x, which is fed into the encoder and the re-generated input $\hat{x}$ that is computed by the decoder. In the learning process the parameters are estimated in a way that the reconstruction error is minimized [20][23][31][33].

Autoencoders can be used to detect outliers in an unsupervised way, as it was done by Andrea Borghesi at the University of Bologna in [34]. Preprocessing is required to obtain a data set containing only good data. After training the network with good data and thus minimizing the reconstruction error for normal behaviour, anomalies can be detected by observing changes in the reconstruction error. With this method faulty conditions can be identified even though they have not been encountered earlier in the training phase [34].

4.3.5 Variational Autoencoders

A variational autoencoder, abbreviated to VAE, is a generative model. This means it can not only reconstruct the input data as an output, but also generate new and random output data. Its structure is similar to the one of an autoencoder, as it also consists of an encoder and a decoder, which are each a neural network on their own. However, a VAE models the parameters of the distribution of an observation, for instance the probability density functions (PDF), rather than the value itself. It can be trained with gradient-based methods.
The encoder delivers a representation of the input via latent random variables to the generative model or decoder. Inside an iteration the parameters are updated according to “expectation maximization” learning. The goal is to learn meaningful representations of the data.
The variational autoencoder learns stochastic mappings between an observed data set or $\boldsymbol{x}$-space and a latent space $\boldsymbol{z}$. The encoder $q_\varphi(\boldsymbol{z}|\boldsymbol{x})$ is used to obtain $\boldsymbol{z}$ during training. A sample distribution $p_\theta(\boldsymbol{z})$ is passed through the decoder. Eventually, $\boldsymbol{x}$ is extracted from a distribution. $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ can be defined as the decoder network. The process is shown in figure 4.7.

Using a single feedforward pass the encoder models the relation between input and latent variables in a reasonably fast way. In the process, however, sampling noise is added to the gradients required for learning. Therefore, the variance increases. This can be solved by the so-called “reparameterization trick”, where the gradient computation is reorganized and thus the variance in the gradients is reduced [33][22][8].

Variational autoencoders belong to Bayesian machine learning. Bayes theorem is a way to update the system belief as new evidence in form of the data x is received. The probability of a

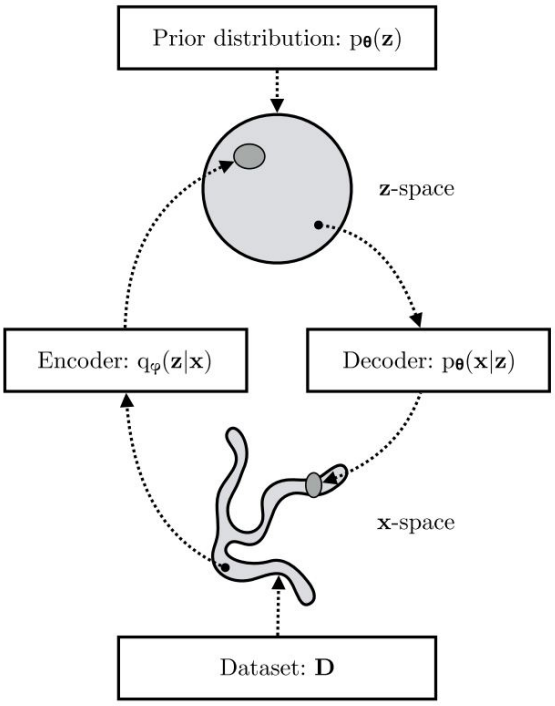


Fig. 4.7: A variational autoencoder consists of two neural networks, the encoder $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ and the decoder $q_{\theta}(\boldsymbol{x}|\boldsymbol{z})$. The input data $\boldsymbol{D}$ passes through the encoder and is mapped to the $\boldsymbol{z}$-space. Then it is transferred to the decoder to compute the $\boldsymbol{x}$-space. This depiction is taken from [33].

hypothesis or latent variable z is then given by

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{4.17}$$

where $p(x)$ is the probability of the data x and $p(z)$ is the probability of hypothesis z. $p(x|z)$ is the probabilty of the data given a hypothesis z or likelihood probabilty. Bayes theorem derives from the conditional probability axiom, which is based on the definition of joint probability. The joint occurance of an event X and an event Y can be described as

$$p(X \cap Y) = p(X|Y)p(Y). \tag{4.18}$$

The 'AND' is commutative, thus the following is valid:

$$p(X \cap Y) = p(Y \cap X) = p(Y|X)p(X), \tag{4.19}$$

$$p(X|Y)p(Y) = p(Y|X)p(X). \tag{4.20}$$

Dividing both sides by $p(Y)$ results in Bayes theorem:

$$p(X|Y) = \frac{p(Y|X)p(X)}{p(Y)}. \tag{4.21}$$

A central task of generative models is to compare two different distributions. A commonly used measure for this purpose is the Kullback-Leibler divergence. The information content decreases when the probability of an event increases. This is similiar to pointing out something 'obvious' to someone. Such a remark does not increase the amount of available information. Thus, the information is inversely related to the probability of an event. Since $log(p(x))$ is directly related to $p(x)$, the negative $-log(p(x))$ is inversely related to $p(x)$.
The information content of an event x in two distributions p and q can be defined as

$$I_p(x) = -logp(x) \tag{4.22}$$

and

$$I_q(x) = -logq(x). \tag{4.23}$$

Thus, the difference of information is

$$\Delta I = I_p - I_q = -logp(x) + logq(x) = log\left(\frac{q(x)}{p(x)}\right). \tag{4.24}$$

The Kullback-Leibler is the expectation of the information difference with respect to the distribution of $q(x)$ and given by

$$D_{KL}(q(x)||p(x)) := E_q[\Delta I] = \int (\Delta I)q(x)dx = \int q(x)log\left(\frac{q(x)}{p(x)}\right). \tag{4.25}$$

The KL divergence is always non-negative

$$D_{KL}(q(x)||p(x)) = -\int q(x)log\left(\frac{p(x)}{q(x)}\right). \tag{4.26}$$

As shown above, the encoder portion of a VAE yields an approximate posterior distribution $q(z|x)$. The weights parametrizing the encoder network are denoted as φ. Thus, the encoder is properly described as $q_\varphi(z|x)$. The decoder yields the likelihood distribution $p_\theta(x|z)$, being parameterised by different weights θ. The KL divergence between the approximate and the real posterior distribution is

$$D_{KL}(q_\varphi(z|x_i)||p(z|x_i)) = -\int q_\varphi(z|x_i)log\left(\frac{p(z|x_i)}{q_\varphi(z|x_i)}\right)dz \geq 0. \tag{4.27}$$

Applying the Bayes' theorem produces

$$D_{KL}(q_\varphi(z|x_i)||p(z|x_i)) = -\int q_\varphi(z|x_i)log\left(\frac{p\varphi(x_i|z)p(z)}{q_\varphi(z|x_i)p(x_i)}\right)dz \geq 0. \tag{4.28}$$

Applying rules of logarithms and several other arithmetic operations lead to

$$logp(x_i) \geq \int q_\varphi(z|x_i)\left[logp_\varphi(x_i|z) + logp(z) - q_\varphi(z|x_i)\right]dz. \tag{4.29}$$

The right hand side of the above inequality can be recognized as Expectation. The end result after two more transformation steps is

$$logp(x_i) \geq \int (q_\varphi(z|x_i) log \left(\frac{p(z)}{q_\varphi(z|x_i)} \right) dz + \int q_\varphi(z|x_i) log p_\theta(x_i|z) dz, \tag{4.30}$$

$$logp(x_i) \geq -D_{KL}(q_\varphi(z|x_i)||p(z)) + E_{q_\varphi(z|x_i)} [logp_\theta(x_i|z)] . \tag{4.31}$$

The right hand side is the variational or evidence lower bound ELBO. It bounds the likelihood of the data and is seeked to be maximized. The Kullback-Leibler term is a constraint on the form of the approximate posterior and thus a regularizer [35].

In a variational autoencoder the objective for optimisation is the evidence lower bound, abbreviated to ELBO. The ELBO is a lower bound on the log-likelihood of the data in the latent space. The difference between the loglikelihood of the variable in the latent space and the ELBO is given by the Kullback-Leibler (KL) divergence. The KL divergence is nonnegative, thus the ELBO has at most the same value as the desired log-probability. Consequently, the goal is to minimise the KL divergence and maximise the ELBO [33][22][8][32].

The key difference between autoencoders (see Section 4.3.4) and variational autoencoders is that the VAE is a stochastic generative model that delivers probabilites as an output, while an AE is a deterministic discriminative model without a probabilistic foundation. This enables the VAE to interpolate between classes in the latent space and generate new output [32]. Furthermore, autoencoders are trained by minimizing the error loss, while variational autoencoders can be trained by maximizing the evidence lower bound [33].

4.3.6 Variational LSTM Autoencoder

A variational LSTM autoencoder, according to [8] abbreviated to LSTM-VAE, is a variational autoencoder containing a LSTM layer in its encoder and decoder network, respectively.
In [32] a VAE is used to detect anomalies in handwritten digits. In this data set the single observations are independent of each other. However, in time series data the sequence of the data points is important. VAE models are not able to analyse information beyond a short time window and outlier detection algorithms that are only using VAEs often fail in detecting long term anomalies. In [7] the combination of LSTM and VAE is used to identify outliers that span over a longer time sequence. While the LSTM module allows the estimation of long term correlation, the VAE detects the features.

The LSTM-VAE used in this work is based on [11]. Its structure is depicted in Figure 4.8. The LSTM layer included in the variational autoencoder enables the detection of outliers in unlabeled time series data. Autoencoders can handle unlabeled data very well. The advantage of VAE over AE, however, is that while the output of the AE only has meaning in a defined interval, this is not the case for the probabilites delivered by the VAE.

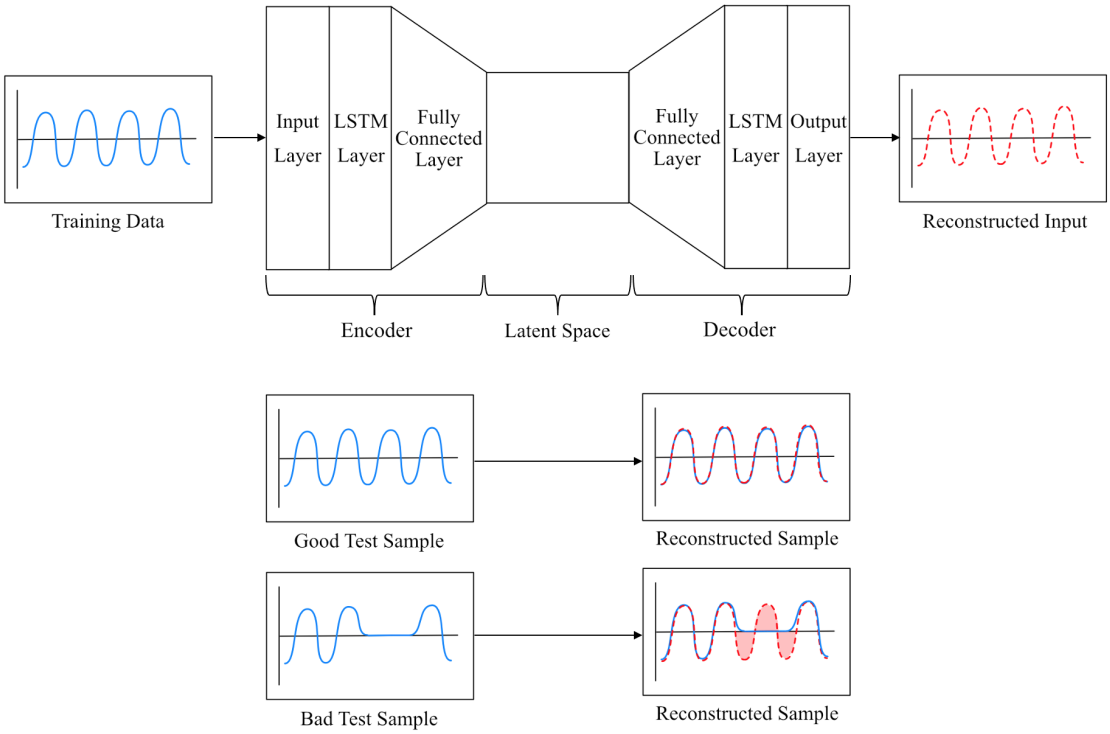


Fig. 4.8: The encoder and the decoder of the LSTM-VAE used in [11] consist of a sequential input or (in case of the decoder) output layer, the LSTM layer and a fully connected layer for dimensionality reduction, respectively. Inbetween encoder and decoder is the latent space, which is introduced in Section 4.3.5. In the training process the model learns a representation (red) of the input data (blue). Below the depiction of the structure of the model two examplary test data points are given. The top one represents a non-anomalous test sample. When the network is trained with non-anomalous data the reconstructed input fits the original input very well and the reconstruction error is very low. The bottom sample shows an outlier. Outliers are ”unknown” to the network, thus the reconstruction error is rather large and the outlier can be detected with statistical thresholding techniques.

Chapter 5
Application of a Hybrid Deep Learning System for Anomaly Detection

This chapter introdcues a hybrid hyperparameter-optimized deep learning method to detect outliers in time-series data. The first part provides background information about the available dataset and the underlying physical process. Part two of this chapter investigates the application of descriptive statistics on KPIs in the context of outlier detection, while the third part explains the deep learning component of the hybrid model. In the fourth part the results of the combined model are evaluated. Part five reflects on the limitations and problems of the model.

5.1 Data Source

The data used in this work derives from machinery involved in the construction of foundation anchors for new buildings. The construction process consists of several steps. The two most important ones are penetration and compaction. In the penetration phase a hole is drilled into the ground. During compaction, this hole is filled with gravel from a gravel tank, which is not one continuous process, but interrupted by the compression of the filling [36].
As depicted in Figure 5.1, the input contains nine channels which together provide multivariate time-series data for each sample. One sample represents one foundation anchor.
One site contains several hundred samples. The length of the time-series data in each sample differs.

The obtained data is partly anomalous. This affects both, statistical and deep learning outlier detection. As stated in Section 2.3.4, using methods based on the median instead of the mean for thresholding decreases the negative impact of outliers in the data for the statistical component. However, non-anomalous data is essential for the training of the neural network as well as the hyperparameter optimization. This is further discussed in Section 5.3.3.

