

TDK-dolgozat

Andor Kiss

Anomaly detection of multivariate time series using machine learning

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

FACULTY OF INFORMATICS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE



Author:

Andor Kiss

Computer Science MSc

2nd year

Supervisors:

Dr. János Botzheim

Associate Professor,

Head of Department of Artificial

Intelligence

Máté Farkas

Industrial Supervisor,

Robert Bosch Kft.

Budapest, 2023

Abstract

Anomaly detection in multivariate time series is a challenging task due to the high dimensionality and complexity of the data. Traditional statistical methods are limited in their ability to handle multivariate data and may not perform well in the presence of outliers or non-Gaussian distributions. Machine learning algorithms, on the other hand, have shown great potential for anomaly detection in multivariate time series. In this thesis, machine learning methods are proposed for anomaly detection in multivariate time series. Popular machine learning methods such as Elliptic Envelope, Isolation Forest, Fuzzy Systems and deep learning techniques such as Convolutional and Fully Connected Autoencoders are compared. Experimental results on a real-world dataset show that machine learning methods can achieve better performance than traditional statistical methods. The trade-offs and limitations of methods are also discussed, along with explainability techniques, which are essential to make the learning algorithms predictions more trustworthy in an industrial scenario.

Contents

1	Introduction	3
2	Dataset	4
2.1	Simplifications	4
2.1.1	Anomalies	7
2.1.2	Signal analysis	10
2.1.3	Signal sampling rate	11
2.2	Data Exploration	11
3	Anomaly detection methods	13
3.1	Input dimensionality	13
3.2	Analyzed data	14
3.2.1	Baseline methods	14
3.2.2	Elliptic Envelope	15
3.2.3	Isolation Forest	16
3.2.4	Autoencoder Neural Network	17
3.2.5	Fuzzy systems	19
3.3	Raw signals	26
3.3.1	Autoencoder	26
3.3.2	Reconstruction error analysis	32
4	Evaluation	34
4.1	Evaluation metrics	34
4.1.1	Confusion matrix	34
4.1.2	Accuracy	35
4.1.3	Precision	35
4.1.4	Recall	36
4.1.5	F1-score	36

CONTENTS

4.2	Evaluation method	36
4.3	Evaluation results	37
4.3.1	Analyzed signals	38
4.3.2	Raw signals	41
4.3.3	Comparison of raw and analyzed signal results	43
5	Future work	44
6	Summary	46
A	Evaluation results	47
	Bibliograhpy	52
	List of Figures	55
	List of Tables	57

Chapter 1

Introduction

In an industrial scenario, when a product is in an early development phase, a large amount of manual labor is involved to make sure that the products don't contain any faults. To resolve this issue, software-based automation can be utilized, which can detect anything out of the ordinary, to make sure no faults will appear in the finished product.

In this paper Machine Learning based algorithms are proposed for diagnostic purposes. The goal is to design algorithms, which can identify and explain problems, and anomalies inside an already measured dataset, which can later be used for different datasets as well.

The main question of this research is "How can Machine Learning based solutions help us in detecting anomalies of a dataset containing several different time series signals?"

Chapter 2

Dataset

The dataset was generated by a durability test done in a cyclically repeating way, which simulated the endurance load of the product during its lifetime. Six similar products were measured at the same time, each generating approximately 500 signals.

2.1 Simplifications

6 * 500 signals would be too much to handle for an algorithm at first so simplifications had to be made. The process started by choosing the most important few signals. Note that the following plots will only show two specific signals, the current and the motor rotation, there are many more signals, which are not shown here. Also, on every plot the values of the time series are normalized to the interval $[-1, 1]$.

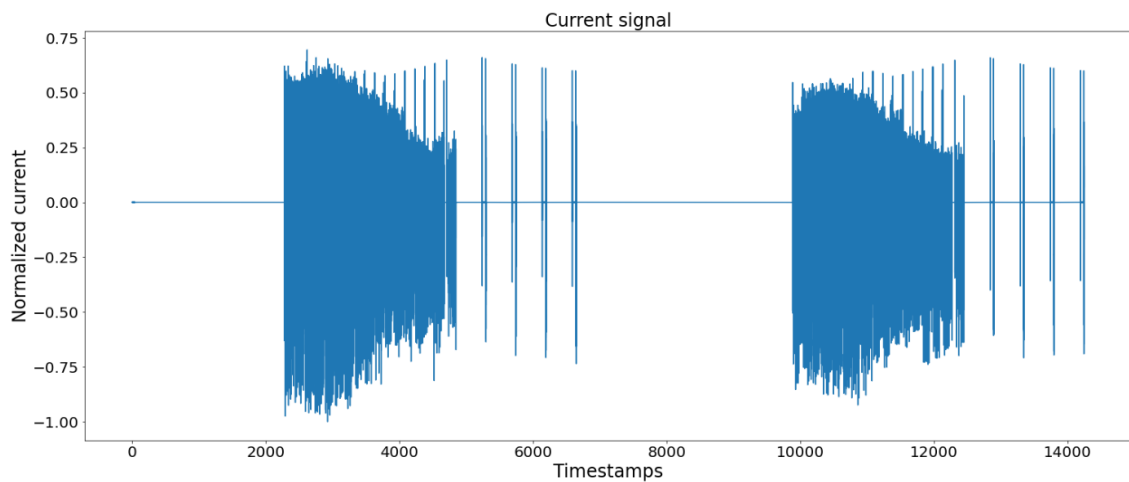


Figure 2.1: Current signal over multiple cycles

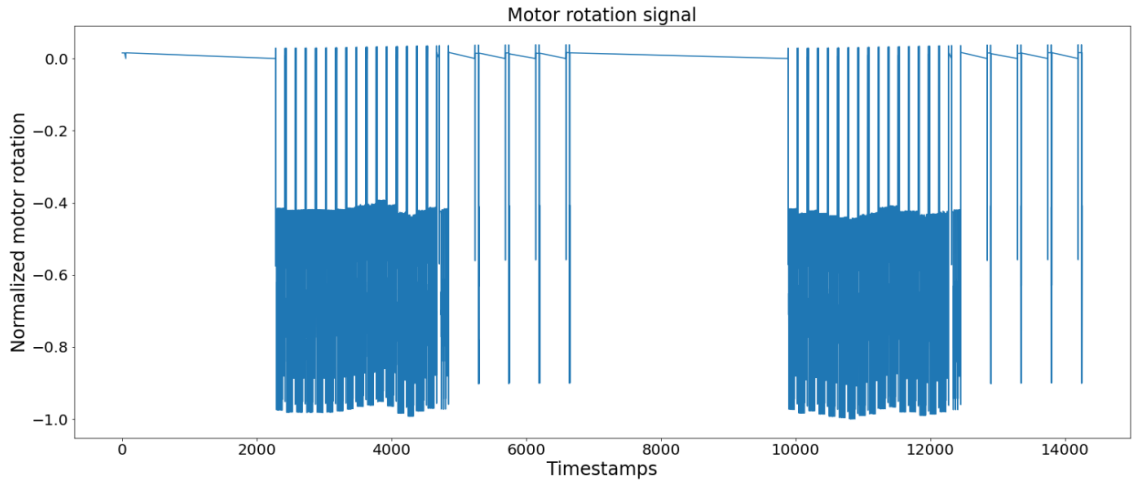


Figure 2.2: Motor rotation signal over multiple cycles

As Figures 2.1 2.2 show, the signals have cyclic repetition and parts where they have a value of zero, and the motor is at an idle state. Because of this nature, even more simplifications can be made: The most important operating modes can be extracted, where they have changing values, meaning that the motor is doing some work.

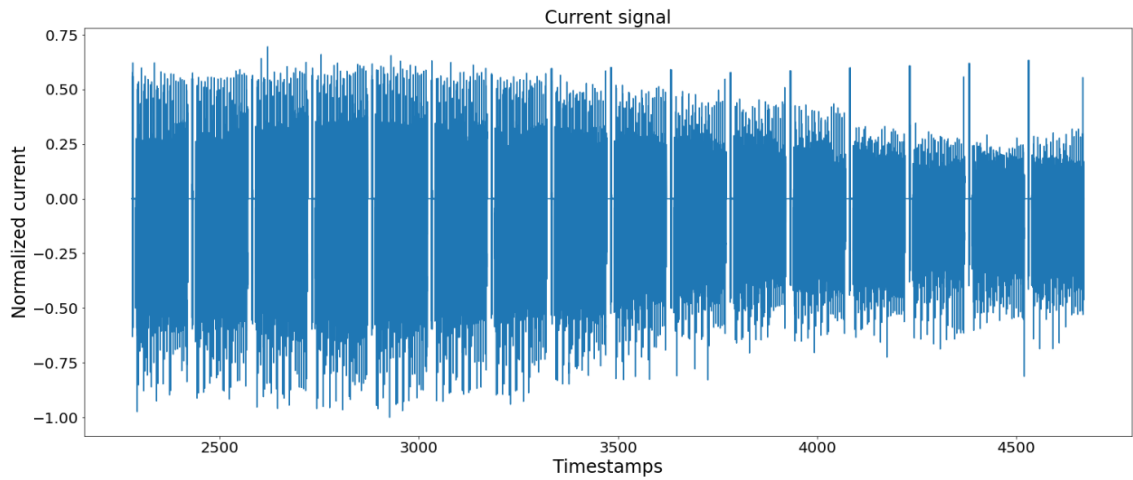


Figure 2.3: Current signal over one cycle in the correct operating mode

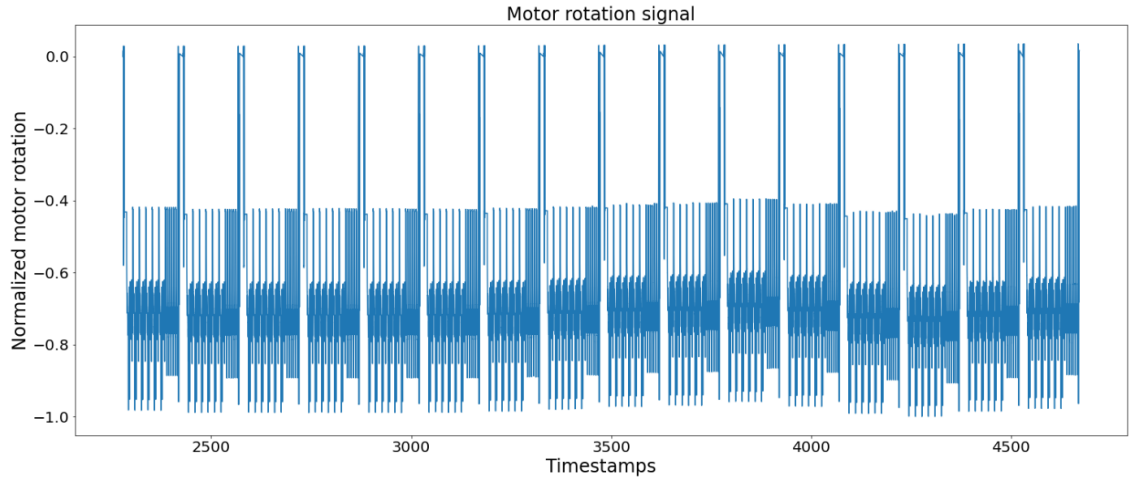


Figure 2.4: Motor rotation signal over one cycle in the correct operating mode

As Figures 2.3 2.4 show, inside a cycle and an operating mode, there is more cyclic behavior. These inner cycles can be segmented using control signals, which indicate where the motor should start an atomic cycle and when it should end one.

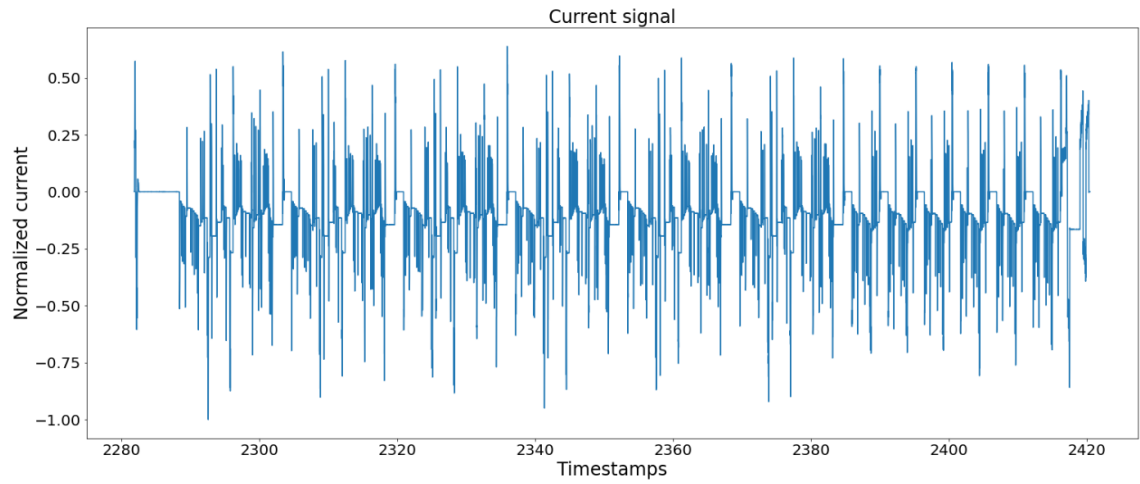


Figure 2.5: Current signal over one atomic cycle

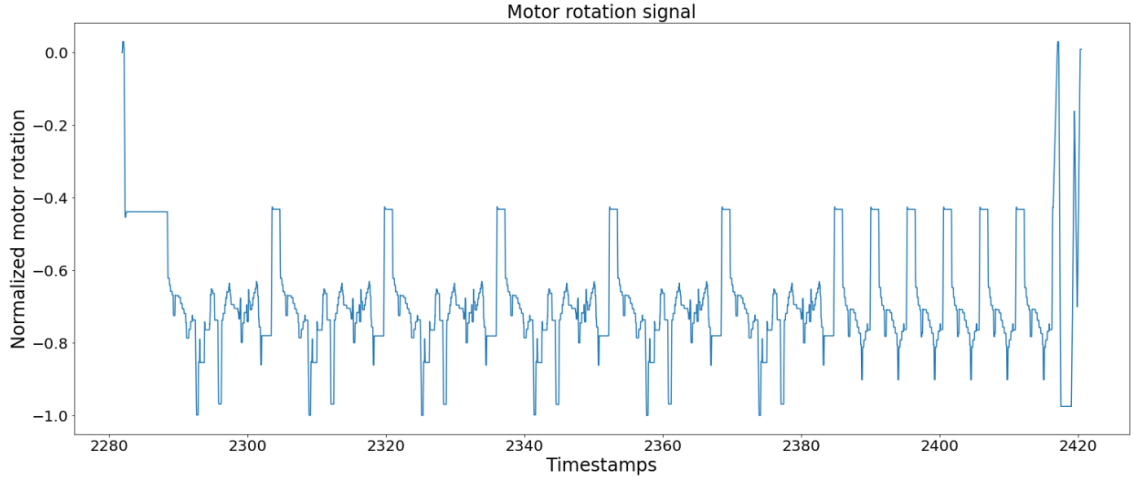


Figure 2.6: Motor rotation signal over one atomic cycle

From the internal atomic cycles shown on Figures 2.5 2.6, the behavior of the motors can be seen. These segmented signals could be used as a dataset for algorithms, which detect irregular, anomalous behavior in the cycles.

2.1.1 Anomalies

The dataset contains the results of a measurement, which does not indicate about every atomic cycle whether it is anomalous or not, meaning the dataset is unlabeled. However it contains a document indicating some known errors reported during the test. Using that document, some easily identifiable anomalies were found. These naturally appearing errors are not faults of the product itself, instead, those are features deliberately put in the measurement by the test engineer to ensure that the software-based anomaly detection will work.

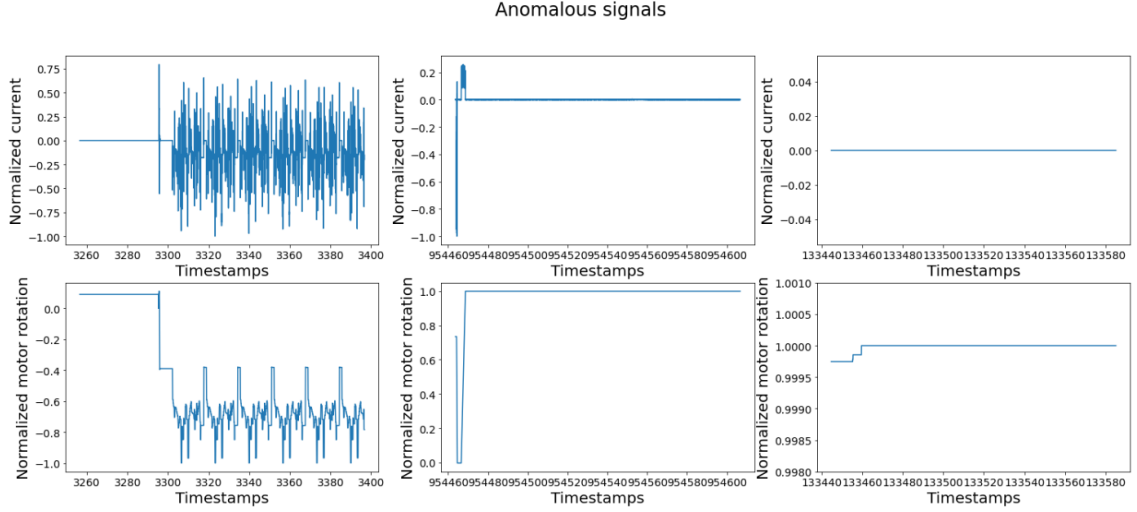


Figure 2.7: Several identified natural anomalies

Figure 2.7 showcases these anomalies on both the current and the motor rotation signal. The first ones are anomalies where the motor does not rotate at the beginning of the cycle, then later it starts working normally. The second ones are anomalies where the signals are constant for almost the whole cycle. The third column showcases anomalies where the motor has completely stopped for the cycle.

Since the goal of the project is not only to be able to find the anomalies, which are already found during testing, but to propose algorithms, which can find other types of anomalies, and previously undetected anomalies as well, using domain knowledge, artificial anomalies were generated, which emulate errors that can theoretically occur during the measurements, which are the following:

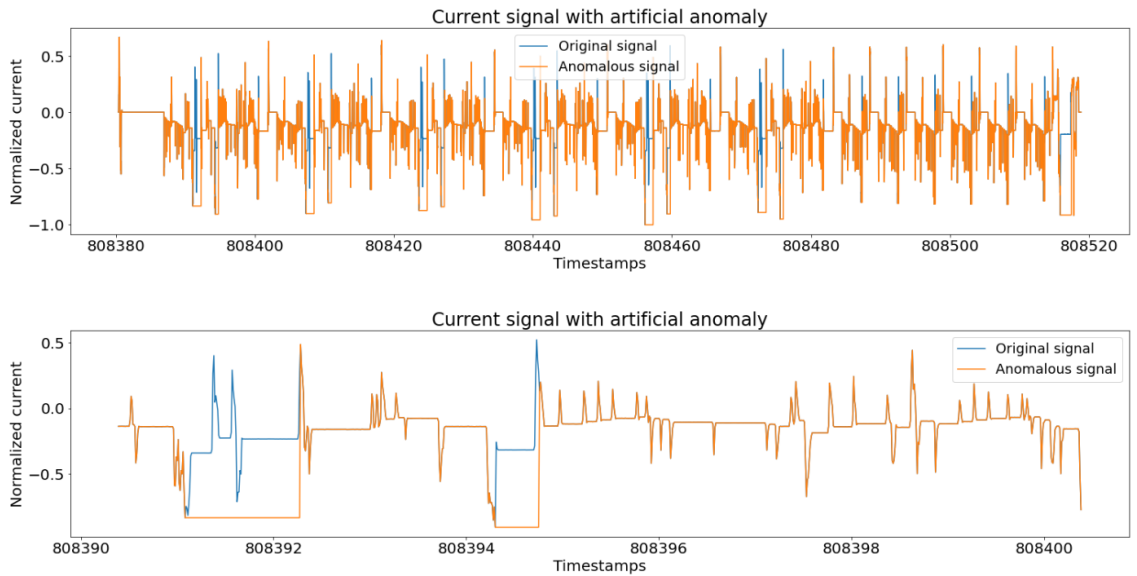


Figure 2.8: Current peak error

Figure 2.8 depicts an error with the current peaks. When the motor has to produce large amounts of torque, it has to take up high amount of current to be able to produce that. These are visible on the plots as peaks in the current signal. After this peak, in a normal case the signal quickly jumps back to a lower absolute value, however when an error like this occurs, the current does not jump back, rather it stays at the peaked position. The second subplot of Figure 2.8 is a zoomed in version, where a peak and a malfunction like this can be seen in more detail.

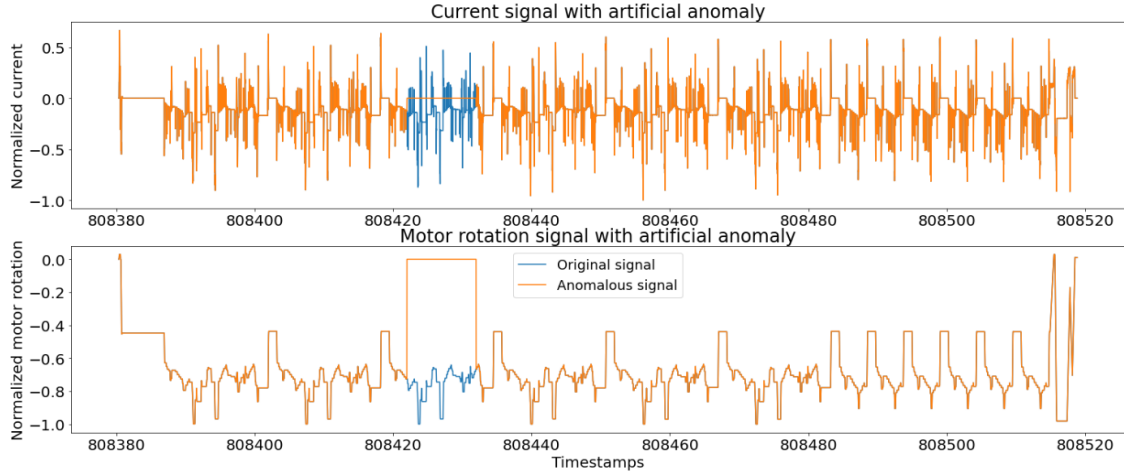


Figure 2.9: Motor stop error

Figure 2.9 shows an artificially generated error, which is similar to one that actually occurred in the dataset, when the motor was stopped for a short time during operation and every signal coming out of it was reported to be a constant zero value.

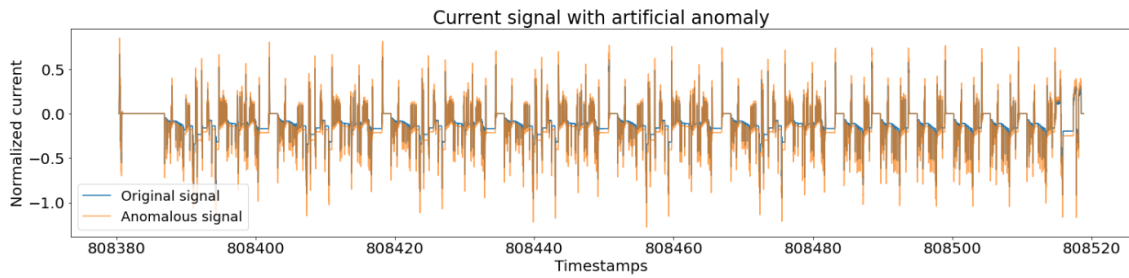


Figure 2.10: Increased current error

On Figure 2.10 an increase of current can be seen on the plot of the anomalous signal. This is due to the motor not having enough lubricant and the increased friction makes it take up approximately 1.3 times the current it normally would. On the plot the anomalous signal has some opacity so the consistently higher current value is easier to see.

2.1.2 Signal analysis

Usually time-series based models are complex and require a large amount of computational power. Before delving deep into the segmented signals by feeding the raw signals into these complex models, a simpler, baseline solution was required. For this baseline solution, the segmented signals were analyzed and each signal was represented by 9 statistical indicators, which were calculated from the signal:

- Mean
- Root mean square
- Standard deviation
- Minimum
- 25%
- Median
- 75%
- Maximum
- Signal length

From these values, a new dataset was constructed. With this approach, information is lost about the signals, but these indicators might describe them well enough for a model to be able to decide whether a signal is anomalous.

A more in depth analysis method was also used, which first cut the signal into n equal parts ($n=10$ was the actual number used), and each part was analysed using the 9 statistical indicators. This method needed more computation, and a more complex data pipeline, however retained more information about the signals, which might be useful for the anomaly detection methods.

2.1.3 Signal sampling rate

When working with raw signals, since most models work with input signals having equal lengths, the sampling rate of signals is an important parameter. In the dataset, different signals can have different sampling rates. Even signals with the same sampling rate can have segments with different lengths, which might be due to inaccuracies in the segmentation process based on the control signals. To guarantee an equal length of signals, before the segments are fed into the models, resampling is needed.

2.2 Data Exploration

The analysis of the segmented signals gave me an opportunity to visualize the data across the whole measurement.