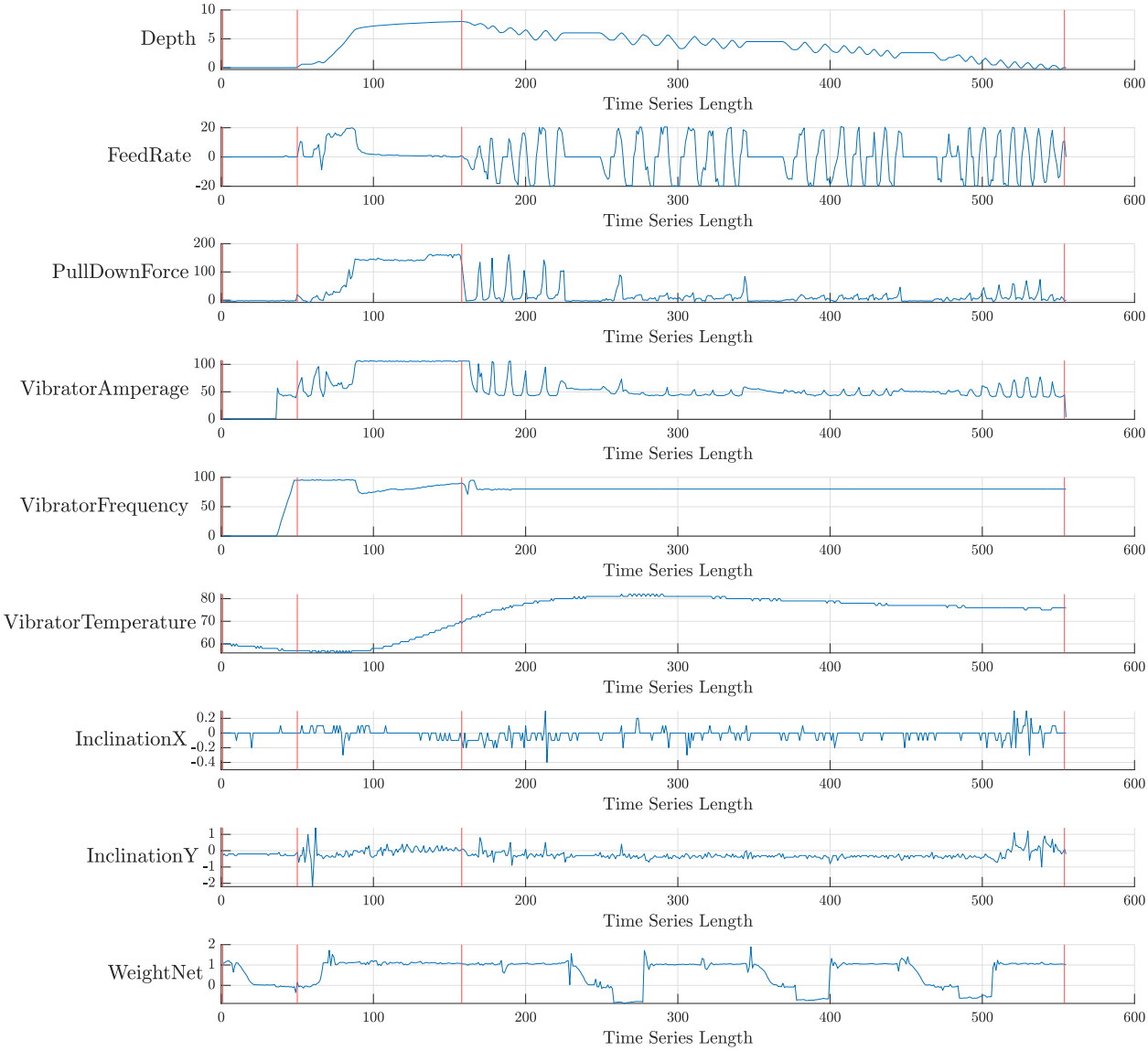


Fig. 5.1: The time-series data of one sample is shown. The y-axis describes the nine input channels respectively. The x-axis depicts the time. The red lines represent the boundaries between different phases. The first phase is the start phase, followed by the penetration step. Then the compaction phase takes place.
The input channels of an exemplary good sample are depicted. After a short start phase the penetration phase starts and the depth increases. If the maximum depth is reached, the compaction phase is initiated. The depth overall decreases, with some oscillating phases and pauses inbetween. The oscillation depicts the compaction, when the drilling head goes up and down in order to compress the gravel that is filled into the drilling hole. The pauses, in which the depth is constant, mark the periods when new gravel is filled into the emptied gravel tank. The feedrate is increasing when the drilling head goes up and gravel is loaded into the hole. When the head goes down the feedrate decreases, as there is no additional gravel fed into the hole when the gravel already in the hole is compressed. The pulldown force increases when the hole is drilled and when the gravel is compressed.

5.2 Statistical Analysis Approach: Descriptive Statistics for KPIs

The statistical outlier detection is conducted by applying descriptive statistics on KPIs. This component of the hybrid approach is based on ongoing work conducted by the Chair of Automation at the University of Leoben. A broad domain and physical knowledge are required in order to develop a statistical model that automatically detects outliers in the construction process. For the detection of anomalies in the provided time series data the following steps are executed: Segmentation of the time-series data, definition of the KPIs and calculation of the threshold.

5.2.1 Segmentation

Most industrial processes consist of several phases or steps. As explained in Section 5.1, the process investigated in this work consists of two steps: penetration and compaction. The penetration phase starts as soon as the force exceeds a certain threshold. When the maximum depth is reached, the penetration phase ends and the compaction phase begins. The end of the compaction phase is determined by the amperage of the vibrator. If it falls below a certain threshold, the process has ended. The thresholds are set taking physical knowledge about the process into account.

5.2.2 Key Performance Indicators

According to [37], key performance indicators (KPIs) focus on those aspects of the performance that are the most critical ones for the current and future success of the process. For descriptive analytics those features are the most relevant ones. In most industries, KPIs are used to describe the current status of a process. The definition of KPIs requires manual work and knowledge about the underlying process [38][39].

A different amount and different types of KPIs are assigned to each segment of the process. Examples for compaction KPIs are the number of gravel fills per compaction phase or the average compaction length per gravel fill. The KPIs are calculated with the nine input variables as well as the measured time.

5.2.3 Calculation of the Outlier Threshold

Figure 5.2 shows the calculated outliers per KPI and sample for the penetration phase of one site. For each sample the IQR per KPI based on the median is calculated. The output is a logical array. The logical arrays of the single KPIs are added up to the outlierness of the single samples in a site. IQR is applied a second time in order to extract a binary array containing the anomalous samples of the site.

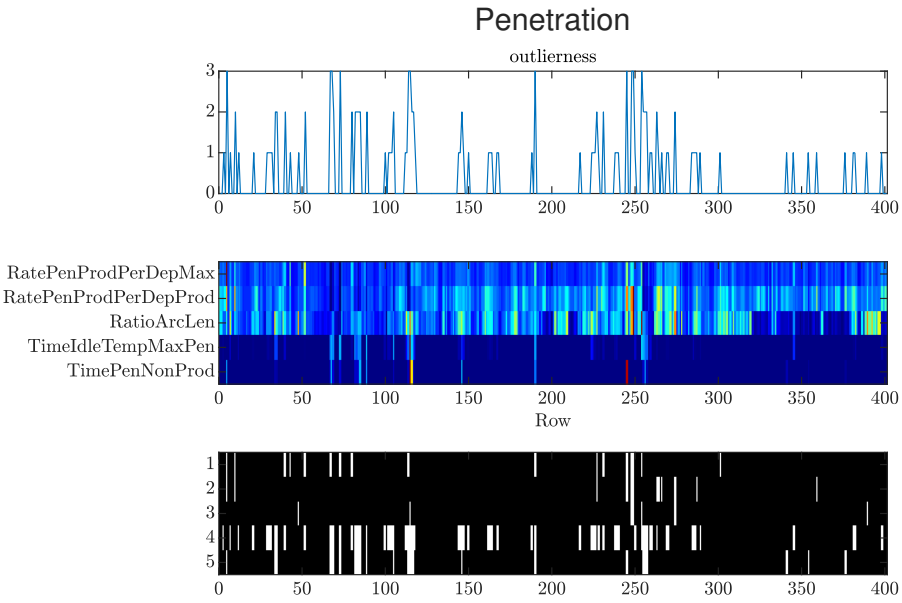


Fig. 5.2: The three graphs visualize the outlierness in the penetration phase. In all three graphs the x-axis depicts the samples. In the first graph the outlierness per sample by adding up the outliers of the single KPIs is shown. In the second graph a heat map for the five KPIs of the penetration phase is plotted. One advantage of the heatmap is, that patterns in the outlierness of the KPIs are easily recognizable. As the order of the samples is not randomized possible fields of outliers on the site can be detected. The white fields in the graph on the bottom represent the outliers per sample and per KPI. A white area indicated an anomalous KPI for this sample.

5.3 Deep Learning Approach: LSTM-VAE

The used deep learning model is a LSTM-VAE, a variational autoencoder containing a LSTM layer in the encoder and decoder, respectively. The algorithm is based on [11]. In Section 4.3.6, its theoretical background is explained. The LSTM-VAE learns the underlying structure of the training data, which is in the best case the normal behaviour of the physical process. Its output are the reconstruction error and the ELBO loss. Both are expected to be relatively low for non-anomalous data points, if trained with non-anomalous data. Consequently, a larger loss implies the existence of an outlier. IQR is applied on both, the sum of the reconstruction error and the ELBO loss per sample, in order to compute a binary array containing the outliers of the site.

5.3.1 Cross-Validation

To prevent overfitting, different datasets have to be provided for the optimization of the hyperparameter, the training and the testing of the neural network.
Thus, it is necessary to divide the available dataset into different subsets. According to [22], fixed portions of training and test data can be problematic if the test set ends up being small. This leads to statistical uncertainty around the estimated average test error. One solution to use all examples in the estimation of the mean test error is the repetition of the training and testing computation on different, randomly chosen subsets of the original data. A common procedure is k-fold cross-validation, which forms several partitions of the dataset by splitting it into k non-overlapping subsets. When a partition is used for testing, the rest of the overall dataset is used for training the network. By taking the average test error across k trials the mean test error can be estimated. Naive estimators, that do not take into account the error correlations due to the overlap of training and test sets lead to an underestimation of the variance. This problem is pointed out in [40].

Based on [22], in this work 25% of all the samples are randomly taken and assigned to the dataset for the hyperparameter optimization. The other samples are used for the training and testing of the deep learning network. Both, the dataset for hyperparameter optimization and the dataset for training and testing apply 3-fold cross validation, respectively. In contrast to the time-series data contained in the samples, it is valid to rearrange and randomize the sequence of the samples, as there is no temporal relevance attached to them.

In 3-fold cross validation two thirds of the data set for training and testing are used for the training of the deep learning network. The other third is applied on the trained network to evaluate its performance on reconstructing the data for the detection of outliers. To introduce as little bias as possible, the apportionment of the data between the three folds in the testing and training dataset is

not fixed, but can be randomly generated for each run. However, for better comparability between the runs and to demonstrate the high range of different results caused by the random initial weights in the neural network, the folds are fixed for 20 runs. In one run each fold is used for testing once and for training twice. Due to the extensive computation time of the hyperparameter optimisation used from [12], which takes around 22 to 24 hours, the apportionment is fixed between the datasets for hyperparameter optimisation and the testing and training.

In order to achieve a better comparability between the results from different runs of the LSTM-VAE, each fold of the 3-fold cross validation is used as a test data set for 20 runs, respectively. Concurrently, each fold is furthermore used in the training of the network for 40 runs. Due to the random weights assigned to the network at the beginning of the learning process, the results are expected to differ over several runs, also if the network is trained with the same dataset.

5.3.2 Sample Labelling

In Chapter 3 supervised, unsupervised and semisupervised learning are explained. In principle, the available dataset provides no knowledge about the output. Thus, the deep neural network is confronted with unsupervised learning.
However, in [7] and [8] the network is trained on non-anomalous data in order to precisely detect outliers via the reconstruction error. The better the network learns to reconstruct the normal behaviour of the physical system, the better its accuracy in detecting outliers is. Anomalous training data negatively influences the learning process and leads to a less distinctive reconstruction error.
Using manipulated data for the training process leads to bias in the system, because certain hypothesises (the ones corresponding to non-anomalous samples) are preferred over others. In [41], a bias is categorised as being appropiate, if the hypothesis space does contain good approximations to the target function. The target function of this work is the representation of the non-anomalous process, thus the introduced bias is considered valid.

By plotting the time-series data for each sample and looking at it, erroneous behaviour is identified and the sample is manually marked as a point of interest, abbreviated to POI. Marked samples show either

1. very long process pauses as depicted in Figure 5.3 or
2. round-trips in the depth variable, as shown in Figure 5.4 or
3. pauses in the process in which the vibrator amperage and frequency values drop down to zero, as can be seen in Figure 5.5.

All three cases indicate the occurance of a possible problem in the construction process. Due to a lack of deeper knowledge about the physical process, potential outliers that do not fall into these

categories are not considered as manually labelled POIs. However, they still could have a negative impact on the deep learning algorithm. Also outliers that depend on more than one dimension are ignored, as in the manual labelling process each variable is treated as a univariate input.
It is not guaranteed that those data points, that are manually labelled as interesting are in fact outliers and negatively impact the process in any way. There is also no guarantee that manual labelling and descriptive statistics correctly detect all the outliers prior to the training of the deep learning network. Thus, the training data is not necessarily completely free of outliers after the application of manual labelling and descriptive statistics.

Both, statistically detected outliers and manually labelled POIs are optionally removed from the datasets for training the network and for hyperparameter optimisation. As partly labelled data is used in the process, the model is considered to be semi-supervised. Eventually, the manual labelling delivered 124 POIs in 401 samples, which is a rate of slightly more than 30%.

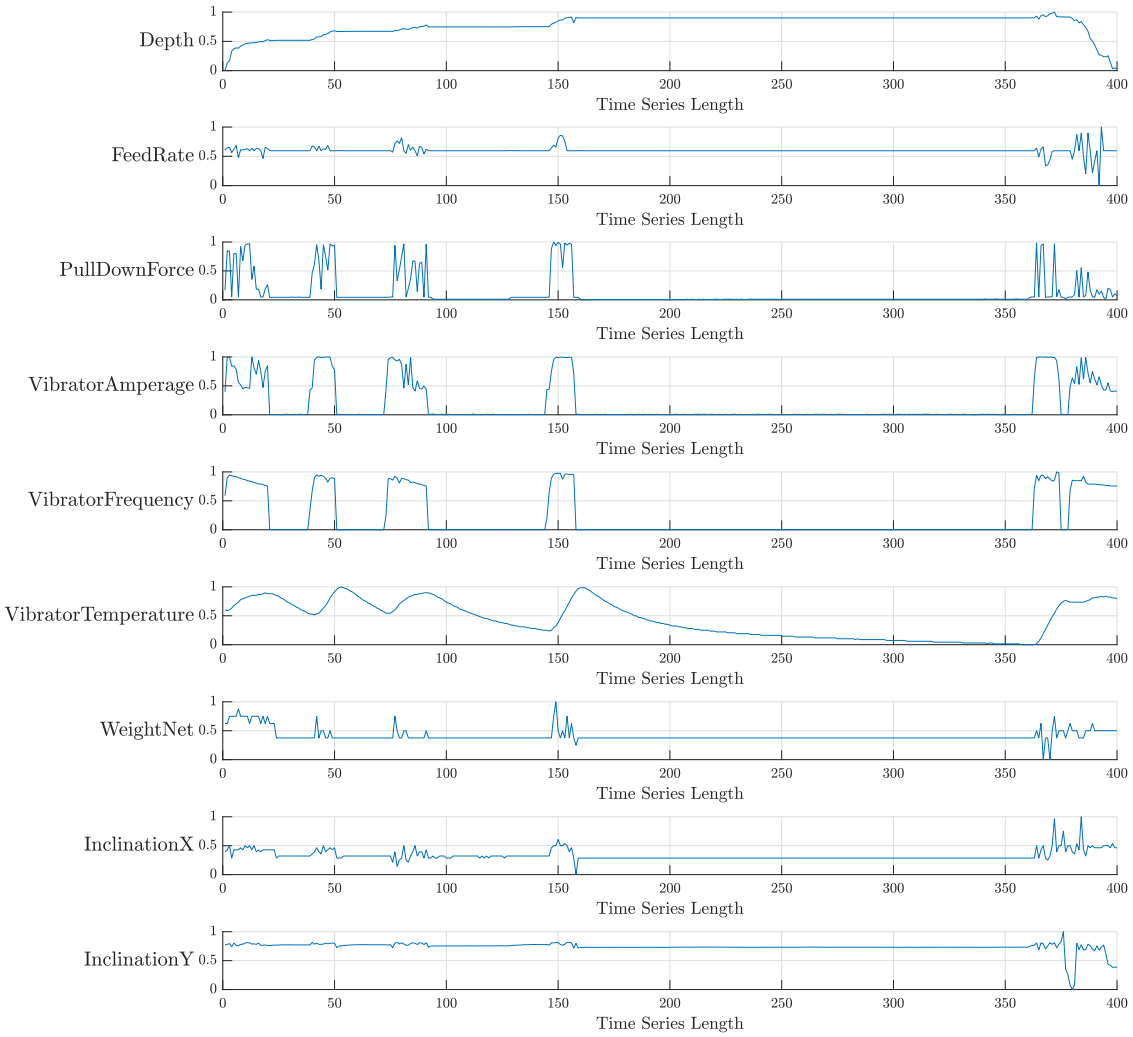


Fig. 5.3: This sample exhibits a long pause between the penetration and the compaction process and is thus considered an point of interest in the manual labelling process. Long pauses can be caused either by functional failure or human error. In any case, long pauses are undesireable, as they are not time-efficient and, as a result, costly.