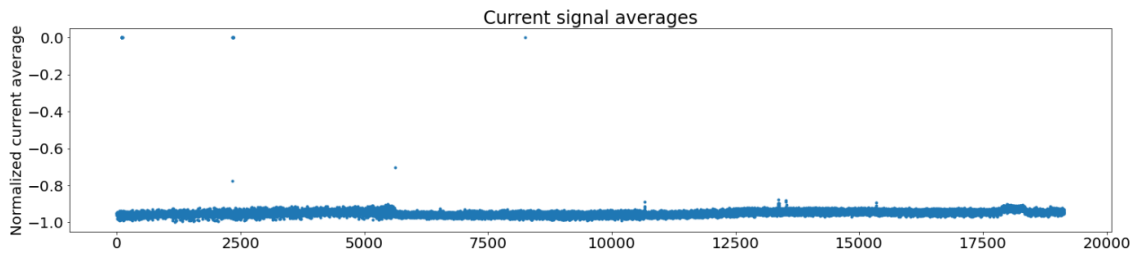


Figure 2.11: Current averages of a motor without anomalous behavior

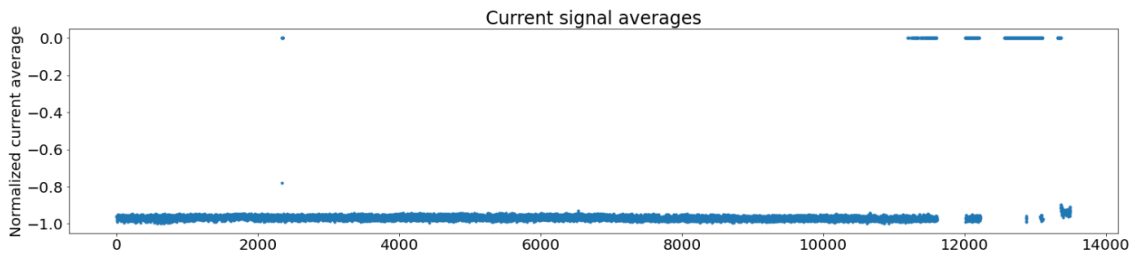


Figure 2.12: Current averages of a motor with anomalous behavior

Figure 2.11 shows the current averages of a motor, which functioned normally during the whole measurement, however, 2.12 shows the same values, but from a motor that started to break down at the end of the measurement, eventually stopping altogether. The amount of cycles with zero average current clearly indicates this anomaly. Also, an interesting fact about these plots lies on the x axis: When the motor stopped and did not generate any data, there were no signals to segment, hence there are fewer values there. Even simple plots like these can function as a simple, visual anomaly detection method. This data distribution can be visualized on histograms as well.

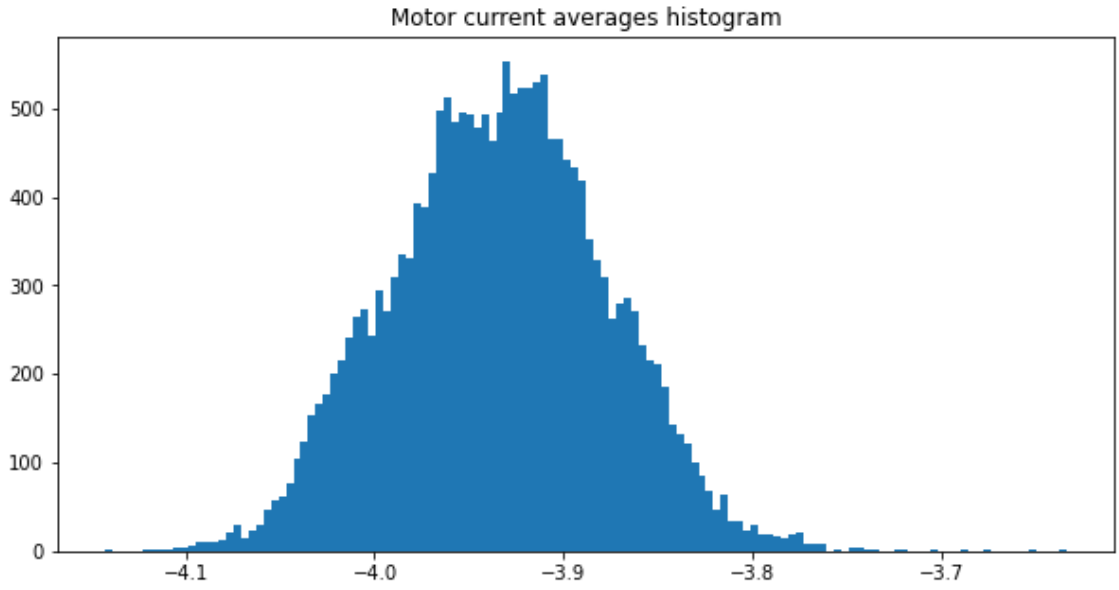


Figure 2.13: Current averages histogram of a motor without anomalous behaviour

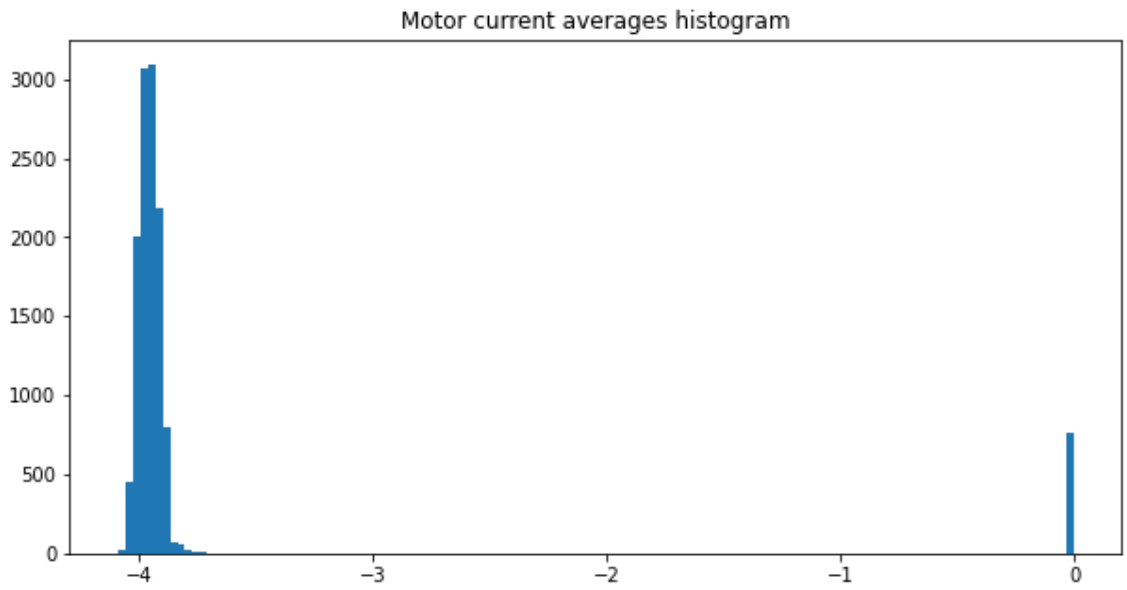


Figure 2.14: Current averages histogram of a motor with anomalous behaviour

On figure 2.14 the increased number of zero current average values is clearly indicated. On Figure 2.13, the current averages of a motor without anomalies resemble a Gaussian distribution.

Chapter 3

Anomaly detection methods

3.1 Input dimensionality

Before delving into the algorithms made for the two distinct most important input types, the analyzed data, and the raw signals, there are some more key questions regarding the inputs of the model.

1. How many signals should each model receive as input?
2. How detailed should each signal be?

Both questions can be applied to models trained on the analyzed and raw signals as well. If a model only receives one signal, it might be easier to learn it, for example with fewer parameters, however, it means that for each signal a new model has to be trained in the final anomaly detection system, and these models will not know about the cross-correlation, which might happen between some signals. The question about signal details refers to the further segmented signals described in Chapter 2.1.2 in the case of analyzed signals. In the case of raw signals, it refers to the sampling rate discussed in Chapter 2.1.3.

The numeric answers to these questions are essential hyperparameters of the models and are compared in the evaluation process.

3.2 Analyzed data

3.2.1 Baseline methods

The baseline methods involved some statistics as they measured how far a value is to the mean of the column. This approach is called the z-score method. To calculate the z-scores of an x datapoint, first, the mean (μ) and the standard deviation (σ) of the distribution are calculated, then the following equation shows the calculation of the z-score for an arbitrary x datapoint in the distribution:

$$z = \frac{x - \mu}{\sigma}$$

Z-score has the same major problem the mean has, which is sensitivity to outliers in the data. Usually when several outliers are present, which can certainly be the case in an anomaly (outlier) detection task, the median is used as a statistical indicator instead of the mean. Z-score has a variant, which uses the median ($\tilde{x}$), called the modified z-score. First, the dataset's mean absolute deviation (MAD) must be calculated.

$$MAD = \frac{1}{N} \sum_{i=1}^N |x_i - \mu|$$

$$z = 0.6745 \cdot \frac{x - \tilde{x}}{MAD}$$

Intuitively this method measures how many standard (or mean absolute) deviations a piece of data differs from a central value of the dataset. After the z-scores or modified z-scores are calculated for every datapoint, the anomalies are found by setting a threshold. If the z-score of a datapoint is above that threshold, it is considered an anomaly.

These methods were able to find the obvious anomalies, since those have the most deviation from the specified central point, however the strength of these methods is their low complexity. They are very easy to set up and run quickly, thus allowing a quick discovery of obvious anomalies in any similar dataset.

3.2.2 Elliptic Envelope

The first learning algorithm used in this project involved some more complicated statistics. The Elliptic Envelope algorithm assumes that the input data comes from a Gaussian distribution. While that is not guaranteed to be exactly the case here with every signal and every statistical indicator representing them, for example, the graphs showing the histograms of the current averages on Figure 2.13, 2.14 resemble a bell curve of a Gaussian distribution on the part where the electric current behaves normally. From this assumption, the shape of the data can be estimated and the outliers can be defined as datapoints, which lie far from the shape. This is the simple idea that is used by the baseline method in chapter 3.2.1 using only one signal. When the data is one or two-dimensional, the estimated Gaussian distribution can easily be plotted, and the outliers visualized. However with an n -dimensional dataset, these distances to the distribution have to be calculated. A widely used metric for this is the Mahalanobis distance:

$$d_M(\vec{x}, Q) = \sqrt{(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})}$$

This measures the distance between an $\vec{x}$ vector and a Q distribution, utilizing the mean vector $\vec{\mu}$ and the covariance matrix Σ calculated from the Gaussian. The problem with simply using this metric is that the $\vec{\mu}, \Sigma$ parameters of the distribution has to be calculated from the data, but the outliers distort the distribution, making this calculation lead to a wrong parameter set, which may result in erroneous estimation of anomalies. This is what the Elliptic Envelope tries to solve by using a robust covariance estimation algorithm called FAST-MCD (Minimum Covariance Determinant). [1] The algorithm searches for a set of datapoints, which are not outliers, based on their Mahalanobis distance and computes their covariance matrix. The determinant of this covariance matrix will be smaller than the determinant of the original dataset's covariance matrix. After that, a next, even smaller dataset is assembled by iteratively repeating this datapoint search based on the distance metric, until convergence is reached, where the determinant is either zero, or is unchanged compared to the previous one. When the final $\vec{\mu}, \Sigma$ parameters are calculated, the Mahalanobis distance of all datapoints are computed using those learned parameters. When the distance of a datapoint to the estimated distribution is large enough, it is considered an outlier.

The strength of this algorithm is that it works with multidimensional data, however, it has some weaknesses when the dimension is too large. The first sign of having too

much dimensions can be observed by looking at the speed of the algorithm. With more dimensions, it takes longer for it to converge, however when the dimensionality is too large, it can even break and not perform well. This was the case with the further segmented dataset I have used. The algorithm started to become slow and even performed much worse than expected on the test set, when the provided training dataset was the further segmented one with more than one signal's detailed statistical indicators.

3.2.3 Isolation Forest

The previously mentioned anomaly detection algorithms goal was to first find the normal part of the data, which does not contain anomalies and consider the datapoints further than usual to the normal data anomalies. The problem with these approaches is that it is optimized for finding the non-anomalous data instead of the outliers. This is where the Isolation Forest [2] algorithm differs from those. It is optimized for finding and isolating anomalous datapoints. The idea takes advantage of two properties of anomalies, which are that they are the minority in the dataset, having fewer instances than the regular data and they have attributes, which deviate from the norm. Here isolation means separating a datapoint from the rest of the dataset. This separation is done by partitioning attributes of the data, which is done by selecting a random attribute and a random split value between the minimum and maximum of that attribute in the dataset. A datapoint is separated from the rest when a partition can be defined this way, where it is the only datapoint. The most important idea of this algorithm is that an anomalous datapoint requires fewer partitioning steps. Partitioning steps are ordered into a tree structure called an Isolation Tree, where each node contains an attribute and a split value, which splits the dataset into two parts, which are stored in the two child nodes. When no more partitioning can be done, a node has no children. When this happens, the length of the node from the root of the tree is calculated, which will represent the number of partitioning steps required to isolate that datapoint.