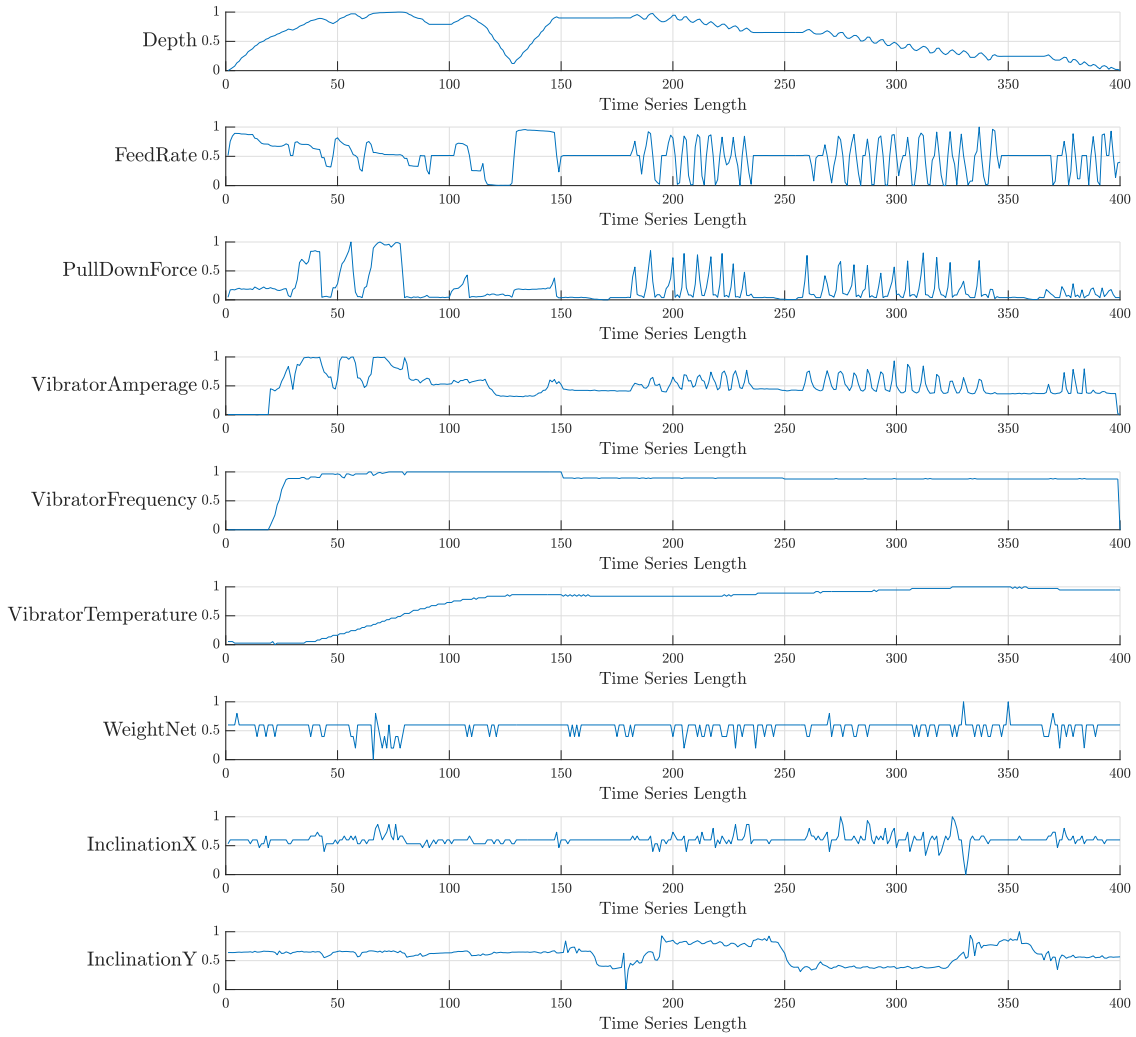


Fig. 5.4: A roundtrip describes an instance where the drilling head elevates to a depth of almost zero, which is the starting point of the overall construction process, in the middle of either the penetration or the compaction phase. This indicates some technical issue, but could also be attributed to human error. It is important to clarify what caused the roundtrip to ensure the quality of the sample.

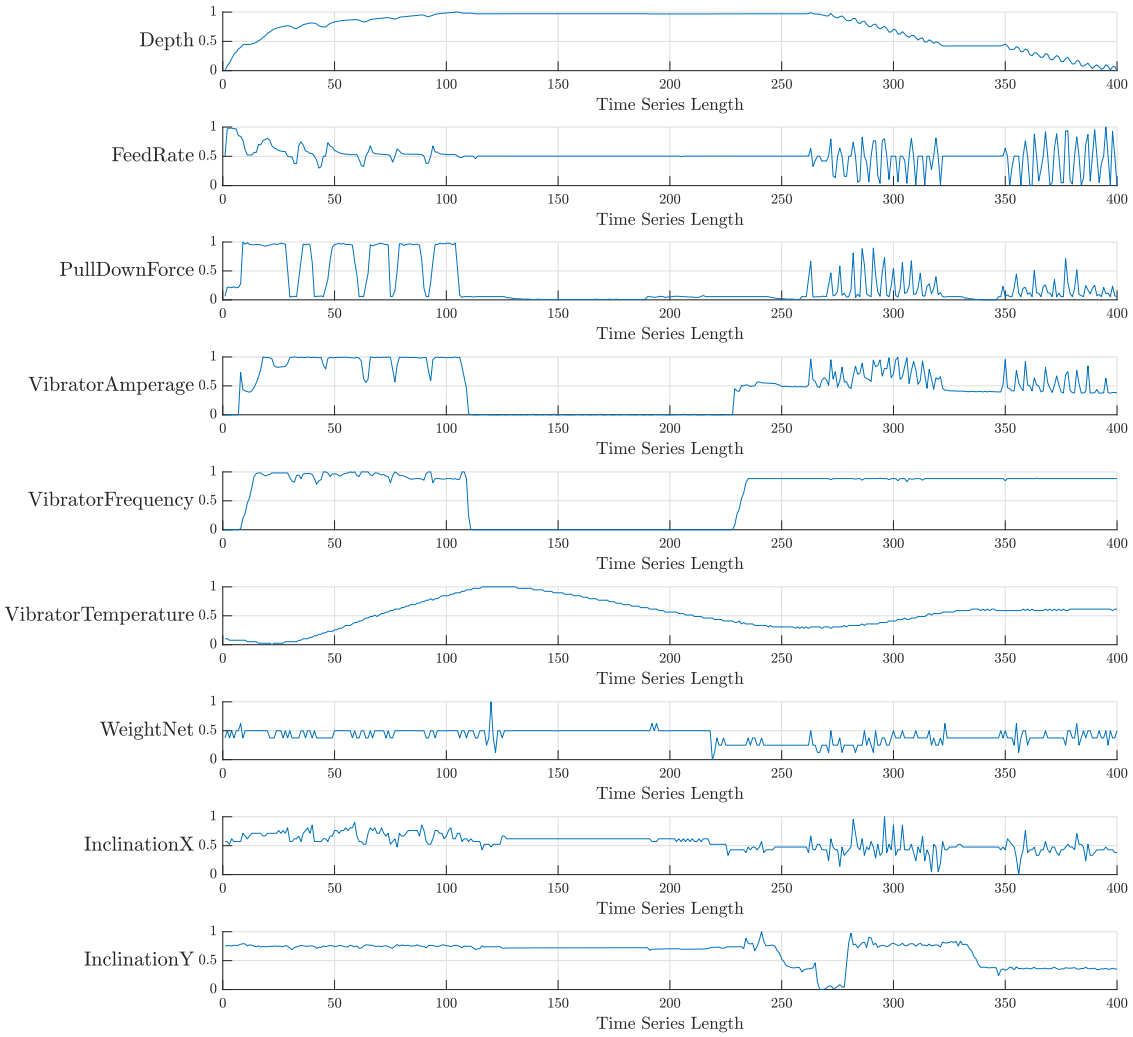


Fig. 5.5: A pause in which the vibration amperage and frequency is dropping to zero is considered a point of interest, as it is not clear whether or not this has an impact on the final quality of the sample.

5.3.3 Preprocessing of the Input

According to [20] and [22], it is recommended to preprocess the input data of a neural network, because the raw data is difficult to represent for many deep learning algorithms. The variables of the training set should have approximately zero mean to avoid large differences in the dynamic range of the various input values.
Both, the training and the testing set, should be scaled to have a similar variance, assuming that all variables are equally important. This can reduce generalization error and the size of the model needed to fit the training set.
The preprocessing steps applied on the test and training data for the LSTM-VAE are taken from [11]. First, the input data is resized to a time-series length of 400. This is done by nearest-neighbor interpolation. Next, the values of the variables are rescaled to an interval between 0 and 1.

5.3.4 Training Options and Hyperparameter Settings

The hyperparameters of a model have an enormous impact on its performance. However, finding the best fitting hyperparameters is a challenging task, which is not further investigated in this thesis. The hyperparameters for the deep learning component of the hybrid model are found according to [12] by hyperparameter optimization with genetic algorithms. The data used for the hyperparameter optimization is free of any samples that are considered an outlier by statistical analysis or manual labelling. In Section 4.2.2 the most important hyperparameters for controlling the machine learning system are explained in more detail. The optimized hyperparameters of this work are presented in Table 5.1.

Hyperparameter	**Value**
Number of Hidden Units in the LSTM Layer of the Encoder	44
Number of Hidden Units in the LSTM Layer of the Decoder	69
Number of Training Epochs	77
Learning Rate	95E-9
Mini Batch Size	5

Table 5.1: Values of the optimized hyperparameters for the LSTM-VAE. The hyperparameter optimization is conducted with genetic algorithms based on [12].

5.3.5 ELBO Loss and Reconstruction Error

According to the confusion matrix in Figure 5.6 the ELBO loss and the reconstruction loss categorize different data points as outliers. This is not expected and gives reason, to treat the outliers detected by both of them seperately.

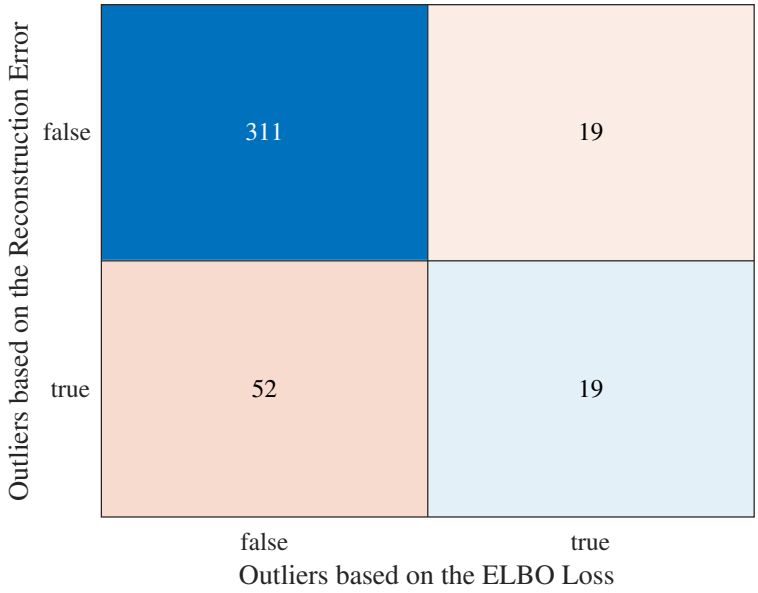


Fig. 5.6: The confusion matrix of the outliers detected by the application of the IQR method on the reconstruction error and on the ELBO loss shows, that both differ in more cases than they match. Thus, the ELBO loss is treated seperatly from the reconstruction error.

The reconstruction loss is calculated as the difference between the reconstructed input data after the decoder and the original input data. It is an array that contains a data point for each time step in the time series. The ELBO loss is a value obtained from the latent space right after the encoder. The ELBO is the difference between the representation of the input sample of the test dataset and the learned representation of the training set in the latent space. It is an array that contains one value per sample.

In figure 5.7 and 5.8 two different types of POIs that are detected as outliers based on the reconstruction error are shown as normalized time-series. In some cases the reconstruction error of a POI is not large enough to be considered anomalous by the IQR method. This is where the ELBO loss steps in, of which one example is shown in 5.9. Further investigation has to be conducted on the representation of the model in the latent space in order to understand, when the ELBO loss is detected as an outlier. If the process pauses are long enough or several short breaks occur in one drilling and filling process, the reconstruction error is sufficient. The ELBO loss performs better at detecting short breaks.

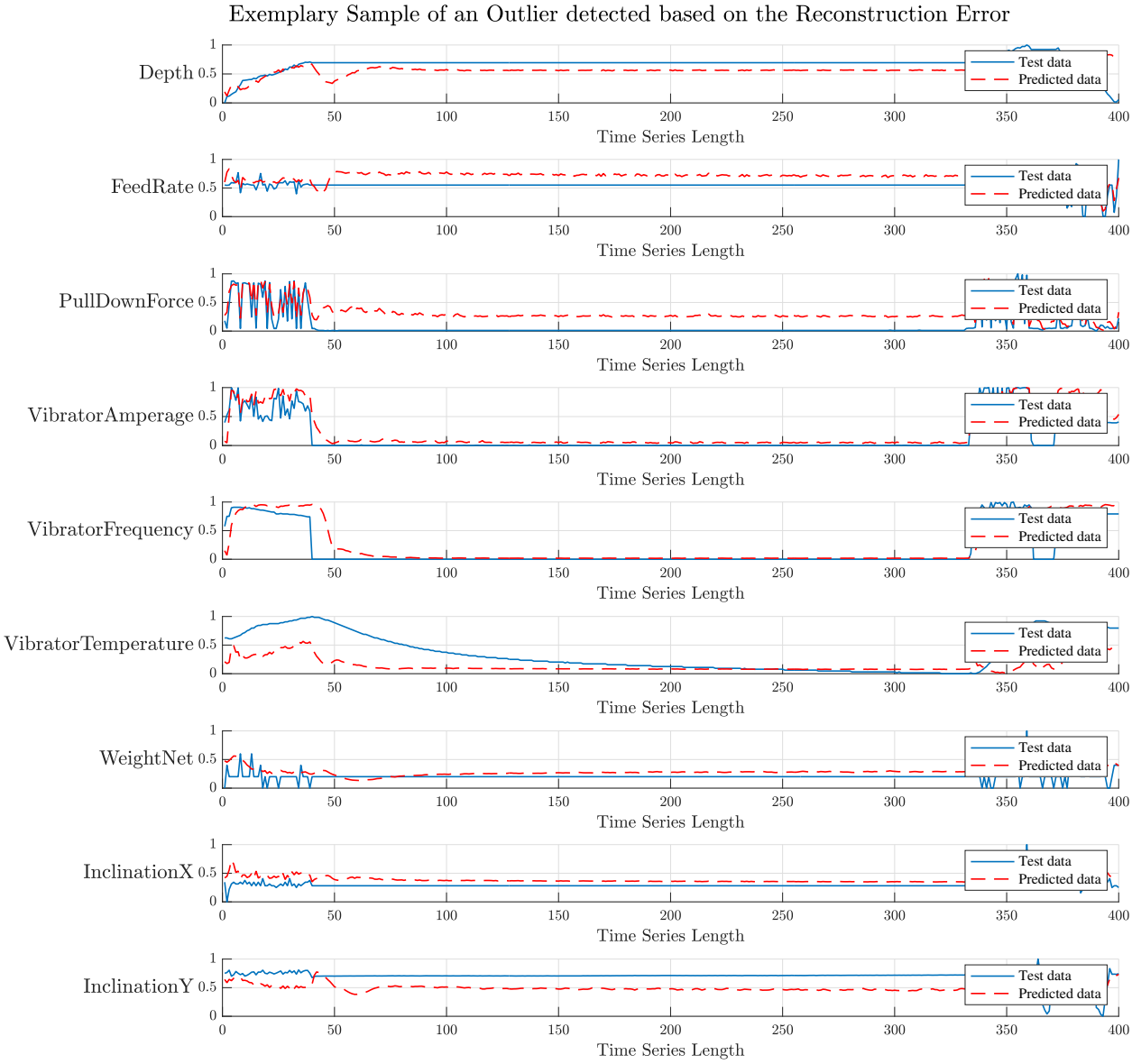


Fig. 5.7: Due to the long pause, the reconstruction error is clearly large enough to be considered an outlier by IQR.

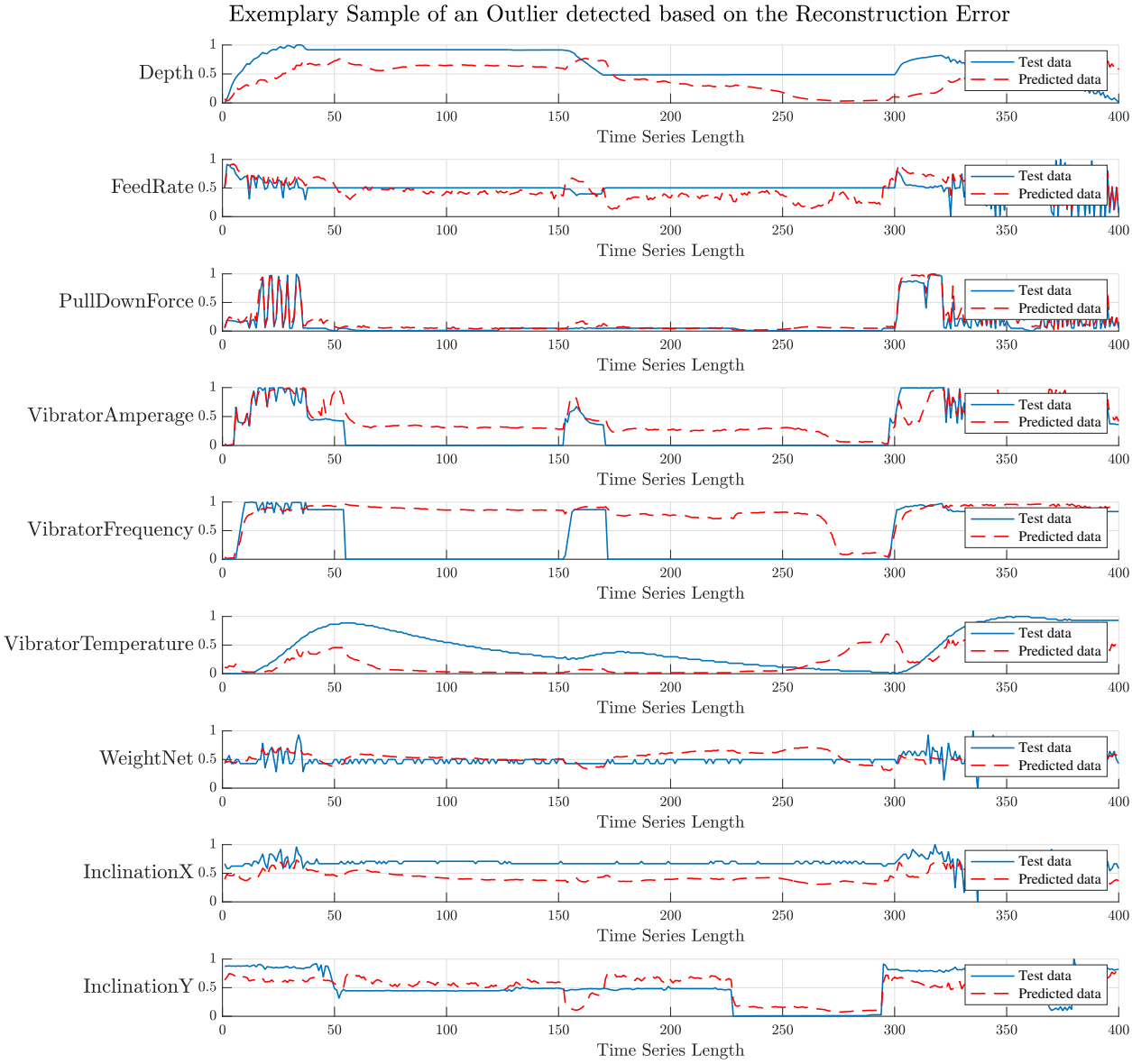


Fig. 5.8: The two pauses add up to a reconstruction error large enough to be detected. The sample is considered an outlier by the deep learning model.

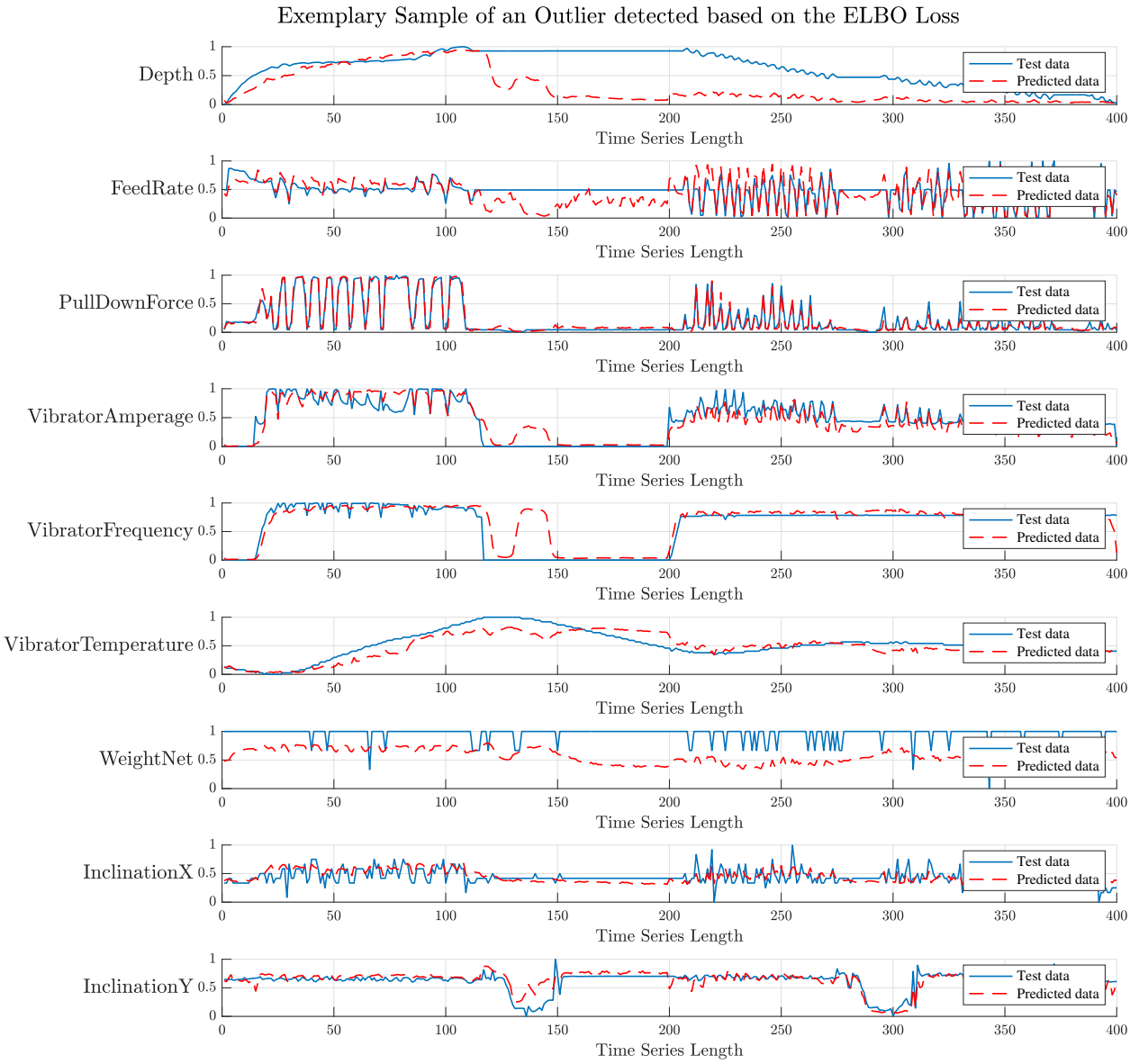


Fig. 5.9: Although considered a POI due to the short pause, the reconstruction error is not large enough for this sample to be declared anomalous when the IQR is computed. However, the IQR of the ELBO loss rules this sample as anomalous.

5.4 Comparison of Statistical Analysis and Deep Learning in Outlier Detection

In this section the results of the statistical outlier detection approach and the LSTM-VAE are compared to each other. The deep learning algorithm for outlier detection is trained with completely unlabelled data. This means, that the training set is assumed to be anomalous and the process unsupervised. Figure 5.11 and the confusion matrix in Figure 5.10 clearly show that outliers detected by the application of descriptive statistics on KPIs differ from those detected by the deep learning algorithm trained with anomalous data.

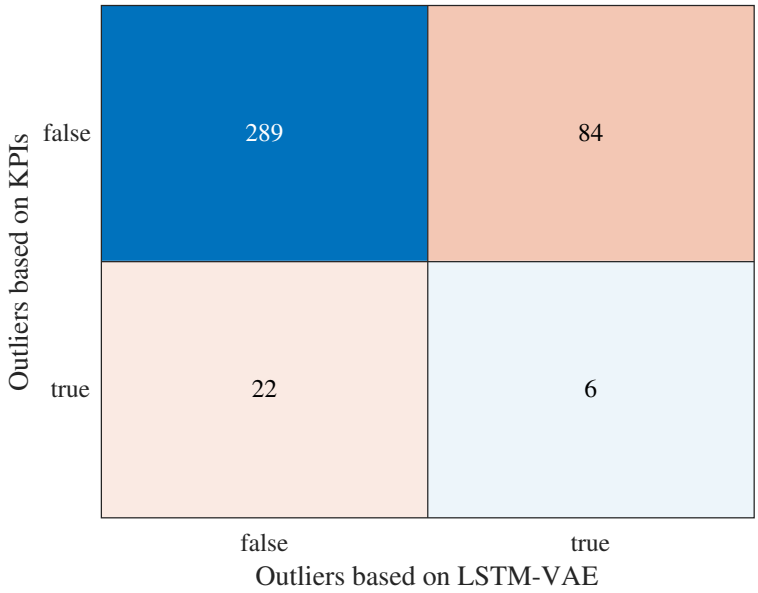


Fig. 5.10: The confusion matrix shows the dissimilarity of the samples considered to be outliers by statistical analysis and deep learning for 20 runs. On first sight it may seem as if the machine learning algorithm detects more outliers than the statistical approach. However, the amount of detected outliers for the LSTM-VAE increases with almost each run. Some samples are marked as outiers in a high number of runs, while others are detected just once in 20 runs. Eventually, a threshold could be needed in order to extract meaningful samples of interest based on the deep learning approach.

While the outliers detected by the statistical approach are constant, the ones detected by the deep learning model slightly change each run, as is shown between Figure 5.11 and 5.12. This leads to a necessary minimum amount of runs to ensure an exhaustive search for outliers. The results of one random run are depicted in figure 5.13 and 5.14. Only one sample is considered anomalous by both, statistical analysis and deep learning. This supports the hypothesis, that deep learning may be helpful to detect outliers that are statistically not traceable.
However, the LSTM-VAE has one disadvantage in contrast to the descriptive statistics. While the latter detects all its outliers for the site at once, the deep learning algorithm detects around 7 outliers

per run on average, whereat the absolute number varies a lot from run to run. This makes several runs necessary.