The method is called Isolation Forest because similar to the supervised Random Forest [3] algorithm it uses an ensemble of trees. The forest is built by first taking random subsets of the data and then training isolation trees with the partitioning algorithm on each of the aforementioned subsets, finally adding the trees together to form an ensemble. The training complexity of the algorithm is $O(ts \log s)$, where t symbolizes the number of trees and s is the subsample size. Using more than one tree is helpful because the training

complexity depends more on the subsampling size than on the tree count, hence it helps reducing it.

The paper proposing the Isolation Forest also mentions using the algorithm in a high-dimensional scenario, where the accuracy of it is reduced, however due to the subsampling part, the training time stays relatively low.

3.2.4 Autoencoder Neural Network

Aside from datapoint isolating methods, reconstruction-based solutions were also tried for this anomaly detection problem. The basic idea was to create a compressed representation of the data and reconstruct the original dataset from that representation. The metric, which measures the success of these algorithms is the reconstruction error:

$$E_{reconstruction} = \frac{1}{N} \sum_{n=1}^N (x_n - x'_n)^2$$

Where x_n is the original datapoint and x'_n is the one reconstructed from a compressed form by an algorithm. The intuition behind these methods is that the algorithm will learn the underlying distribution of the correct part of the dataset, being able to reconstruct it well, but not learning it on the anomalous part, thus having a high reconstruction error on the outliers.

As a linear method, Principal Component Analysis (PCA) can be used. The PCA algorithm calculates a matrix for compression and reconstruction from the covariance matrix of the assumed Gaussian data distribution. This algorithm can work well with simple datasets, however, the disadvantage lies in its simplicity, as it only uses a linear transformation.

A more advanced algorithm was an autoencoder neural network. This neural network consists of an encoder part, which reduces the dimension of the input to the dimension of a defined bottleneck layer, utilizing non-linear activation functions between the layers. After the dimension reduction, the reconstruction is done by the decoder part of the network, which also uses non-linear activation functions. The implemented model consisted of several Fully Connected Layers.

Fully Connected Layer

A fully connected layer, also known as a dense layer, is a type of neural network layer that performs the matrix multiplication of the input vector and a set of weights, followed by the addition of a bias term and the application of an activation function. Mathematically, it can be represented as follows:

Let $\mathbf{x} \in \mathbb{R}^n$ be the input vector, $\mathbf{W} \in \mathbb{R}^{n \times m}$ be the weight matrix, $\mathbf{b} \in \mathbb{R}^m$ be the bias vector, and $f(\cdot)$ be the activation function. Then, the output of a fully connected layer can be expressed as:

$$\mathbf{y} = f(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})$$

where $\mathbf{y} \in \mathbb{R}^m$ is the output vector.

In this equation, the dot product of the weight matrix and the input vector performs the matrix multiplication, while the bias vector is added element-wise to the result. Finally, the activation function $f(\cdot)$ is applied element-wise to the result to produce the final output vector. The used activation functions were Rectified Linear Unit (ReLU) nonlinearities, which are defined using the following equation [4]:

$$ReLU(x) = \max(0, x)$$

Neural networks are often trained with the Stochastic Gradient Descent (SGD) [5] [6] optimization algorithm, and here a modified version of it called Adam [7] was used. The Adam algorithm adaptively changes the learning rate for a better fit during training, allowing the model to start learning quickly with a higher learning rate and make more precise steps during the later parts of the gradient descent with a lower amount. After every epoch, the model is used for prediction on a validation set, which consists of some easily identifiable anomalies, for example, signals where the current is zero during the whole cycle. Using this set, the training can be stopped early, similar to supervised learning scenarios, where early stopping is often used to abort the training cycle at an optimal point, where the complexity of the trained machine learning model handles the bias-variance tradeoff the best. Since this is an unsupervised learning task, the model outputs are not labels but reconstructed datapoints. In the validation step, the network reduces the dimensionality of the whole dataset and reconstructs it, allowing the calculation of the reconstruction error. After this error value is calculated for every datapoint, it is sorted in a descending way. The validation score is calculated by counting how many datapoints from the validation

set can be found in the first n points of this sorted list, where n is the number of documented anomalies. This prevents the model from overfitting on the whole dataset, because the training is only continued to a point where the reconstruction error is higher on anomalous datapoints than on normal ones. After the training has finished, a reconstruction error threshold is set using the validation dataset. The datapoint is considered anomalous when the error is larger than the threshold. During inference, the model reconstructs the inputs, calculates the reconstruction error, and then uses the earlier defined threshold to predict whether it is anomalous.

Figures 3.1 and 3.2 display these errors along with an example threshold, with points above it considered anomalies.

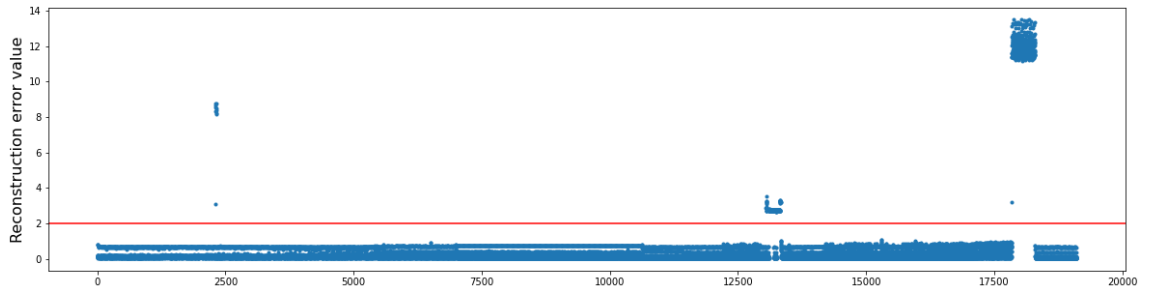


Figure 3.1: Reconstruction error over time

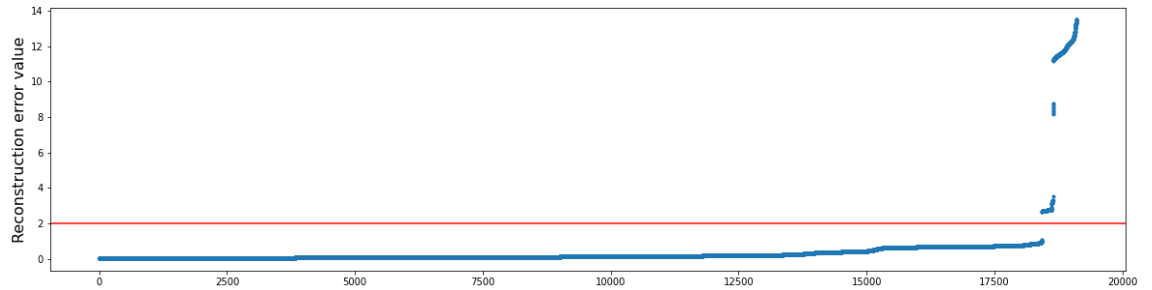


Figure 3.2: Sorted reconstruction error values

3.2.5 Fuzzy systems

Traditional set theory and computers use binary-valued boolean logic, where they represent something with a value of 0 or 1 depending on whether it is part of a set or not. However the world around us doesn't work that way, our observations are usually not black and white, but rather grey or it could be said: fuzzy. Fuzzy logic uses this property and defines membership functions, which indicate degrees of belonging to a set with a continuous value between zero and one with both ends included [8]. In fuzzy systems,

most of the time trapezoidal membership functions are used. To define these trapezoids, two properties of membership functions should be discussed.

Given a fuzzy set A and a membership function μ_A :

$$\text{core}(A) = \{x \in X | \mu_A(x) = 1\}$$

$$\text{support}(A) = \{x \in X | \mu_A(x) > 0\}$$

These core and support properties of fuzzy sets also yield sets of real numbers. To define a trapezoidal membership function four values are used:

$$a = \min_x \text{support}(A)$$

$$b = \min_x \text{core}(A)$$

$$c = \max_x \text{core}(A)$$

$$d = \max_x \text{support}(A)$$

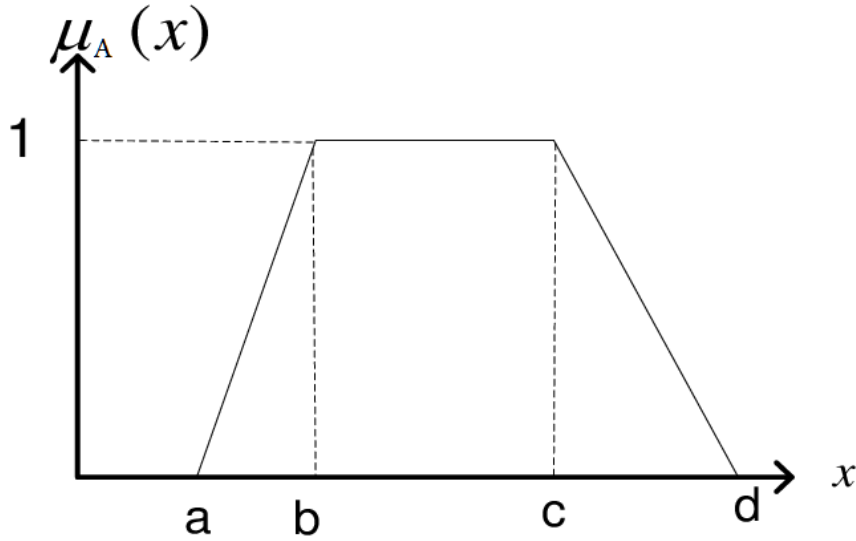


Figure 3.3: Trapezoidal membership function

Figure 3.3 illustrates a trapezoidal membership function defined by the four values: a, b, c, d . Fuzzy systems use these membership functions to construct rules. Every rule has antecedents and one consequent membership function. Antecedents correspond to the input of the rule, and consequents determine the output based on the antecedents. A fuzzy system is a union of several fuzzy rules. To make predictions with a fuzzy system, the Mamdani inference [9] process is used.

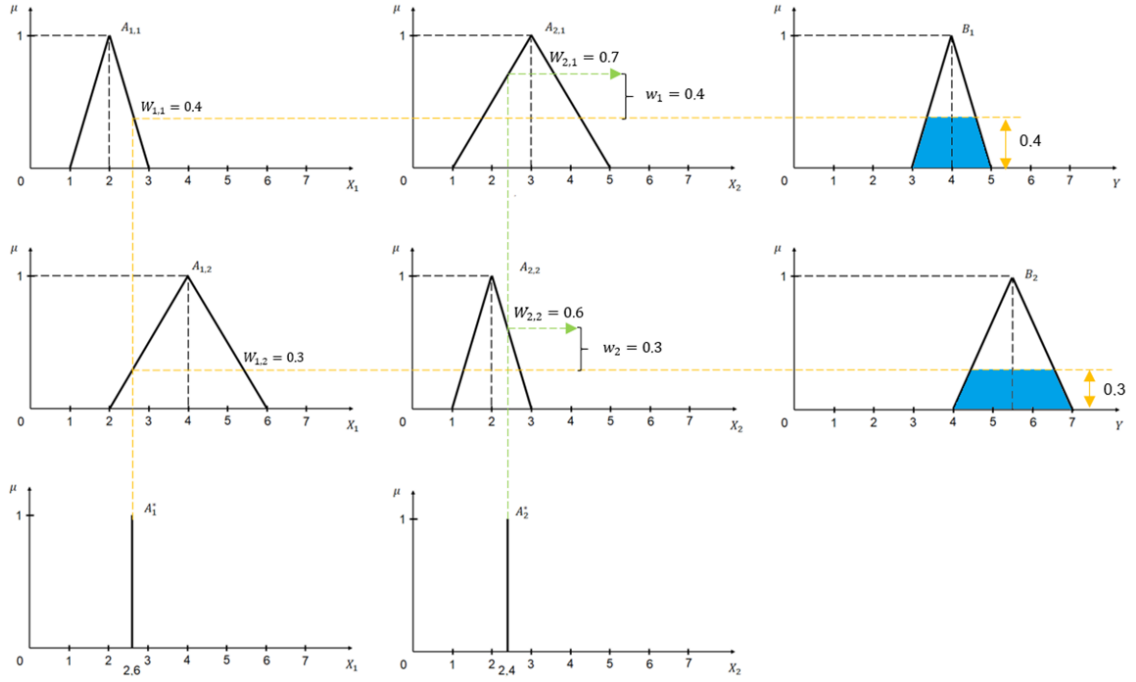


Figure 3.4: The Mamdani inference process

Figure 3.4 illustrates this inference process with a special kind of trapezoidal membership function, where the constraint $b = c$ makes the function a triangle. The system receives an input vector and for every rule, the value of each antecedent is calculated with the corresponding element of the input vector. Then the area of the consequent is calculated for every rule by having the height of the minimum value of the antecedent. The final prediction is made by using a defuzzification method on the union of every rule's consequent part's area. Since with every consequent calculation the most relevant antecedent, which has the minimum value for the input is known, this property can be used as an explanation of the system's choice. This makes fuzzy systems explainable methods, which is a highly sought-after property of machine learning models, especially in industrial conditions, where black box models might not receive enough trust from the customer. Creating fuzzy systems, which can predict whether a signal is anomalous, would require expert knowledge, which extends beyond this project's scope. Fortunately, fuzzy systems can be trained in a supervised manner with local optimization algorithms such as SGD or global optimizers for example genetic algorithms.