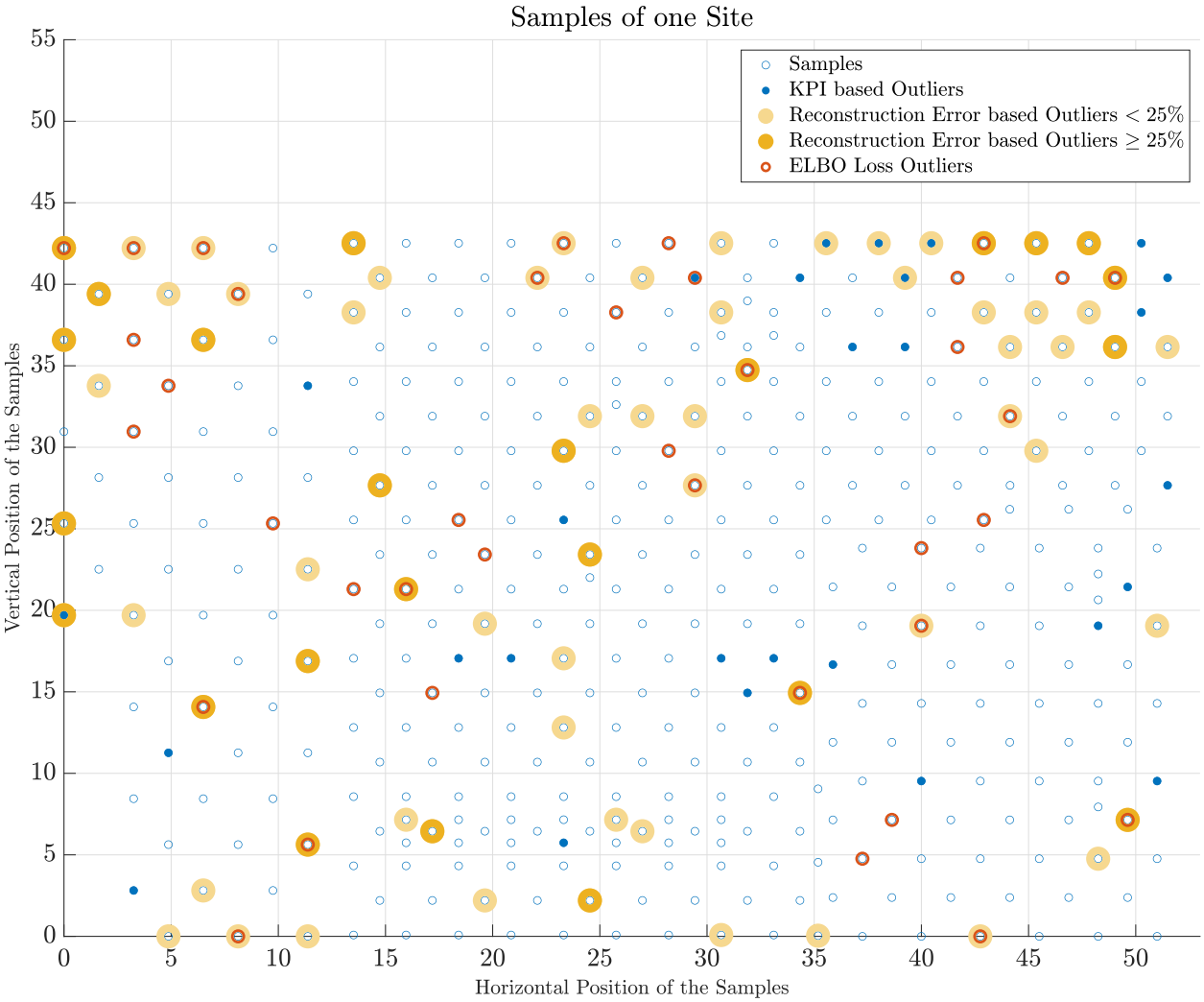


Fig. 5.11: The positions of the samples in a construction site and the corresponding detected outliers are shown. While the KPI outliers are constant, the outliers originating from the LSTM-VAE change with each run. They are distinguished in those deriving from the reconstruction error and those based on the ELBO loss. There is some overlap, but most ELBO loss outliers do not adress the same samples as the reconstruction error outliers. The outliers in the reconstruction loss are marked up to 18 times in 20 runs. Thus, they are detected in 90% of the runs. In contrast, the highest number of matching detected samples is 6 for the ELBO loss. This means the most probable correctly detected anomaly based on the ELBO loss is detected in 30% of the runs. As the range in probability is higher for the outliers detected by the reconstruction error, they are seperated into those that are detected in over 25% of the runs, and those that are marked in less than 25% of the runs. In Figure 5.12 the LSTM-VAE is trained and tested with differently sampled data, thus the detected outliers by the neural network differ. However, there are some outliers unchanged, which indicates that they have a high probability to in fact be anomalous.

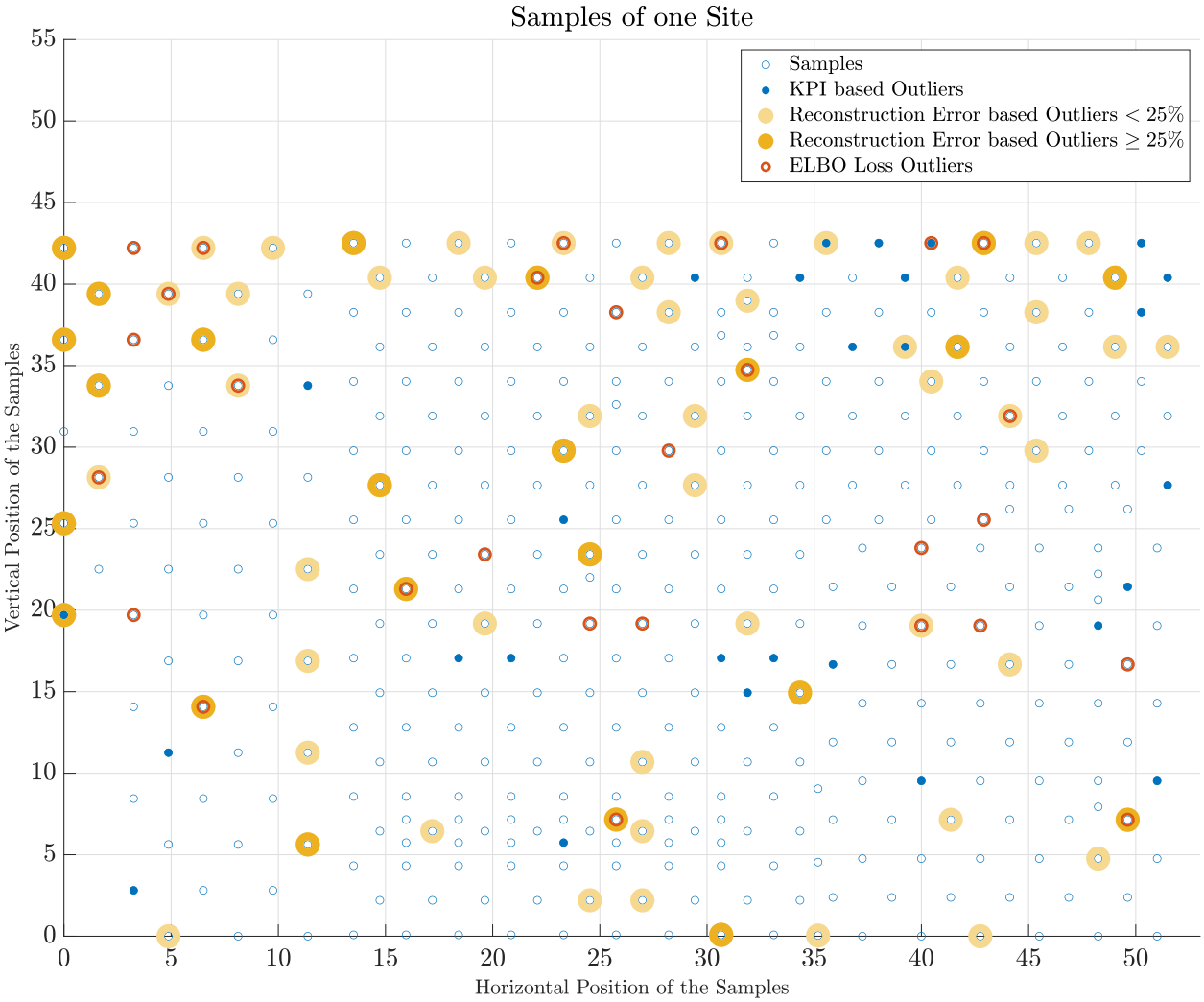


Fig. 5.12: After resampling the data for 3-fold cross validation and 20 runs of training and testing the LSTM-VAE delivers partly different results. Especially those outliers that remain unchanged between different runs strongly support the hypothesis, that the inclusion of machine learning yields new, statistically undetected outliers.

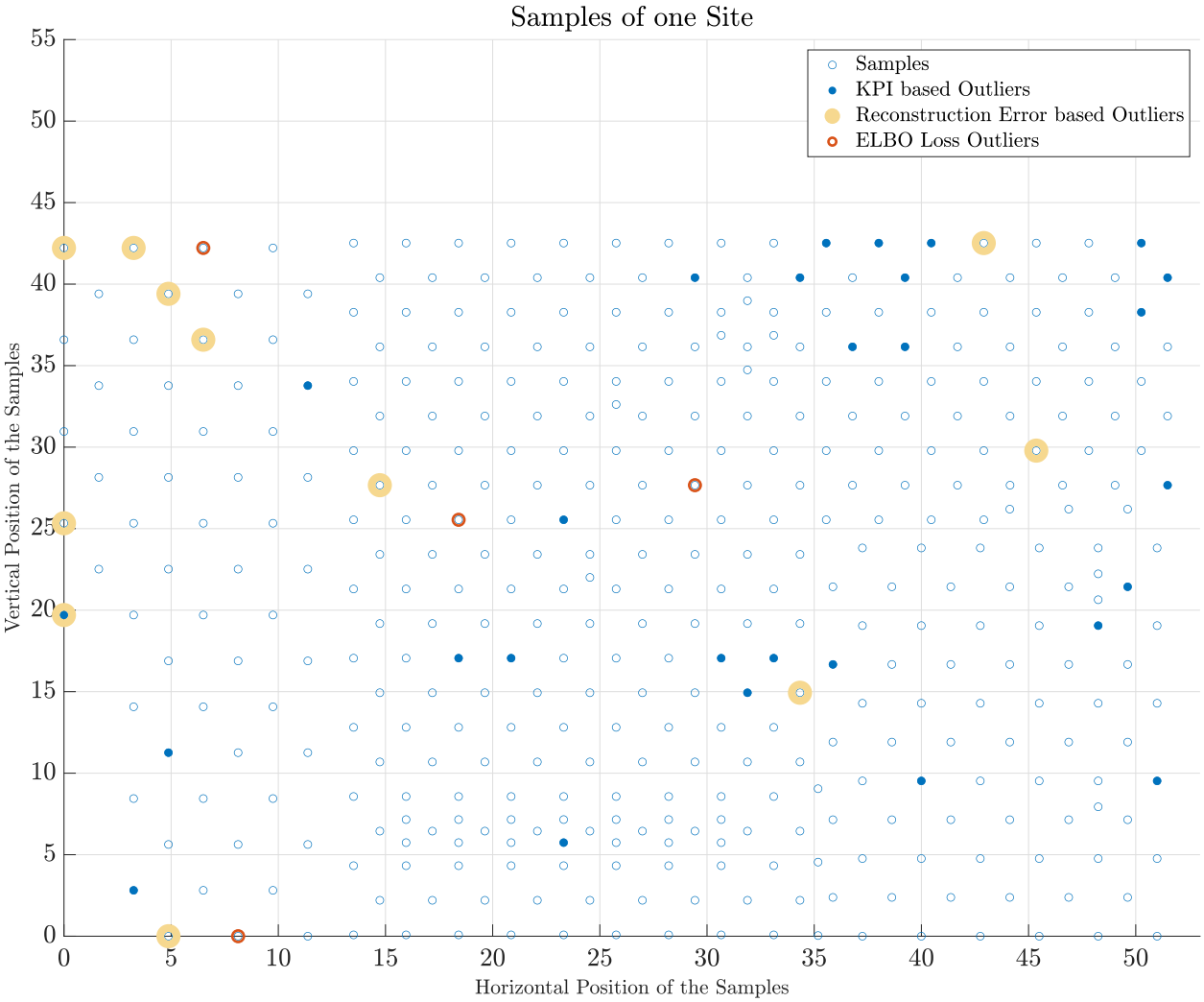


Fig. 5.13: The results for a single run are presented. The deep learning algorithm detects 15 ouliers in this run, one of which matches an outlier marked by the statistical approach. More runs are necessary to ensure a more exhaustive search. The sparse detection of outliers after one run shows, that several runs could be necessary in order to yield more robust results.

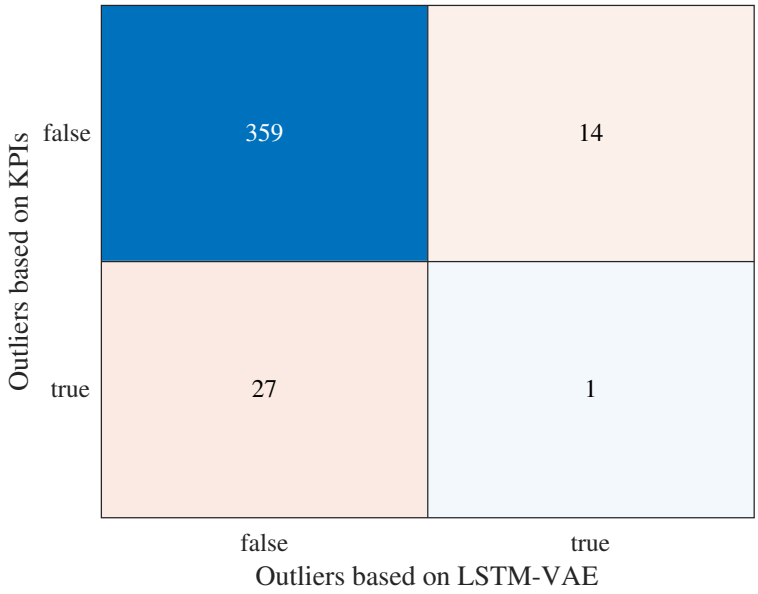


Fig. 5.14: The confusion matrix of one run shows that the application of the IQR method on an outlier score originating from a machine learning model yields very different results than on KPIs.

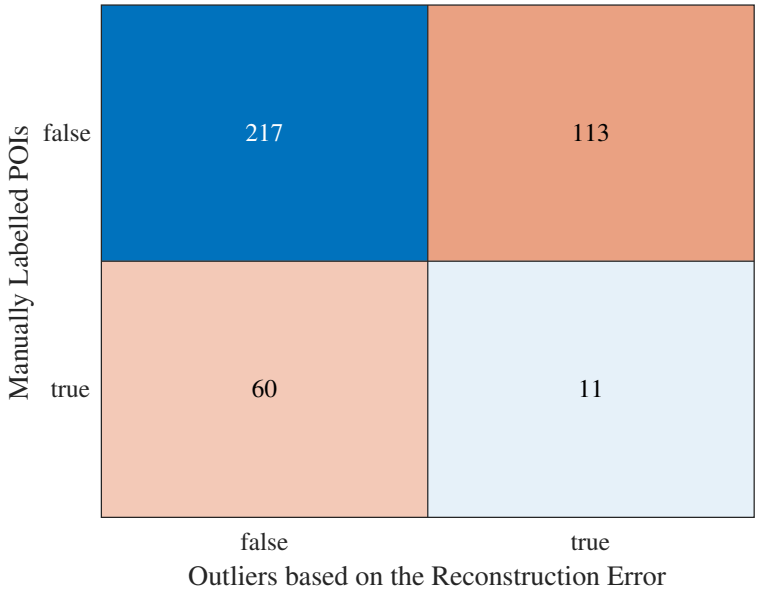


Fig. 5.15: The confustion matrix compares the outliers detected based on the reconstruction error and the manually labelled POIs. Interestingly, the outliers detected based on the ELBO loss have slightly more matches with the manually labelled data, as is shown in 5.16.

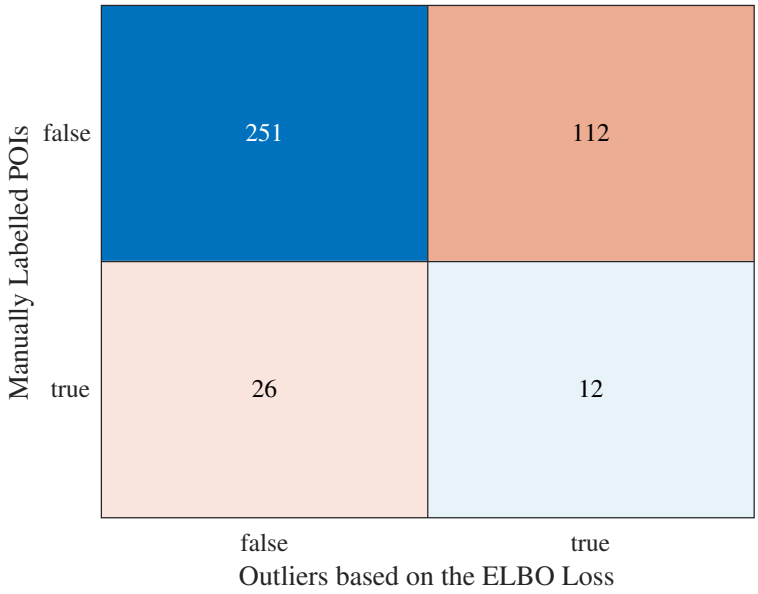


Fig. 5.16: The confustion matrix compares the outliers detected based on the ELBO loss and the manually labelled POIs. The outliers detected based on the ELBO loss have more matches wit the manually labelled POIs than those outliers, that are defined based on the reconstruction error. This indicates that the ELBO loss can help to further improve the accuracy of outlier detection. However, this needs to be further investigated, as the detection rate per 20 runs is still low compared to the outliers defined by the reconstruction error.