Genetic algorithms [10] are iterative stochastic search methods that computationally simulate the natural evolutionary process using the idea of survival of the fittest. In every iteration first, a population is generated and the individuals are ranked based on a fitness function. The best ones are selected, and their attributes (called genes) are crossed over

with each other and then mutated randomly. After that, the new individuals are inserted into the population and the next iteration begins.

Bacterial evolutionary algorithms are based on the idea of genetic algorithms, but instead of mimicking the evolutionary process of mammals, they are inspired by microbial evolution. Instead of the selection process, crossover and mutation, bacterial mutation and gene transfer are the steps of the iteration.

Global search methods can explore large search spaces, but generally have problems with fine-tuning, while local search methods can quickly converge to local minima. These techniques can be combined, which is called a memetic algorithm [11]. Bacterial evolutionary algorithms can be extended with a local search step between the bacterial mutation and the gene transfer step in every iteration, forming a bacterial memetic algorithm [12]. The algorithm for the local search can be any function optimization method, for example, SGD, which is popular in optimizing neural networks. The algorithm used in this implementation was the Levenberg-Marquardt method [13] [14] [15].

For this anomaly detection task, the fuzzy systems are trained by a bacterial evolutionary algorithm in a supervised manner. Since there is no available supervised dataset, one should be generated first by using one of the unsupervised methods mentioned earlier. Because the ground truth dataset for the training of the fuzzy system will depend on the performance of an already trained unsupervised model, only an evaluated, well-performing, trustable algorithm can be chosen for this prior task. The generated dataset will have two classes, with 0 representing anomalous, and 1 representing normal data-points.

After the supervised dataset has been created, the training can start. The first step of training any evolutionary algorithm is to initialize a population. In this case, each individual in the population is a fuzzy system, where the genes are the parameters of the membership functions. Usually, the initialization is done randomly, however in this case we can leverage the fact that fuzzy systems are in fact trainable expert systems. Expert systems [16] are early algorithms in the field of artificial intelligence, which emulated human experts' decision-making and reasoning processes using if-then-else constructs. In fuzzy systems, these if-then-else constructs appear in the form of antecedent and consequent membership functions. To initialize the population, rules were formulated about the signals:

- If the standard deviation of signals coming from the motor is 0, the motor has stopped, meaning it is an anomaly.

- If the current has both 0 mean and standard deviation, there is no current flowing through the motor during the cycle, meaning it has stopped again.
- If the motor rotation angle has too large standard deviation, the motor rotates too much, which is anomalous behavior.
- If a descriptor of a signal is further away from its mean by n times its standard deviation, it is an unusual, probably anomalous signal

Note that with a well chosen n value for the last rule, it can also describe the first three rules, which means one rule is sufficient to describe these observed properties of the signals. This rule also corresponds with the baseline model discussed in Chapter 3.2.1. For every column of the dataset, the trapezoidal membership functions of the antecedents are initialized as:

$$\begin{aligned} a &= \text{mean}(\text{column}) - 2\text{std}(\text{column}) \\ b &= \text{mean}(\text{column}) - \text{std}(\text{column}) \\ c &= \text{mean}(\text{column}) + \text{std}(\text{column}) \\ d &= \text{mean}(\text{column}) + 2\text{std}(\text{column}) \end{aligned}$$

The consequent as:

$$\begin{aligned} a &= 1 - \varepsilon \\ b &= 1 - \varepsilon \\ c &= 1 + \varepsilon \\ d &= 1 + \varepsilon \end{aligned}$$

Which will create a rectangular membership function with height 1 and width ε . This basically means that when all antecedents have a membership function value greater than zero, the consequent will be in close proximity to one, which means the system will predict that the input is not an outlier. However when any antecedent has a membership degree of zero, the consequent and the prediction will be zero, indicating that the datapoint is anomalous. Since during inference we know which antecedents have zero membership value, the explanation for the system's choice is known. During training, a fitness function is needed to determine which individuals are the best. For that, the individuals make predictions on the dataset based on their membership functions, and the training error is calculated using the Canberra distance metric, which is a well-suited choice for anomaly detection tasks [17].

$$e_n = \frac{|y_n - t_n|}{|y_n| + |t_n|}$$

This equation describes the Canberra distance for datapoint n of the dataset with y_n being the value predicted by the model and t_n being the ground truth.

The model is then trained using a fitness function and a bacterial evolutionary or a bacterial memetic algorithm. During training, some fuzzy rules changed, however, there were unchanged rules as well, which highlights the advantage of applying domain knowledge to the rule base.

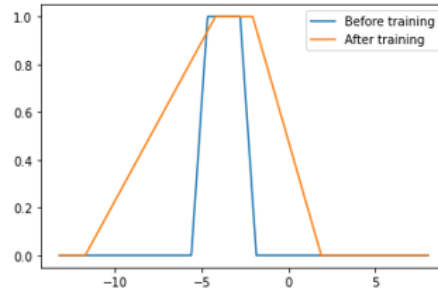


Figure 3.5: Changed rule

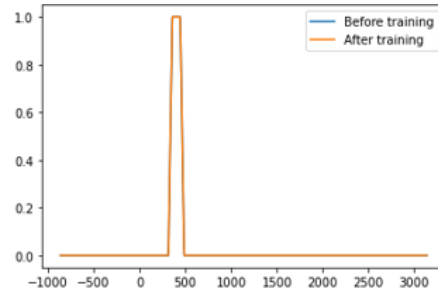


Figure 3.6: Unchanged rule

Figures 3.5 and 3.6 highlight one signal from each of the two categories.

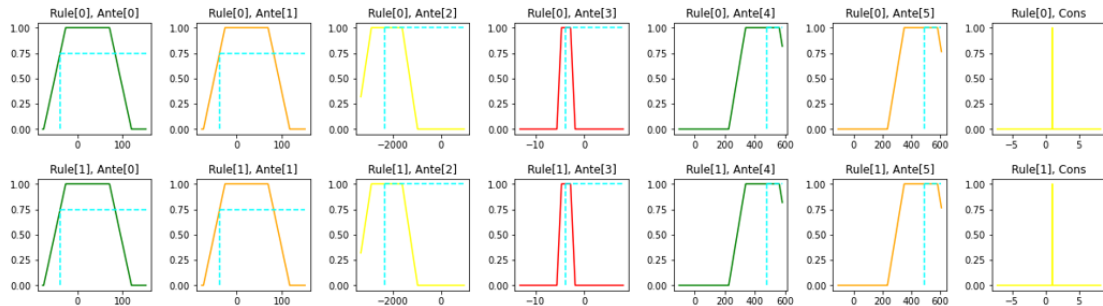


Figure 3.7: The rule base before training

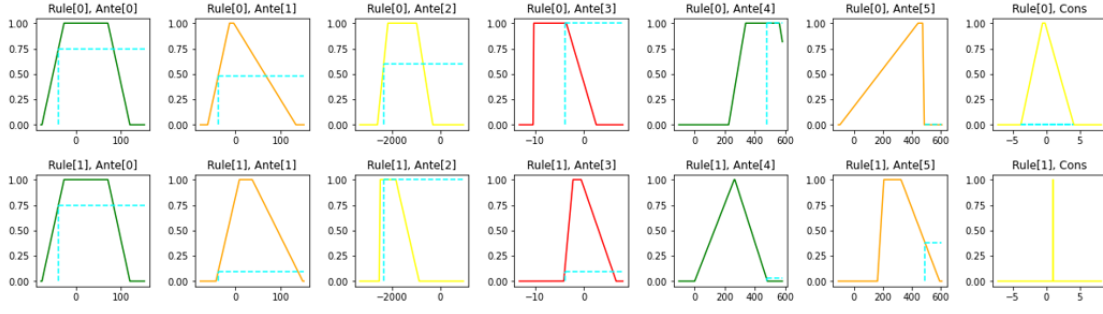


Figure 3.8: The rule base after training

Figures 3.7 and 3.8 show how the whole rule base changed. When the training has finished the final fuzzy system will be the best individual of the population, according to the fitness function.

The use of a fuzzy system on top of a pre-trained model has the advantage of being explainable and incorporating the knowledge of the baseline model with additional refinement through training. However, this also adds to the training overhead as it requires training an additional model. The training process using a bacterial evolutionary algorithm took around 7 minutes, while using a bacterial memetic algorithm with an extra local search step took approximately 45 minutes, which is a considerable increase in time. It should also be noted that this is a supervised algorithm built upon an unsupervised one, which could either lead to a decrease in error or pose as a disadvantage, depending on the performance compared to the original unsupervised algorithm. If the results of the fuzzy system are better, it suggests that it is capable of rectifying mistakes made by the unsupervised algorithm.

3.3 Raw signals

3.3.1 Autoencoder

The analysis of raw signals was performed using various autoencoder neural network architectures. The pipeline consisted of the following steps: first, each segmented signal was uniformly sampled, then fed into the network for dimension reduction and reconstruction. Finally, the reconstruction error was computed by comparing the input signal to the reconstructed signal using Equation 3.2.4. As neural networks are modular in nature, it is essential to first define the layers, which act as the foundational components. In addition to the fully connected layers already defined in Chapter 3.2.4, the following layers were used:

Convolutional Layer

A 1D convolutional layer is a type of operation that is commonly used in deep learning for processing sequences of data, such as time series, audio signals, and sequences of words in natural language processing. The core idea behind a 1D convolutional layer is to apply a linear operation to a sliding window of input data, so as to learn a compact representation of the local patterns in the data.

A 1D convolutional layer can be defined mathematically as follows:

$$y_i = \sum_j x_{i+j} \cdot k_j$$

Here, y_i is the output value at time step i , x_{i+j} is the input value at time step $i + j$, and k_j is the weight associated with the filter. The filter is a 1D vector of weights that are learned by the network during training, and the goal of the convolutional layer is to apply this filter to the input data in a sliding window fashion.

In a deep learning context, multiple 1D convolutional layers can be stacked on top of each other, with the output of one layer being used as the input to the next. This allows the network to learn increasingly complex representations of the data, and enables it to detect patterns that span multiple time steps. Additionally, non-linear activation functions, such as ReLU, can be applied after each convolutional layer to introduce non-linearity into the model and to improve its ability to model complex patterns in the data.

Pooling Layer

A 1D pooling layer in a neural network is used for down-sampling, reducing the spatial dimensions of the feature maps. It operates on 1-dimensional data such as sequences or signals and performs pooling operation on these sequences.

Given an input feature map of size (w_0, c) , where w_0 is the width, and c is the number of channels, the 1D pooling layer will take a kernel of size (k_w) , where k_w is the width of the kernel. The kernel is moved along the width of the feature map, and at each position, the maximum (or average, depending on the type of pooling operation) of the values within the kernel region is taken, resulting in a reduced feature map of size (w_1, c) , where $w_1 = \lfloor \frac{w_0}{k_w} \rfloor$.

1D pooling is used in Convolutional Neural Networks (CNNs) to reduce the computational cost and in the case of an autoencoder to help the model learn a representative bottleneck by reducing the spatial size of the input and retaining the most important information.

Upsampling Layer

A 1D upsampling layer in a neural network is a layer that increases the spatial resolution or dimension of a 1D feature map or tensor. The operation involves replicating the values of each element in the input tensor and placing them in consecutive locations in the output tensor, effectively increasing the number of elements by a factor that is determined by the upsampling rate. For instance, if the upsampling rate is 2, the length of the input tensor is increased by a factor of 2 in the output tensor. In convolutional autoencoders, upsampling is used to reconstruct the original signal in the decoder by increasing the dimension of the data coming from the bottleneck, serving as an inverse operation of the pooling.

Dropout

The dropout layer is a regularization technique used to prevent overfitting in deep learning models. It works by randomly "dropping out" or zeroing out a specified percentage of neurons during each training iteration. The idea is to prevent the model from relying too heavily on any one feature, as the neurons that are dropped out change randomly with each iteration. This helps to ensure that the model is not overfitting to the training data, and generalizes better to new, unseen data.

Mathematically, the dropout layer can be represented as:

$$y = x \odot m$$

where x is the input to the dropout layer, m is a binary mask with the same shape as x , and $\odot$ denotes element-wise multiplication. The elements of m are either 1 or 0, and are generated using a specified probability distribution, such as a Bernoulli distribution, with a specified dropout rate. During inference, all neurons are kept, so the output of the dropout layer is simply the input.

From these layers, the following neural networks were constructed:

Fully Connected Network

The fully connected autoencoder applied to raw signals is similar to the one described in Chapter 3.2.4, but with a larger parameter space. The input signals are first flattened into a one-dimensional format and then processed by the encoder and decoder, both of which consist of a series of fully connected layers. This model has the largest number of parameters among the models discussed in this paper because the neurons in the first layer are connected to every point in the input signal. This results in high memory requirements during training, but the large number of parameters also makes the model well-suited to the signals.

Fully Convolutional Network

The fully convolutional network is composed of convolutional layers in the encoder section, accompanied by pooling layers to reduce the dimensions. In the decoder section, the convolutional layers are supported by upsampling layers that restore the original dimensionality of the signal. A skip-connection approach, similar to the U-Net architecture used for image segmentation [18], was also explored. This approach employs convolutions to identify local patterns in signals using a sliding window, as opposed to assigning a parameter to every point in the signal, as is the case with the fully connected network.

Hybrid approach

The hybrid approach integrates the locality-focused method of the convolutional autoencoder with fully connected layers. In the encoder, the signal's dimensions are reduced

through convolution and pooling operations, then flattened, and at the bottleneck, the compressed representation is learned through fully connected layers. Once the fully connected layers have learned the underlying distribution of the signals, the original signal is reconstructed using convolutional and upsampling layers. In this hybrid approach, a Large and a Small model were both implemented with the goal of having the same, or not significantly worse results on the task with significantly fewer trainable parameters. This is similar to the idea behind the ALBERT [19] model, which is a transformer [20] model designed to perform NLP tasks with results not worse than the BERT [21] model.

The networks were trained in a manner similar to the autoencoder described in Chapter 3.2.4, but on raw signals instead of their statistical descriptors. The validation procedure and early stopping were similar to those described in Chapter 3.2.4. The Adam algorithm [7] was used for optimization. After the network reconstructed a signal, the reconstruction error was calculated for each point in the signal and then summed. Figures 3.9 and 3.10 display these summed errors along with an example threshold, which identifies points above the threshold as anomalies.

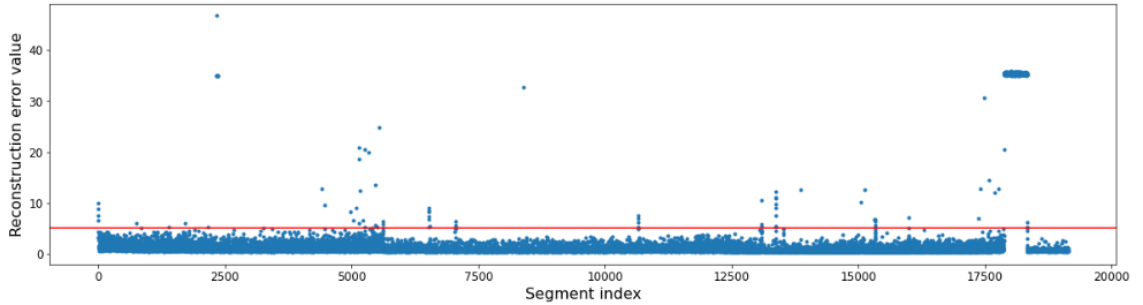


Figure 3.9: Reconstruction error over time

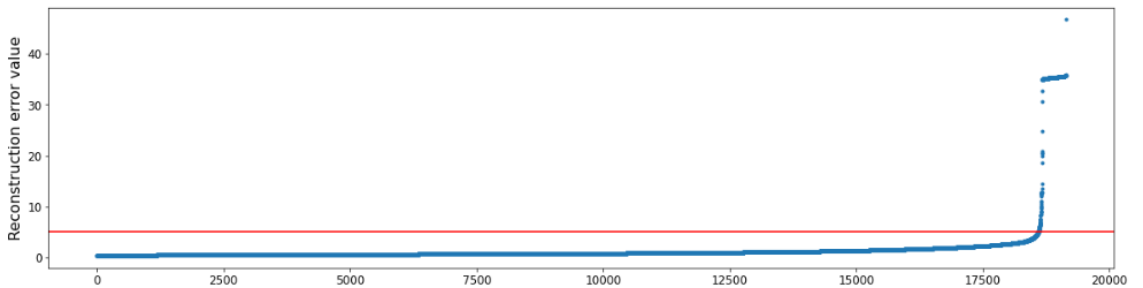


Figure 3.10: Sorted reconstruction error values

The following figures show examples of signals fed to the autoencoder for reconstruction. The figures include both the original and reconstructed signals, with the reconstruction errors prior to summation. The term "shifted reconstruction error" that appears on the following plots, means nothing else, but the original reconstruction error signal, shifted downwards by one. This is calculated by subtracting 1 from each point of the reconstruction error signal. The purpose of this shift is only to improve the visibility and make the plots clearer.