Differences between Outliers detected by LSTM-VAE and Descriptive Statistics on KPIs

The conducted computations show, that the outliers detected by the deep learning algorithm are hardly overlapping with those detected by the statistical analysis. This suggests, that the application of machine learning methods in outlier detection are successful in detecting new types of anomalies, which are not traceable by statistical analysis.

A reason for this could be that statistical methods like IQR analyze data in a univariate way, whereat dimensionality reduction, as conducted by the LSTM-VAE, is a multivariate outlier detection technique. In contrast to the input, the output of the LSTM-VAE is a one dimensional outlier score, which can be treated in a univariate manner. Thus, calculating the IQR for the outlier score of the LSTM-VAE delivers hypothetically more accurate results than calculation the IQR for multivariate time-series data.

In figure 5.17, 5.18 and 5.19 three normalized samples which are considered to be outliers by the application of descriptive statistics on KPIs are depicted. They show the kind of outliers detected by statistics while being overlooked by the deep learning component. The generation of KPIs for the physical process demands a lot of work and domain knowledge. In exchange, there is a big advantage to the application of statistics and physical knowledge for the detection of outliers: The results are interpretable and consistent with the physical system.

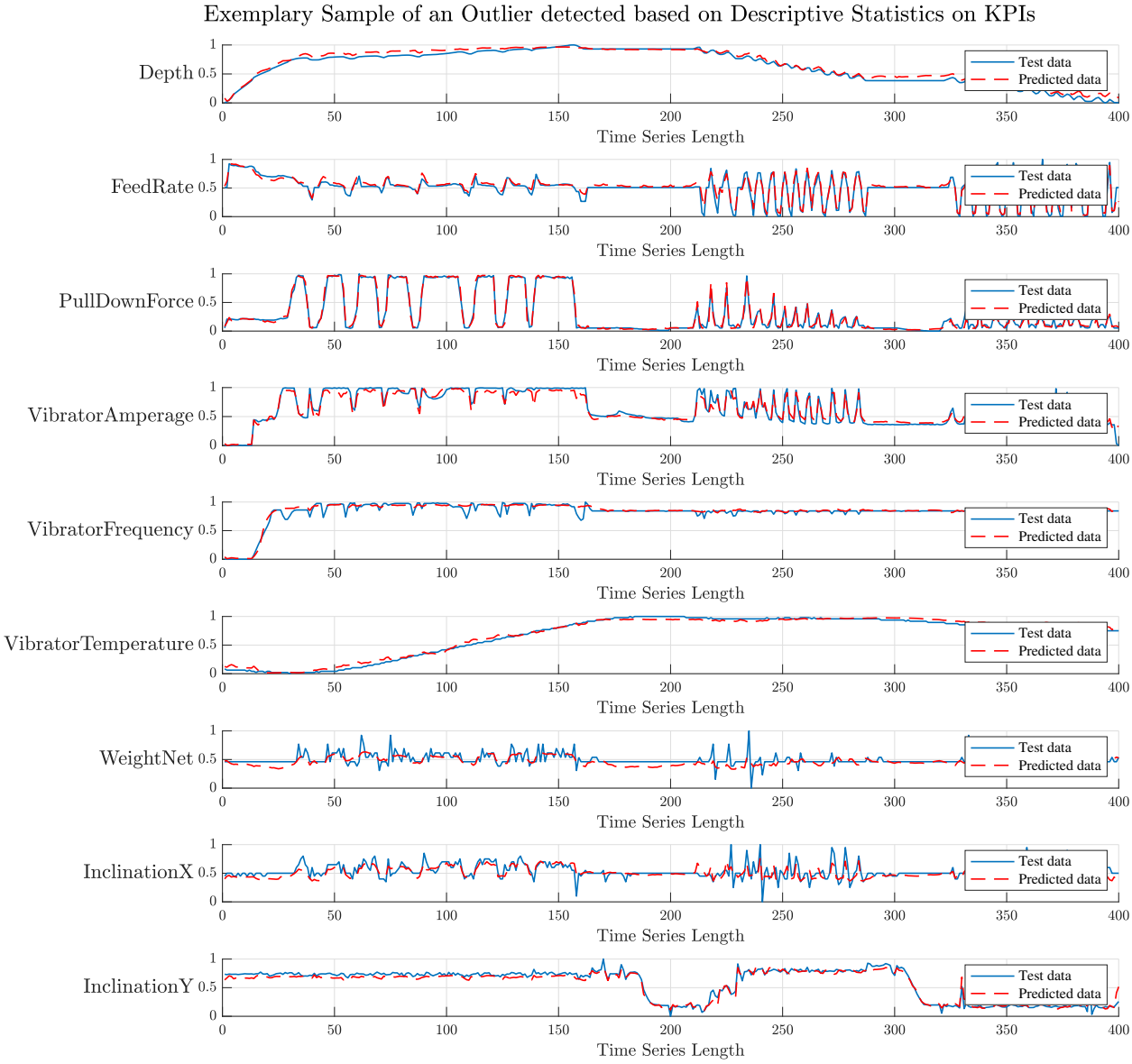


Fig. 5.17: This sample is not considered a POI by manual labelling, as there is neither a short pause, a long pause nor a roundtrip. There are several possible explanations on why this sample is an outlier. By looking up the definition of the KPIs and the unnormalized data the decision of the outlier method can be retraced. Assuming, that the statistically detected outliers are correct, them not being POIs contradicts the assumption that all outliers are POIs while not all POIs are outliers.

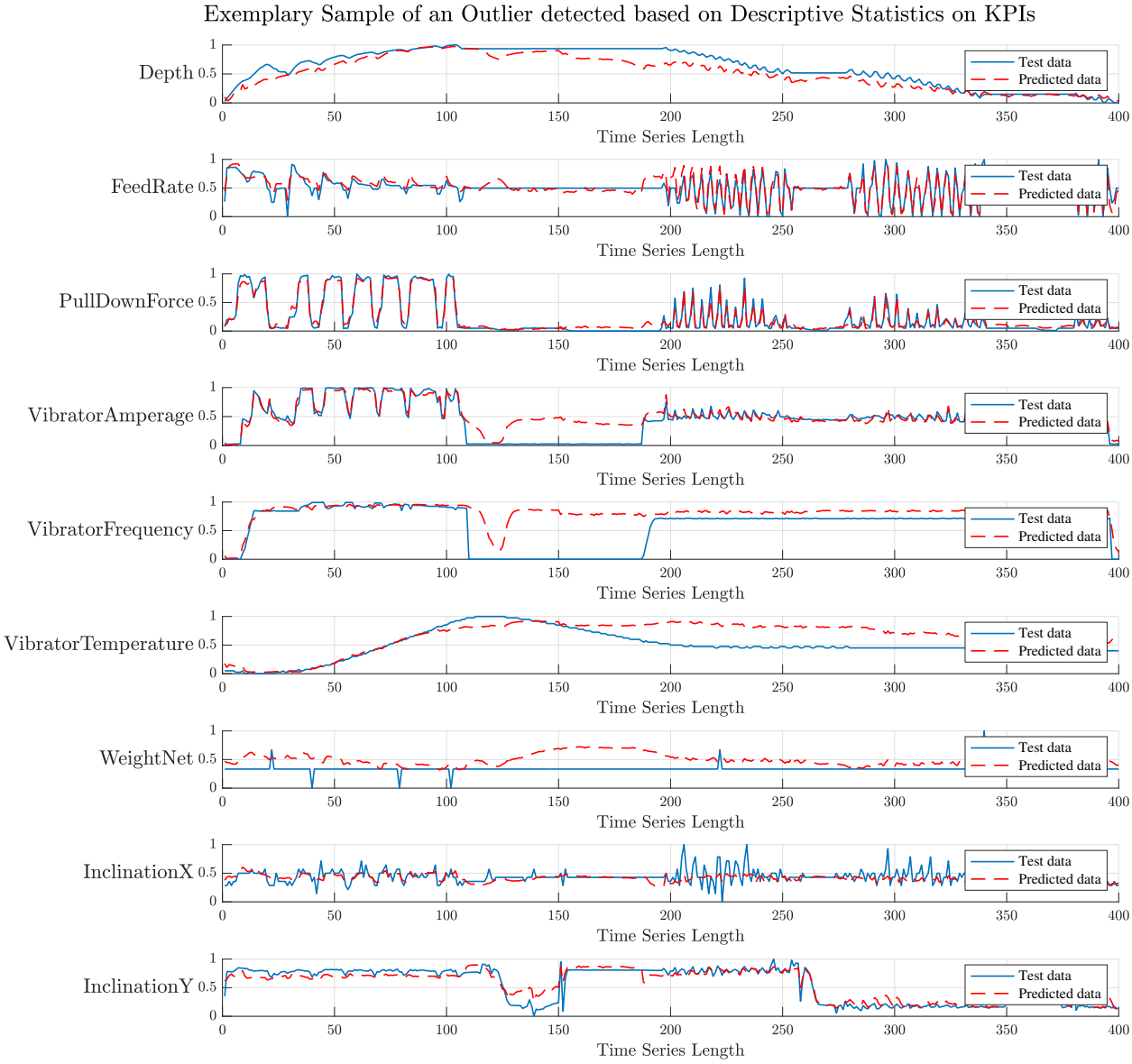


Fig. 5.18: Due to its break, this sample is considered a POI by manually labelling. The reconstruction error is too small to be considered an outlier by the IQR method. Also the ELBO loss is not considerably high enough in this sample. The statistical detection of outliers is a meaningful addition to the deep learning model.

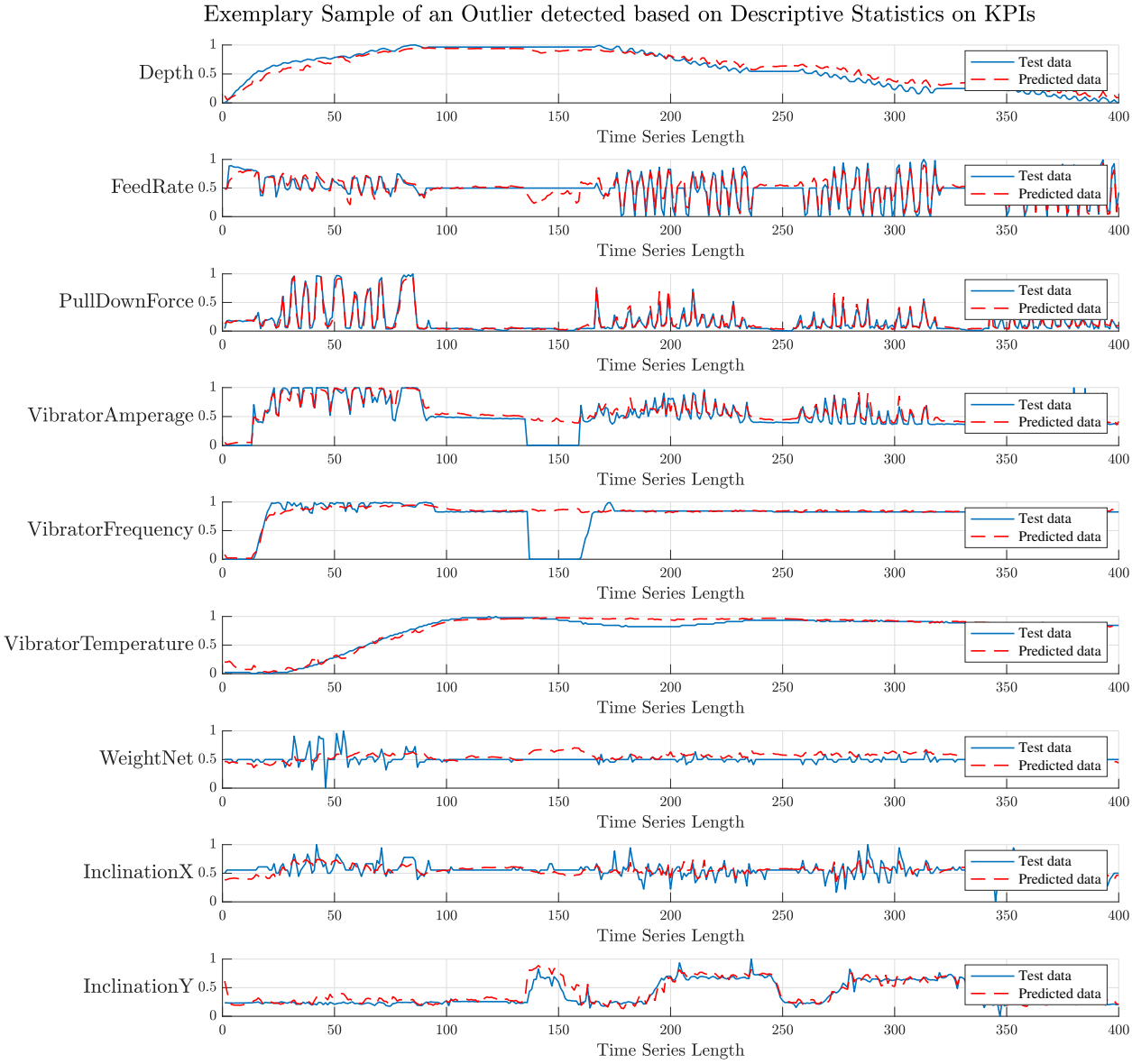


Fig. 5.19: This sample exhibits a rather short break, which is not recognized by the reconstruction loss. This sample is also not detected by the ELBO loss. This is another example for how the hybrid model performs better than the respective components of the model.

5.5 Evaluation of the Hybrid Deep Learning Model

This section presents the results obtained by testing and training the hybrid deep learning model. A parallel serial hybrid model is used for this approach. Its structure is depicted in 5.20. The results of the descriptive statistics component affect the input data of the deep learning part, because the data points considered anomalous by descriptive statistics are rejected from the training set. The parallel nature of the model derives from the final combination of the results delivered by both, the deep learning approach and the statistical approach.
The deep learning component delivers an outlier score, but no thresholds. When confronted with physical data, machine learning must be combined with statistical methods in order to deliver binary results. Thus, a hybrid approach is indispensable. To further improve the outcome, the results of statistical anomaly detection and deep learning are combined. This also leads to new nuances in the outlierness of samples, as samples that are detected as outliers by both approaches are considered more likely to be in fact anomalous than samples detected by one approach.

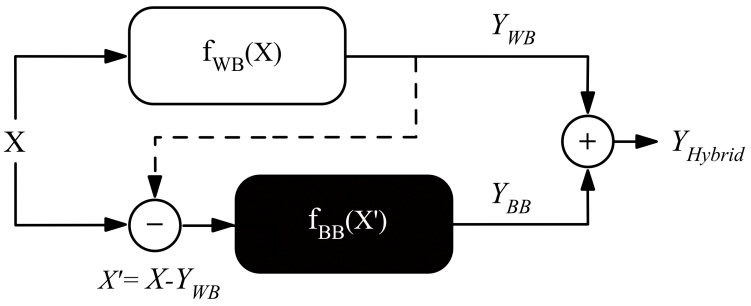


Fig. 5.20: The structure of the parallel serial model shows how the output of the statistical outlier detection Y_{WB} is substracted from the input X before it is fed into the neural network $f_{BB}(X')$. The dashed line demonstrates, that this is only the case for the training of the network. The test set is not X' but X.

The hybrid model is once used in an unsupervised manner and once trained with manually labelled data. As the manual labelled data does not guarantee a non-anomalous data set this approach is considered to be semi-supervised. The computed outputs are simply added together. This is expected to result in

1. more detected outliers in total and
2. a more accurate detection of outliers, when both, descriptive statistics and the deep learning model, consider them as anomalous.

5.5.1 Unsupervised Training

The hybrid model is expected to yield more accurate results if the LSTM-VAE is trained with non-anomalous data. One possible solution to automatically gain non-anomalous training data is the application of descriptive statistics on process KPIs of the whole dataset and reject the detected outliers in the training data set as shown in figure 5.20. No manual labelling is needed for the training and testing process.

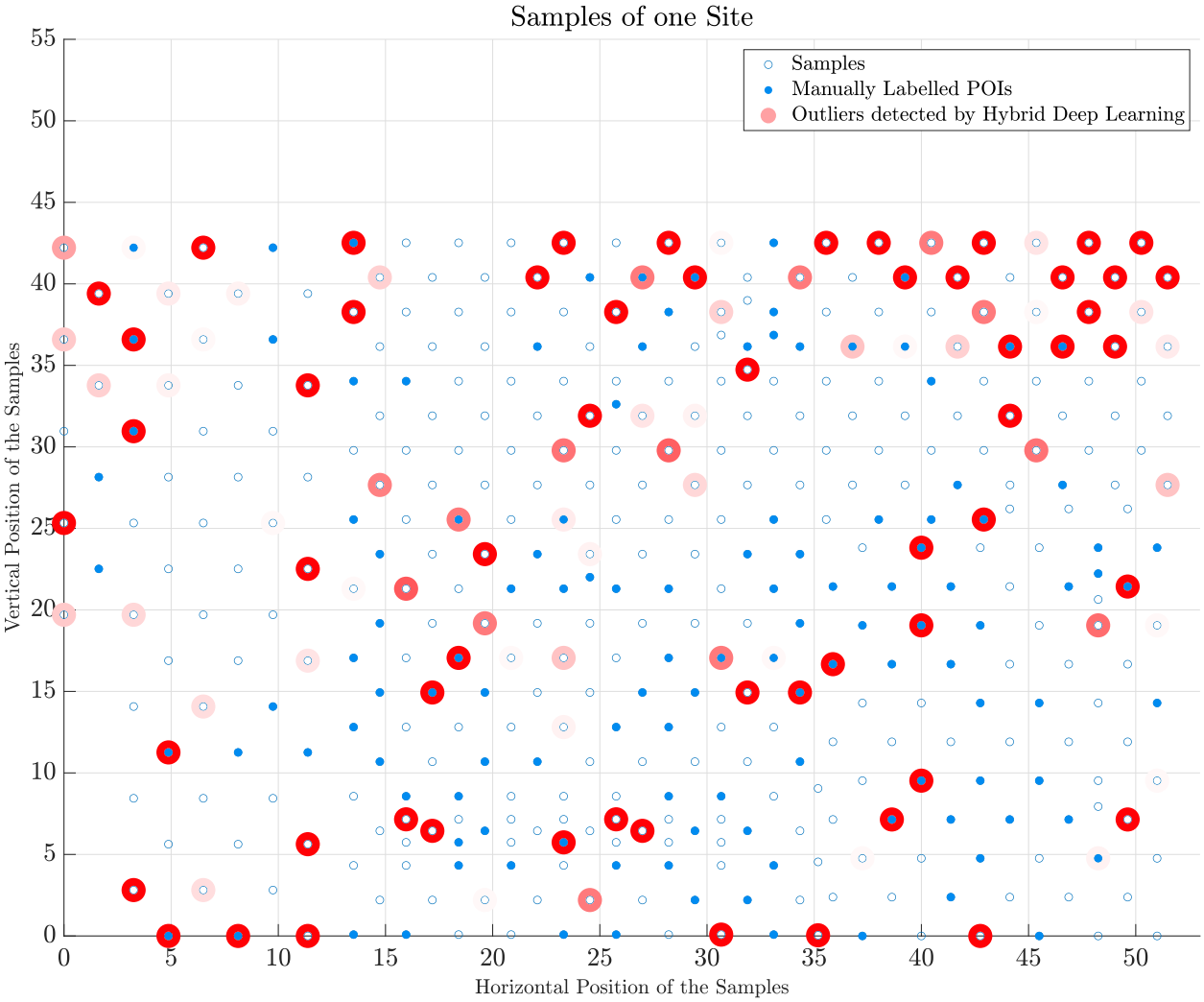


Fig. 5.21: The outliers from both, the LSTM-VAE after 20 runs and the descriptive statistics component are added together in order to get the outliers of the hybrid model. The less transparent the marker, the stronger is the outlierness of the sample. A high outlierness means a high probability that this sample is in fact anomalous. The manually labelled POIs are plotted to show, which detected outliers in fact look anomalous when looking at the corresponding time series.

In Figure 5.23 an exemplary run on a test data fold is depicted. The IQR of the summarized reconstruction error is calculated in order to define outliers. It can be seen that samples that are

considered as anomalous by descriptive statistics do not have a distinctively high reconstruction error and are therefore not detected by the deep learning algorithm.

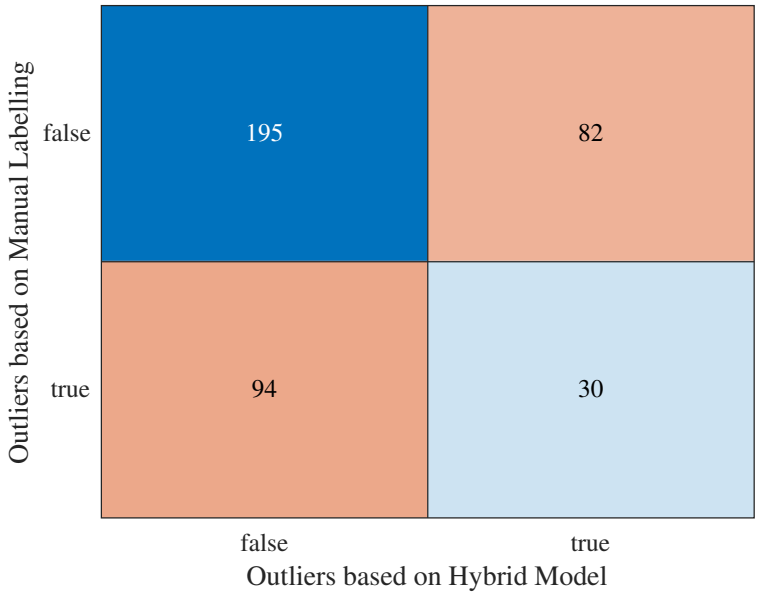


Fig. 5.22: The confusion matrix compares the outliers detected by the hybrid deep learning model with the manually labelled POIs. The manually labelled POI can not be considered as real outliers, as the labelling process is highly biased. However, the POIs indicate, whether or not the corresponding time-series data looks anomalous.

Fig. 5.23: This figure depicts the summation of the reconstruction error of the samples in one test data fold of an examplary run. On the basis of the summarized reconstruction error the IQR method is applied and the threshold for outliers calculated. Although the training data is not manually cleaned, the manually labelled POIs are plotted for comparison. Interestingly both, the machine learning part as well as the statistical component of the hybrid model, consider points to be outliers that are not POIs.

5.5.2 Semi-Supervised Training with Manually Labelled Data

In [7] and [8], non-anomalous data is used for training a neural network. However, both lack an explanation on how this non-anomalous data is obtained. Data originating from actuators and sensors in a physical system is very unlikely to be free from outliers. This implies that some prior knowledge about the underlying process is crucial in order to clean the training data from outliers. In the following section it is shown that using less anomalous training data in fact does deliver better results in the context of outlier detection.
The data is manually labelled by plotting the time series and checking it for anomalous behaviour. Possibly anomalous samples are marked as points of interest and rejected from the training dataset. According to its definition given in Section 3.4, the approach is now semi-supervised.
The application of descriptive statistics on KPIs yields outliers that are not considered to be potential outliers in the manual labelling process. Removing both, POIs and outliers detected by statistical analysis, is expected to further decrease the rate of possible outliers in the training data set. Training the network with solely ”perfect” data is expected to increase the difference between the reconstruction error of anomalous data points and non-anomalous data points, in order to achieve a more accurate detection of outliers.

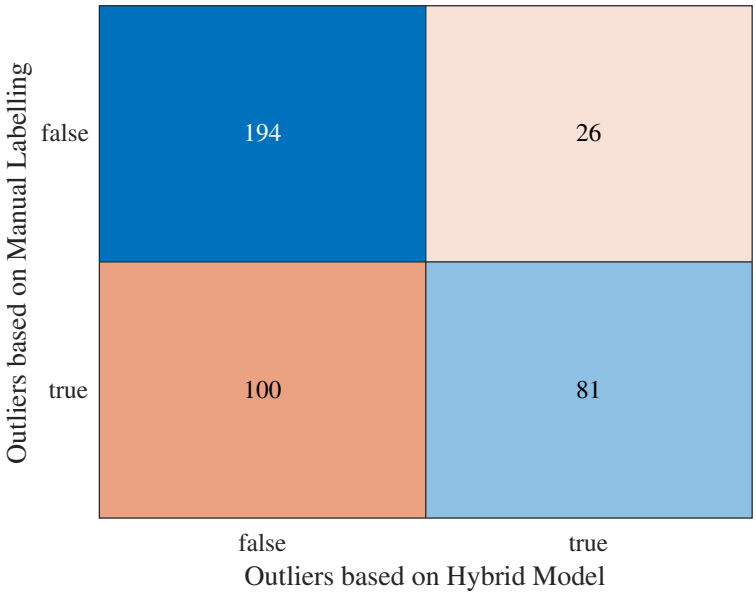


Fig. 5.24: The confusion matrix shows that the semi-supervised training of the hybrid deep learning model significantly increases the number of POIs that are detected by both, manual labelling and the hybrid model by more than a half. The amount of outliers detected by the hybrid model that are not confirmed by the manually labelled data drops to a third compared to the unsupervised approach. This is not necessarily an improvement, because wrong positives are not as critical as wrong negatives in the construction of fundaments for buildings.

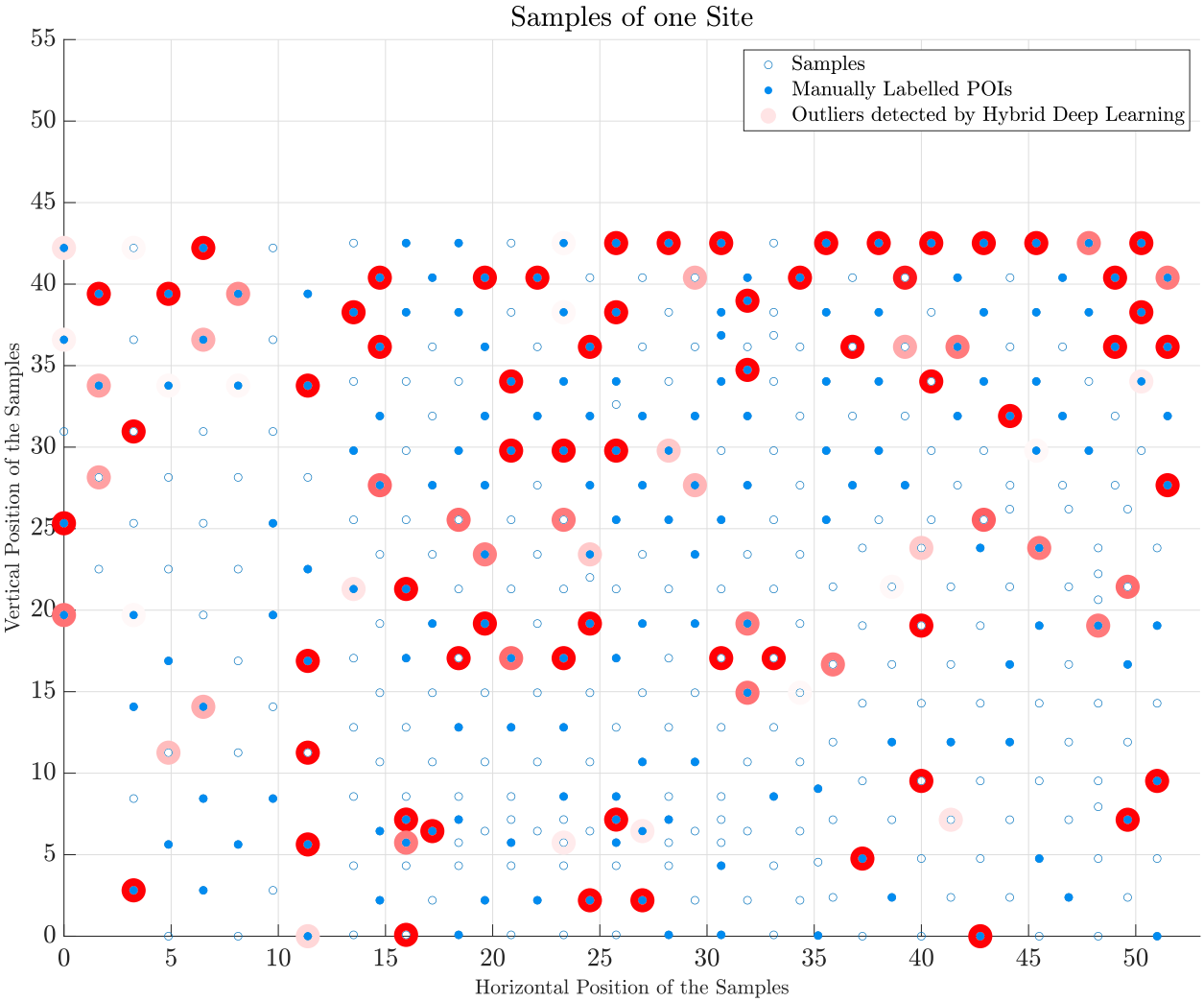


Fig. 5.25: Similar to 5.22 the outlierness of the samples is given by the transparency of the marker. Most detected samples are matching the outliers from the unsupervised approach.

Figure 5.26 depicts how the reconstruction error is more distinct in the semi-supervised approach than in the unsupervised approach, especially in contrast to 5.23. Thus, more outliers are detected.