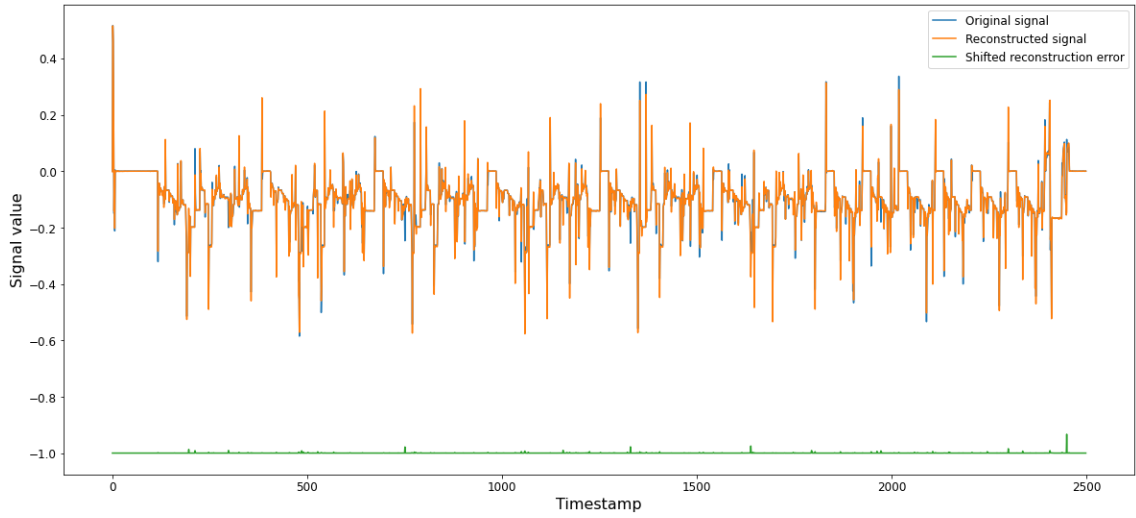


Figure 3.11: Reconstruction of a normal signal

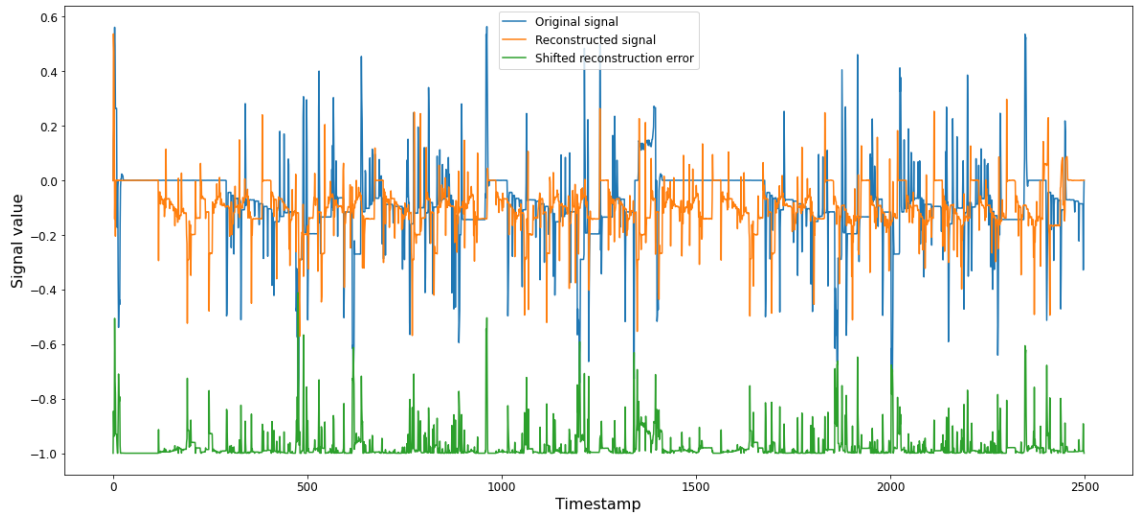


Figure 3.12: Reconstruction of a signal with a documented natural anomaly

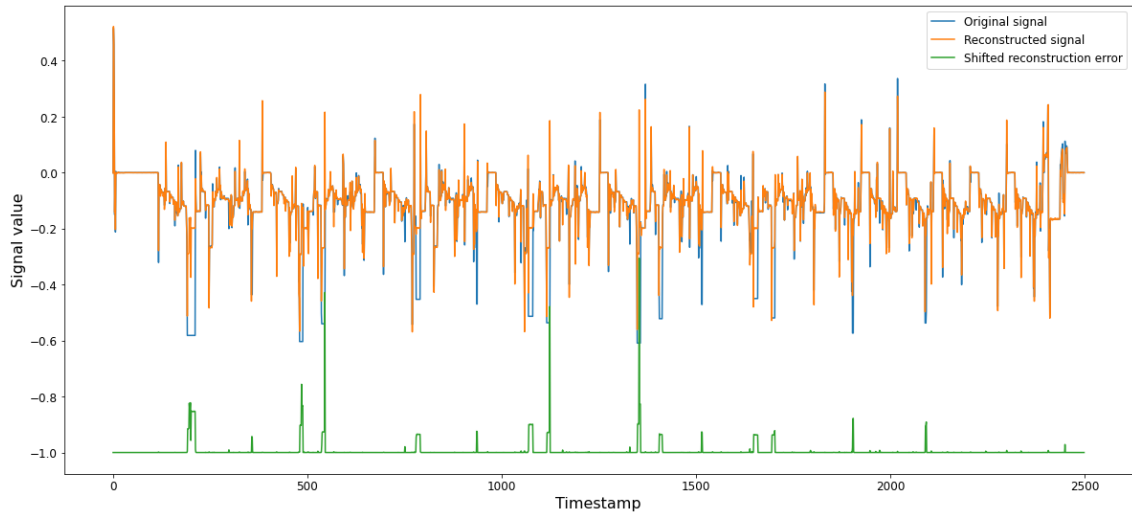


Figure 3.13: Reconstruction of a signal with a current peak error

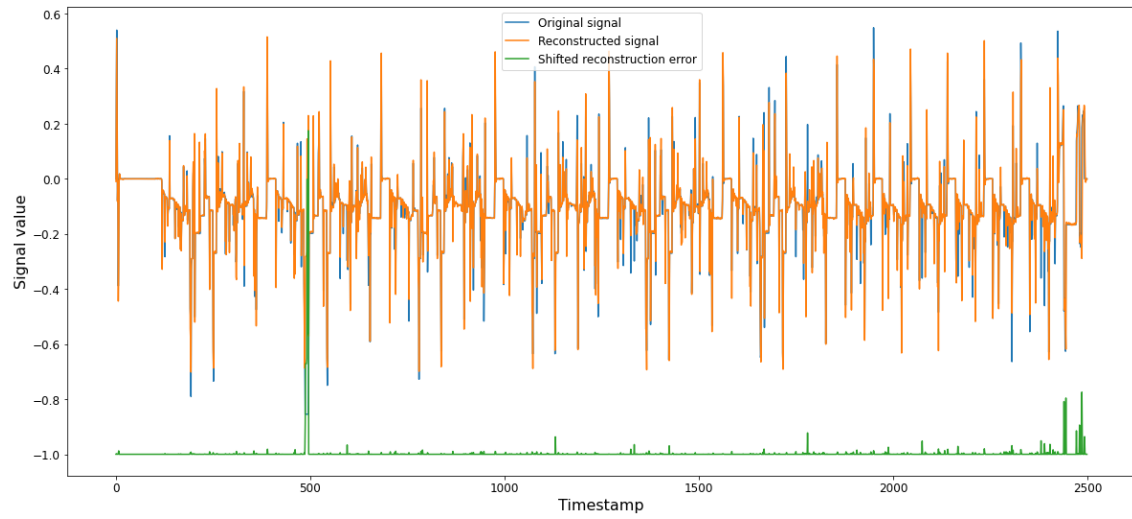


Figure 3.14: Reconstruction of a signal with a more difficult current peak error

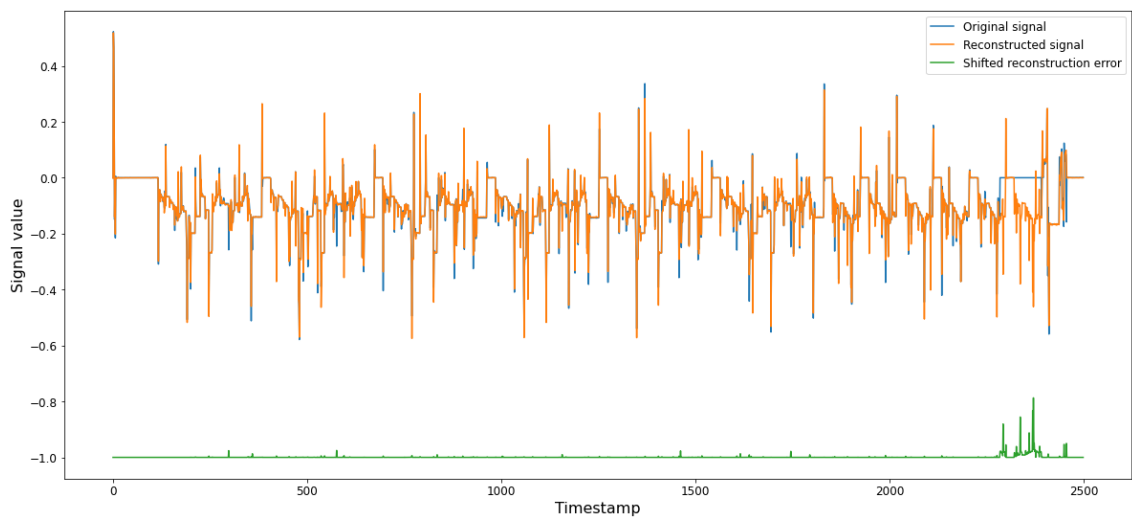


Figure 3.15: Reconstruction of a signal with a simulated motor stop error

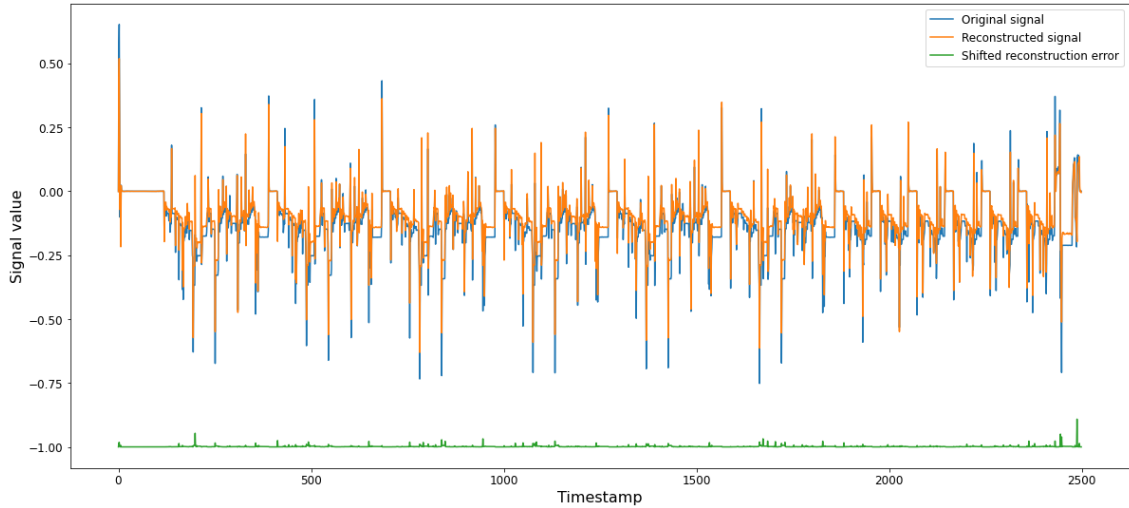


Figure 3.16: Reconstruction of a signal with an increased current error

Since the reconstruction error is represented as a continuous signal, these autoencoders are inherently interpretable. The plots, similarly to the Figures above can show where each signal deviates from the distributions learned by the models, and even the different types of errors are visible.

3.3.2 Reconstruction error analysis

As previously stated, the reconstruction errors are represented as signals, which are summed when the threshold-based anomaly detection is performed. However, instead of summing them, other properties of the signals could be utilized. A simple analysis was developed where each reconstruction error was represented by 8 values:

- Mean
- Standard deviation
- 5 largest values

The logic behind this idea is similar to the analyzed signal approach to anomaly detection described in Chapter 2.1.2, but with different statistical indicators. Since the reconstruction error can never be negative, the minimum is not a descriptive indicator. The selected indicators help model occasional spikes and peaks in the signals. An Elliptic Envelope, which is a robust unsupervised algorithm, was fitted on these analyzed signals. The algorithm estimated the parameters of the distribution of the reconstruction errors and was used in conjunction with the autoencoders for the final prediction. During training,

the autoencoder was trained to learn the distribution of the raw signals and reconstruct them using the learned parameters. Next, the reconstruction error signals were computed for the training set, and the Envelope was fitted using the calculated signal descriptors. During inference, the autoencoder was used to compute the reconstruction error signals, and the classification was made by the pre-trained Envelope using the analyzed errors.

Chapter 4

Evaluation

4.1 Evaluation metrics

In the evaluation of the trained algorithms, binary classification metrics were employed. Prior to the evaluation process, a standard was established for the outputs of the models and each model's output was transformed to this standardized format. In this case, anomalies were designated with the label 1 while non-anomalies were assigned label 0, thus the anomalies were assigned positive labels.

4.1.1 Confusion matrix

A confusion matrix is a table that is used to evaluate the performance of a binary classification model. It compares the predicted labels to the true labels of the test set, and it provides a summary of the model's performance in terms of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

A confusion matrix for a binary classification problem has four entries:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 4.1: Confusion matrix

- TP: Number of observations correctly classified as anomalies by the model.
- TN: Number of observations correctly classified as normal observations by the model.

- FP: Number of observations incorrectly classified as anomalies by the model.
- FN: Number of observations incorrectly classified as normal by the model.

The confusion matrix was not plotted for every used algorithm, only the metrics defined with the help of it were used.

4.1.2 Accuracy

Accuracy is the simplest classification metric, which can be used. It is defined as the proportion of correct predictions made by the model, out of all predictions made. The accuracy of a model can be computed by comparing the predicted labels to the true labels of the test set.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

It's worth noting that accuracy can be a misleading metric when the classes are imbalanced, which can certainly be the case in an anomaly detection task since only a small proportion of the data is anomalous, thus other metrics have to be used.

4.1.3 Precision

Precision is a commonly used metric to evaluate the performance of an anomaly detection model. It measures the proportion of true positive predictions (anomalies) out of all positive predictions made by the model.

The equation for precision is:

$$Precision = \frac{TP}{TP + FP}$$

Precision plays a critical role in evaluating the performance of anomaly detection models. It measures the model's ability to accurately identify true anomalies amongst all observations that it classifies as anomalous. High precision ensures that there will be a low number of false alarms, even if some anomalies go unnoticed.

4.1.4 Recall

Recall (also known as Sensitivity or True Positive Rate) is a commonly used metric to evaluate the performance of an anomaly detection model. It measures the proportion of actual positive observations that were correctly classified as positive by the model.

The equation for recall is:

$$Recall = \frac{TP}{TP + FN}$$

Recall is an important metric for anomaly detection tasks because it measures the ability of the model to detect actual anomalies. A high recall value means that the model can detect most of the actual anomalies. Usually, there is a tradeoff between precision and recall. A model with high precision will have a low recall and vice versa. The optimal balance between precision and recall depends on the characteristics of the data and the problem being solved. In this problem, the focus was on recall, because false alarms are not as costly as undetected anomalies.

4.1.5 F1-score

F1-score is a balance between precision and recall and is a well-suited metric for imbalanced datasets. It is defined as the harmonic mean of precision and recall:

$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = 2 \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

This was the most important evaluation metric for this anomaly detection task because it takes into account both the ability to detect anomalies and the ability to avoid false alarms.

4.2 Evaluation method

The evaluation was done in two phases. In the first phase, a test set was generated based on data the models had trained on. This can be done here because the models were trained in an unsupervised manner, hence no knowledge was gained by them based on the labels. As mentioned earlier, some parts of the training dataset are known to be anomalous. Another fact about this dataset is that the beginning usually contains no anoma-

lies, which can be explained by the nature of the product and the measurement process. To construct the first test set, the documented, known anomalies were used, along with the same amount of data taken from the beginning of the measurement, which was labeled non-anomalous for testing purposes. This led to a balanced test set, containing the same amount of anomalous signals and normal ones. The models were evaluated on this dataset, and the accuracy, precision, recall, and F1-scores were calculated. The goal of the first evaluation phase was to make sure that the models can reliably find the obvious anomalies.

The second test set was comprised of signals from the start of the measurement, which were not part of the first test set (to avoid overlap between the two test sets), and anomalies, which were artificially generated from these signals. The goal of this second test set is to evaluate the performance of the models on new, unseen data that are known to be anomalous but are not as easily recognizable as the anomalous signals in the first test set. As mentioned in Chapter 2.1.1 three types of artificial anomalies were generated. For this test set, one of the anomaly types, the current peak error had two versions, an easy and a hard version. In the easy version, every current peak corresponding to a high torque was altered (about 20-30 peaks), however, in the hard version, only one of the peaks was modified. From each of the four anomaly types, namely the current peak hard, current peak easy, motor stop error, and the increased current error, 50 samples were generated from different non-anomalous signals, leading to a size of 400 for the second test set. On this test set, the accuracy, precision, recall, and F1-score metrics were calculated as well, but the F1-score was calculated for each of the anomaly types separately too.

4.3 Evaluation results

Since various algorithms were evaluated for both the analyzed and raw signal representations, their results will be presented in separate chapters, and a final conclusion will be drawn at the end. In the subsequent chapters, only the best-performing algorithms according to the F1-score metric will be presented as it was considered the most significant metric. For other metrics and algorithms, refer to Appendix A.

4.3.1 Analyzed signals

Table 4.2: Best performing algorithms according to the F1-score metric

Algorithm	Dataset	Signal	F1_score	Artificial Anomaly F1_score	Current peak easy F1_score	Current peak hard F1_score	Motor stop F1_score	Increased current F1_score
baseline	normal	current	0.99389	0.57272	0.78935	0.05819	0.25433	0.81877
modified zscore baseline	normal	current	0.99357	0.69481	0.82042	0.12489	0.48073	0.97426
Envelope	normal	all	0.97924	0.83470	0.96794	0.11480	0.96042	0.96366
Isolation Forest	normal	current	0.94779	0.79671	0.91023	0.29537	0.84803	0.94719
Envelope	further segmented	rotation	0.98784	0.44975	0.14324	0.13456	0.96071	0.15557
Envelope	further segmented	current	0.71897	0.85229	0.90524	0.64950	0.90401	0.89313
Envelope	further segmented	torque	0.36768	0.44624	0.16427	0.13694	0.96300	0.11151
Autoencoder	further segmented	all	0.69262	0.71102	0.73501	0.57457	0.75605	0.76290
Autoencoder	further segmented	current	0.88036	0.70483	0.74124	0.59869	0.73317	0.73531
Fuzzy sys- tem	normal	all	0.98232	0.75718	0.80021	0.05202	0.82626	0.96861
Fuzzy + bma	normal	all	0.97958	0.82577	0.93815	0.06268	0.96125	0.97788

The results of the evaluation of the algorithms trained on analyzed signals are presented in Table 4.2. The baseline method showed exceptional performance on the first test set, primarily due to the distinctive nature of the anomalies present in this set. These anomalies, mostly represented by signals that were zero throughout, were easy to detect due to their significant deviation from the normal distribution of the current signal. As a result, the baseline z-score model was able to detect these anomalies without the need for a complex training process. The baseline, particularly the modified z-score baseline, also performed well in detecting artificial anomalies with increased current and easy current peak errors, probably because those had more deviation from the usual values than the other artificial anomalies. In detecting increased current, the modified z-score baseline was almost the best-performing algorithm, only surpassed by the Fuzzy system trained with a bacterial memetic algorithm, which utilized rules based on the principles of the baseline model.

When it comes to detecting artificial anomalies, the Elliptic Envelope proved to be the most effective, particularly when trained on the further segmented current signal. This is

due to the fact that artificial anomalies had the greatest impact on the current signal. The further segmented dataset provided more information to the algorithm, which didn't face any issues with dimensionality as only one signal was present in the dataset. The Envelope trained on every signal, but without further segmentation, and the Fuzzy system trained with a bacterial memetic algorithm were the closest runners-up. The Envelope trained on the further segmented rotation or torque signals was able to detect the motor stop anomaly with an impressive F1-score of 0.96. It's worth noting that the Elliptic Envelope trained on every signal was able to achieve an F1-score of 0.96 in every type of artificial anomaly except the hard current peak error. This anomaly was better detected by the Isolation Forest and the Autoencoder. This resulted in a lower overall Artificial Anomaly F1 score compared to the Envelope trained on the further segmented current signal, as the latter had a much higher F1 score in the hard current peak metric, despite slightly lower results (around 0.9) when tested on the other three artificial anomaly types. Unfortunately, this high dimension count made it perform worse on the first test set, comprised of natural anomalies. Higher dimensional solutions, such as training the model on every further segmented signal brought more information with them, the algorithms couldn't learn them well.