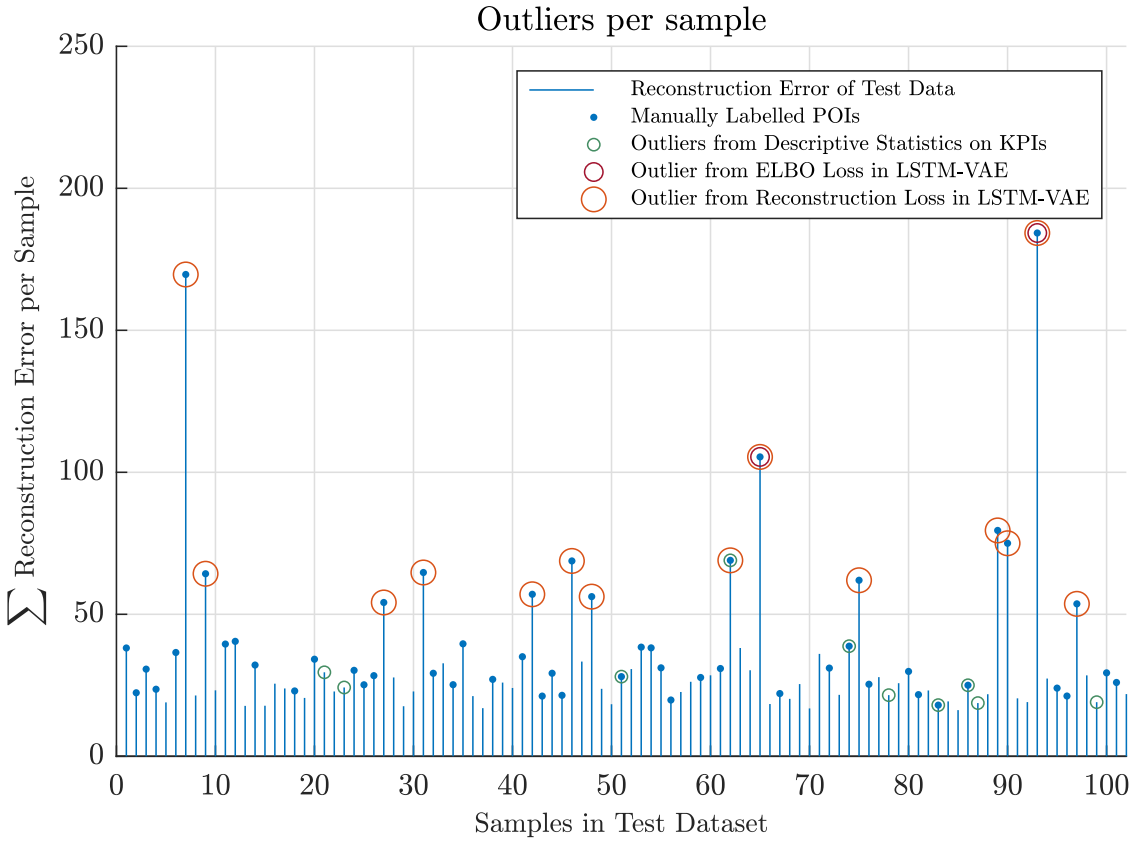


Fig. 5.26: The contrast between outliers and non-anomalous data is stronger than in the unsupervised approach. A possible explanation is that in the unsupervised approach the training data does still contain POIs and the neural network learns the patterns of the POIs. Thus, the network reconstructs them as normal samples and the contrast between POIs and normal samples diminishes. In contrast to the unsupervised approach, all outliers detected by the deep learning component are also manually labelled POIs. The overlap of outliers detected by statistical analysis and deep learning remains the same between the two approaches.

5.6 Limitations to Machine Learning

This section discusses the limitations to and problems with the application of deep learning for the detection of outliers. The performance of the deep neural network in the context of outlier detection varies due to two things: the training data set and the randomly initialized weights of the network at the beginning of the training.

Even if the model is trained several times with the exact same dataset the resulting networks perform very differently. There is no guarantee that the best performing model (containing the best weights for accurate outlier detection) out of the three that are trained in three-fold cross validation yields in fact the best results. Even if the best model is chosen out of 60 trained models, there is no guarantee there is not a possibly better one. There is no certainty whether or not the best possible model is found after the conducted 20 runs. The reason for this is the random weights initialization at the beginning of the training of the neural network. In Matlab the seed for the random control number generator can be changed. However, this would induce bias and possibly worsen the models' learning process [42].

It is very likely that 20 runs are not sufficient for a statistically meaningful comparison between the models trained with different data sets. The computation of 20 runs on a GPU server already costs 3-4 hours. Computing more runs in order to select the best trained model is a question of computation time and power. These things are not an issue in the statistical analysis of KPIs for outlier detection.

Another problem when using machine learning for outlier detection is interpretability. Due to the hidden nature of the layers in a deep learning network it is impossible to trace back the network's decisions. Especially when erronous behaviour occurs, for instance in the reconstruction of the input data, it is highly difficult to find the source of the problem. The intuitive solution to this would be to simply train the model again and hope for better results. This is a highly random and nontransparent procedure.

Due to the random nature of the training process in machine learning, results are difficult if not impossible to reproduce. Surely, one can save the trained network. However, taking parts of the network that work well and retrain parts that underperform is not an option. The lack of reproducability and the amount of random variables in the network cause the problem of determining a minimum amount of runs needed in order to gain statistical significance in the comparison of differently trained networks.

The results in the figures 5.22 and 5.24 show that the deep learning model performs better, the less anomalous the training data is. In order to gain less anomalous data, knowledge about the underlying physical process is necessary. This is the case for both methods, statistical analysis on KPIs and manual labelling of POIs. Thus, it is shown that the incorporation of physical knowledge improves the results of machine learning models as expected.

Due to the manual labelling being highly subjective there is a chance that some outliers are left undetected. In some applications this is not a big problem. For instance, it is not too problematic, if a web application suggests an unfitting song or video based on the already consumed content. However, in the construction of anchoring piles for the foundation of buildings, anomalous samples can cause a high risk to the safety of the final structure of the building. As false negatives may be invalid, the detection of outliers with deep learning is to be considered with caution to ensure the safety and quality of a physical process.

In summary, the main limitations in the application of deep learning for the detection of outliers are:

1. Randomness: Due to the amound of random variables in the network, there is no assurance that a trained network really is the best possible one. This also causes a lack of statistical significance when networks trained with different data sets are compared.
2. Reproducability: The results are not reproducable if the trained network is not saved. Using the same initial weights with different training data would introduce possibly disadvantageous bias. A trained network is inseperably connected to its training data set.
3. Interpretabilty: The detected outliers by the deep learning method are usually neither interpretable nor explainable.

Chapter 6
Conclusion and Future Avenues of Investigation

In general, the combination of statistical analysis and deep learning for the detection of outliers yields a more exhaustive and slightly more accurate detection rate than the particular components of the combined approach. The application of deep learning in the context of anomaly detection finds outliers, which are not found by traditional statistical methods. Especially those statistically undetected outliers that remain unchanged between different runs strongly support the hypothesis, that the inclusion of machine learning yields the detection of new types of outliers.

Even better results are achieved by defining possible outliers as points of interest prior to the training of the network. Rejecting those from the training data set delivers less anomalous training data. This process requires the inclusion of physical knowledge in the deep learning model.

Furthermore, calculating the IQR for both, the reconstruction error and the ELBO loss delivers more accurate results than solely using the reconstruction error. However, the accuracy of the results is measured by comparing the detected outliers with the manually labelled POIs. Thus, the evaluation of the accuracy of the detected anomalies has to be considered with caution.

Regarding future avenues of investigation, an evaluation of different descriptive statistical methods than the calculation of the IQR on the outlier score of the deep learning model might be valuable, as the outliers detected by the LSTM-VAE are also dependent on the used thresholding technique. Furthermore, the difference between the detection of outliers based on the ELBO loss and on the reconstruction error might be further explored, in order to get a better understanding of the differences of the respective detected outliers.

The fundamental assumption behind variational autoencoders is the presence of Gaussian distributions. However, neither the input data nor the reconstruction error delivered by the LSTM-VAE are normally distributed. This further complicates a statistically meaningful evaluation of the hybrid deep neural network. A very important topic for all future work is to investigate statistical significance when it comes to comparing and testing different machine learning models.

List of Figures

List of Tables

References

[1] Douglas M. Hawkins. *Identification of outliers / D.M. Hawkins*. English. Chapman and Hall London ; New York, 1980, x, 188 p. : ISBN: 041221900.

[2] Ane Blázquez-García et al. *A review on outlier/anomaly detection in time series data*. 2020. arXiv: 2002.04236 [cs.LG].

[3] Chih-Fong Tsai and Ming-Lun Chen. "Credit rating by hybrid machine learning techniques". In: *Appl. Soft Comput.* 10 (Mar. 2010), pp. 374–380. DOI: 10.1016/j.asoc.2009.08.003.

[4] Taeshik Shon and Jongsub Moon. "A hybrid machine learning approach to network anomaly detection". In: *Information Sciences* 177.18 (2007), pp. 3799–3821. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2007.03.025. URL: https://www.sciencedirect.com/science/article/pii/S0020025507001648.

[5] Mohammed Abdelrahim, Carlos Merlos, and Taehyung (George") Wang. "Hybrid Machine Learning Approaches: A Method to Improve Expected Output of Semi-structured Sequential Data". In: *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*. 2016, pp. 342–345. DOI: 10.1109/ICSC.2016.72.

[6] Elias Hagendorfer. "Knowledge Incorporation for Machine Learning in Condition Monitoring: a Survey". English. In: null ; Conference date: 16-09-2020 Through 18-09-2020. 2020.

[7] Shuyu Lin et al. "Anomaly Detection for Time Series Using VAE-LSTM Hybrid Model". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 4322–4326. DOI: 10.1109/ICASSP40776.2020.9053558.

[8] Daehyung Park, Yuuna Hoshi, and Charles C. Kemp. *A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder*. 2017. arXiv: 1711.00614 [cs.RO].

[9] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. "Outlier Detection: How to Threshold Outlier Scores?" In: *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing*. AIIPCC '19. Sanya, China: Association for Computing Machinery, 2019. ISBN: 9781450376334. DOI: 10.1145/3371425.3371427. URL: https://doi.org/10.1145/3371425.3371427.

[10] Stuart Hameroff and Roger Penrose. "Consciousness in the universe: A review of the 'Orch OR' theory". In: *Physics of Life Reviews* 11.1 (2014), pp. 39–78. ISSN: 1571-0645. DOI: https://doi.org/10.1016/j.plrev.2013.08.002. URL: https://www.sciencedirect.com/science/article/pii/S1571064513001188.

[11] Stefan Herdy. "Machine Learning in the Context of Time Series". MA thesis. Chair of Automation, University of Leoben, 2020.

[12] Anika Terbuch. “LSTM Hyperparameter Optimization: Impact of the Selection of Hyperparameters on Machine Learning Performance when applied to Time Series in Physical Systems”. MA thesis. Chair of Automation, University of Leoben, 2021.

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

[14] Sabyasachi Basu and Martin Meckesheimer. “Automatic outlier detection for time series: An application to sensor data”. In: *Knowl. Inf. Syst.* 11 (Feb. 2007), pp. 137–154. DOI: 10.1007/s10115-006-0026-6.

[15] Lynne Seymour, Peter J. Brockwell, Richard A. Davis. *Introduction to Time Series and Forecasting*. 3rd. Springer, 2016. ISBN: 978-3-319-29852-8.

[16] Hagendorfer Elias. “Evaluation of the Potential of Deep Learning for Manufacturing Process Analytics”. MA thesis. Chair of Automation, University of Leoben, 2018.

[17] Ji Zhang. “Advancements of Outlier Detection: A Survey”. In: *ICST Transactions on Scalable Information Systems* 13 (Feb. 2013), e2. DOI: 10.4108/trans.sis.2013.01-03.e2.

[18] Peter Goos, David Meintrup. *Statistics with JMP: Graphs, Descriptive Statistics, and Probability*. Wiley, 2015. ISBN: 9781119035701.

[19] *Documentation of the isOutlier-function in Matlab*. URL: https://www.mathworks.com/help/matlab/ref/isoutlier.html.

[20] Sergios Theodoridis. *Machine Learning: A Bayesian and optimization perspective*. 2nd. 2019. ISBN: 9788578110796.

[21] Christoph Molnar. *Interpretable Machine Learning*. Leanpub, 2019.

[22] Ian Goodfellow, Yoshua Bengio , Aaron Courville. *Deep Learning*. 2016. ISBN: 9780262035613.

[23] Kalidas Yeturu. *Handbook of Statistics*. Vol. 43. 2020. Chap. Machine Learning Algorithms, Applications and Practices in Data Science.

[24] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. 1st. The MIT Press, 2010. ISBN: 0262514125.

[25] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”. In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519. URL: http://dx.doi.org/10.1037/h0042519.

[26] Michel Denuit, Donatien Hainaut, and Julien Trufin. *Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions*. Jan. 2019. ISBN: 978-3-030-25826-9. DOI: 10.1007/978-3-030-25827-6.

[27] Thomas M. Mitchell. *Machine Learning*. 1st ed. USA: McGraw-Hill, Inc., 1997. ISBN: 0070428077.

[28] Klaus Greff et al. “LSTM: A Search Space Odyssey”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (Oct. 2017), pp. 2222–2232. ISSN: 2162-2388. DOI:

10.1109/tnnls.2016.2582924. URL: http://dx.doi.org/10.1109/TNNLS.2016.2582924.

[29] Sepp Hochreiter, Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 08997667.

[30] R. DiPietro, G. Hager. *Handbook of Medical Image Computing and Computer Assisted Intervention*. 2019. Chap. Deep learning: RNNs and LSTM.

[31] Walter Hugo Lopez Pinaya et al. *Machine Learning: Methods and Applications to Brain Disorders*. Elsevier, 2019. Chap. Autoencoders.

[32] Jinwon An and S. Cho. "Variational Autoencoder based Anomaly Detection using Reconstruction Probability". In: 2015.

[33] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392. ISSN: 1935-8245. DOI: 10.1561/2200000056. URL: http://dx.doi.org/10.1561/2200000056.

[34] Andrea Borghesi et al. "Anomaly Detection Using Autoencoders in High Performance Computing Systems". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), pp. 9428–9433. ISSN: 2159-5399. DOI: 10.1609/aaai.v33i01.33019428. URL: http://dx.doi.org/10.1609/aaai.v33i01.33019428.

[35] Stephen Odaibo. *Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function*. 2019. arXiv: 1907.08956 [cs.LG].

[36] *RSV Process Description by Keller*. URL: https://www.kellergrundbau.at/expertise/verfahren/ruettelstopfverdichtung-rsv.

[37] David Parmenter. *Key Performance Indicators: Developing, Implementing, and Using Winning KPIs*. 3rd. Wiley, 2015. ISBN: 9781119019848.

[38] Marcus Thorström. "Applying machine learning to key performance indicators". MA thesis. Department of Computer Science and Engineering, Chalmers University of Technology, University of Gothenburg, 2017.

[39] Harkiran Kaur and Aanchal Phutela. "Statistical Dimension Identification and Implementation for Student Progression System". In: *InternationalL Journal of Innovative Technology and Creative Engineering* 8 (2018). ISSN: 2045-8711.

[40] Yoshua Bengio and Yves Grandvalet. "No Unbiased Estimator of the Variance of K-Fold Cross-Validation". In: *J. Mach. Learn. Res.* 5 (Dec. 2004), pp. 1089–1105. ISSN: 1532-4435.

[41] Thomas G Dietterich, Eun Bae Kong. *Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms*. 1995.

[42] *Documentation of the rng-function in Matlab*. URL: https://www.mathworks.com/help/matlab/ref/rng.html.