In evaluating the performance of the algorithms in detecting anomalies, it is observed that the Isolation Forest and the Autoencoder did not yield desirable outcomes compared to the Elliptic Envelope algorithm. The Autoencoder showed surprising results, demonstrating superior performance in detecting hard current peak errors, with an F1 score range of 0.4 to 0.6. However, it performed poorly in other metrics. The Autoencoder excelled in higher dimensional scenarios, especially when it was trained on raw signals. On the other hand, the Isolation Forest had subpar results overall but was not affected by high dimensionality, which resulted in a fast and consistent training time. In comparison, the Elliptic Envelope's training time significantly increased when handling higher dimensional data, taking several minutes to train compared to just a few seconds.

The evaluation of the Fuzzy system showed that it performed well when trained on top of the Elliptic Envelope algorithm, which itself performed well after being trained on the normal analyzed dataset that contained every signal. The Fuzzy system was not trained on datasets that only contained one signal, as the importance of explainability was deemed lower in such cases where only one signal was considered and therefore the Fuzzy system did not have to explain which signal was anomalous. Compared to the Fuzzy system trained with the bacterial evolutionary algorithm, the one trained with the

bacterial memetic algorithm performed better. However, it also required more time for the same training iteration amount, due to the added local search step. The results showed that the trained Fuzzy system outperformed the Elliptic Envelope on the first test set and the increased current and motor stop error metrics. However, it did not outperform it on the current peak error metric.

Explainability of Fuzzy systems

As mentioned in Chapter 3.2.5, Fuzzy systems are explainable algorithms because they use linguistically definable rules and the Mamdani inference shows which antecedent had the most effect on the consequent. To evaluate whether it provides correct explanations, a testing procedure was done on the artificial anomalies. The reason behind using the artificial dataset for this evaluation is that there it is known which signals are anomalous, and there are cases where only one signal, the current contains anomalies. In the naturally occurring anomalies, the reported anomalies are motor stop errors, where every signal is affected, which makes it pointless to use in this scenario.

Most easy current peak errors were found by the fuzzy system and as the explanation it returned the standard deviation descriptor of the current signal, which makes sense, because the longer peaks of the current peaks increase it, since they deviate unusually too much from the mean. The model could not find the hard current peak errors unfortunately, as indicated by the low F1-score it produced as presented in Table 4.2. In one case, however, when it could find it, the explanation was the standard deviation of the current as well. The simulated motor stop errors and the increased current errors were both mostly found by the algorithm. The explanations for the errors were not convincing in the motor stop error case. The algorithm mostly explained the anomaly by one of the errors maximums, mostly the torque's, but sometimes the rotation's and temperature's maximum was also predicted. The unfortunate fact about these results is that the motor stop error rarely affects the signals maximums and the temperature is not changed here, because that is an ambient value, which comes externally, not internally from the motor like the other measured signals. The results were much more convincing on the increased current error, where it predicted the current's minimum and standard deviation values as being anomalous, and it was right because this artificial anomaly affects those values significantly.

4.3.2 Raw signals

Table 4.3: Best performing algorithms according to the F1-score metric

Algorithm	Dataset	Signal	F1_score	Artificial Anomaly F1_score	Current peak easy F1_score	Current peak hard F1_score	Motor stop F1_score	Increased current F1_score
CNN FC L	raw 10000	current	1.00000	0.88825	0.94242	0.70802	0.91682	0.95567
CNN FC L + Envelope	raw 14000	current	1.00000	0.86781	0.95154	0.88374	0.61874	0.93818
CNN FC L + Envelope	raw 2500	current	1.00000	0.90973	0.92756	0.89615	0.86162	0.94844
CNN FC L + Envelope	raw 5000	all	1.00000	0.66060	0.73876	0.33469	0.94961	0.39133
CNN FC L + Envelope	raw 5000	current	1.00000	0.90800	0.93796	0.91375	0.83417	0.93801
CNN FC S + Envelope	raw 10000	current	1.00000	0.87197	0.94418	0.88284	0.66823	0.95270
CNN FC S + Envelope	raw 2500	current	1.00000	0.89790	0.93792	0.88709	0.80096	0.95448
FC NET + Envelope	raw 14000	current	0.99965	0.86847	0.94983	0.87873	0.64537	0.93954
FC NET + Envelope	raw 2500	all	1.00000	0.80416	0.93921	0.73011	0.93912	0.49886
FC NET + Envelope	raw 2500	current	1.00000	0.91572	0.93782	0.89682	0.87490	0.94976
FC NET + Envelope	raw 5000	all	1.00000	0.76281	0.90382	0.47828	0.95710	0.56034
FC NET + Envelope	raw 5000	current	1.00000	0.90822	0.93776	0.89871	0.84275	0.94664

The results of the evaluation of the algorithms trained on the raw time series signals are presented in Table 4.3. In this table, only the best-performing algorithms are shown, which were the hybrid approach with both convolutional and fully connected layers, and the fully connected network. Unfortunately the fully convolutional, and the U-Net based approach utilizing skip connections couldn't perform well enough. The reason for it could be that those approaches weren't researched initially for reconstruction based anomaly detection, instead were used for learning a transformation on the input, namely semantic segmentation. To model the distribution of the signals, the neural networks need fully-connected layers, which learn these values from the signals processed by the convolutional layers, the convolutional layers by themselves are not enough. On the first test set, the naturally occurring anomalies, nearly all hybrid, and fully connected models reached a perfect F1 score of 1. The reason behind this was simple, the natural anomalies deviated too far from the usual signals and the reconstruction error was always high. The additional Elliptic Envelope could help most of the trained neural networks to find the artificial anomalies,

especially the hard current peak errors. Those errors were harder to find by only utilizing the sum of the reconstruction errors, since those sums did not deviate enough to be above the threshold to be considered anomalies. However, the maximum of the reconstruction error signal was high due to the unusual peak found in the data, making it valuable information to be extracted from the error signals.

In finding the easy current peak and increased current artificial anomalies the algorithms generally performed well, producing F1 scores above 0.9 and in the best cases even above 0.95. The motor stop errors were harder to find by the algorithms when only trained on the current signal. The algorithms could only rarely reach the F1 score of 0.9. However, when more signals were given to the algorithm during training, this error was found with an F1 score above 0.95. The reason for this is that the motor stop error is visible in every signal, while the rest only affect the current. Unfortunately training the model on too many signals required more memory and computational power. Because of this, most trainings were only done on the current signal. A surprising fact about the signals is that the sampling rate did not matter much in the results. Originally with a sampling rate of 100 Hz, most signals contained around 14000 datapoints. The datapoint count to which the signals were resampled, can be seen in the Dataset column of Table 4.3.

Overall in finding the artificial anomalies, the algorithms could reach F1 scores above 0.9. The fully connected networks had overall a little better result than the convolutional ones, however those algorithms had the most parameters, since connections had to be made to every datapoint in the input signal. The fully connected networks contained around 90 million trainable parameters for every signal they were trained on. The convolutional architecture did not depend on the input signal count, because of the filters the convolutional layers have. From this architecture, a Large and a Small version were both designed, the Large containing around 60 million parameters, while the small containing around 9 million. The smaller models overall had worse results, but not significantly worse, so they could be useful in scenarios where the available memory and computational power are limited.

4.3.3 Comparison of raw and analyzed signal results

Overall in finding the natural anomalies, which make the first test set, the algorithms trained on the raw signals easily outperformed every algorithm trained on the analyzed signals, because of their perfect scores. However, this performance increase is not significant, since the baseline model could almost reach this perfect score as well, making it suitable for detecting these natural anomalies with much fewer parameters. When the F1-score of every artificial anomaly is taken into account, the anomalies on the raw signals yielded better results again. The reason for this performance increase lies mostly in the ability of these models to find the hard current peak errors with better precision. That intricate error type only distorts the signal in a narrow area, which hardly makes any difference in the statistical indicators the signals are described with during the signal analysis process. For this error type, the examination of the whole signal is needed. On the rest of the artificial anomaly types, the two methods had almost similar results. In some cases, algorithms used on analyzed signals could even outperform the autoencoders trained on the whole signals. When taking the explainability factor into account as well, the Fuzzy system could provide some correct explanations, however, the power of the visual explanations provided by the continuous reconstruction error signals is unmatched.

Analyzing the signals is one additional step in the data preprocessing pipeline, however, it can allow the use of algorithms, which need much fewer parameters than large neural networks fitted on the raw signals. These algorithms can detect some types of anomalies without the need for training on expensive GPUs, and thus can be a better choice on smaller devices or simpler datasets.

In an industrial scenario however, when the required computational power is provided and the goal is to maximize the ability of the model to find every possible anomaly, the use of large neural network structures is more beneficial.

Chapter 5

Future work

The project was conducted on a pre-existing measurement with a comprehensive dataset. In the future, new measurements will be performed without the benefit of a complete dataset. To address this issue, an incremental learning approach (also known as online learning) can be utilized. Given that most motors typically perform optimally at the beginning of their lifecycle, the models can be trained on the initial dataset, which contains no anomalies, to learn the fundamental distribution of the signals. As additional data becomes available, anomaly detection can be performed on the new data using the previously trained model. When anomaly detection is done on new signals, explanations, especially the visual ones can indicate which signals show the malfunction of the product. A numerical value, such as the summed reconstruction error, or the Mahalanobis distance could even indicate the magnitude of the predicted anomaly. Once a sufficient amount of newly generated data has been obtained, the models can be further trained or retrained to improve performance.

The combination of multiple models has the potential to bring out their advantages and strengths. In this project, after the preprocessing steps such as segmentation are applied to the data, a baseline model can be established with ease, highlighting the most prominent anomalies. In the case of low-dimensional data, an Elliptic Envelope can effectively and efficiently detect data points that deviate significantly from the core distribution. For higher-dimensional datasets, an Isolation Forest can be utilized to isolate these anomalies. Based on the results from these simple models, a more complex algorithm, such as an autoencoder, can then be trained for a more in-depth analysis of the raw signals. To ensure that the autoencoder learns the distribution of the normal data and not the anomalies, the discovered anomalies can be removed from the training set prior to training the net-

work. Different models might excel in finding different types of anomalies. These models could even be combined to create an ensemble of different models, which could vote on whether the input signal is anomalous or not. For this approach, new model architectures could be researched. Instead of the simple analysis of the time series signals or the reconstruction error signals, some more advanced signal processing ideas could also lead to a performance increase in already existing models.

The project was focused on a selected set of crucial signals, which required advanced anomaly detection techniques, such as the current signal. However, for other signals, a simpler method would suffice, highlighting the need for a flexible and adaptable approach to anomaly detection that can be tailored to the complexity of the signals in question.

Chapter 6

Summary

This thesis evaluates various algorithms for the purpose of identifying the most effective one for a research project aimed at detecting anomalies in the data. The thesis provides an overview of the functioning of different algorithms and offers insights into their applications to similar datasets. The conclusion is that there is no universal algorithm that performs optimally for all datasets and that the best algorithm depends on specific characteristics such as the number of outliers, dimensionality, or available computational power. It is therefore crucial to consider these factors when selecting an algorithm and to perform thorough evaluations of multiple algorithms to determine the most suitable one. The research was conducted in an industrial setting, where computationally expensive algorithms can be employed, but in smaller projects, the most accurate algorithm may not necessarily be the best choice. For easily identifiable anomalies, simple statistical methods can work, however for more complex anomalies, machine learning-based solutions especially modern deep neural networks yield the best results. These solutions can learn and extract relevant information from the multivariate time series signals and can indicate whether anomalies are present in them and even explain where these anomalies are.

Appendix A

Evaluation results

Table A.1: F1-score metric of every algorithm

Algorithm	Dataset	Signal	F1_score	Artificial Anomaly F1_score	Current peak easy F1_score	Current peak hard F1_score	Motor stop F1_score	Increased current F1_score
baseline	normal	current	0.99389	0.57272	0.78935	0.05819	0.25433	0.81877
modified	normal	current	0.99357	0.69481	0.82042	0.12489	0.48073	0.97426
zscore								
baseline								
Envelope	normal	all	0.97924	0.83470	0.96794	0.11480	0.96042	0.96366
Isolation	normal	all	0.96091	0.72851	0.59239	0.22348	0.93163	0.92840
Forest								
Envelope	normal	rotation	0.91982	0.12494	0.15359	0.09289	0.13604	0.11037
Isolation	normal	rotation	0.94247	0.41723	0.28096	0.27661	0.72466	0.26207
Forest								
Envelope	normal	current	0.12005	0.14465	0.16799	0.11646	0.15356	0.13261
Isolation	normal	current	0.94779	0.79671	0.91023	0.29537	0.84803	0.94719
Forest								
Envelope	normal	torque	0.98470	0.43453	0.12044	0.08780	0.95484	0.12227
Isolation	normal	torque	0.95754	0.53471	0.39866	0.32683	0.88381	0.35715
Forest								
Envelope	further	all	0.57369	0.70162	0.62266	0.27480	0.95748	0.68593
	segmented							
Isolation	further	all	0.97886	0.44256	0.29671	0.13192	0.22995	0.85779
Forest	segmented							
Envelope	further	rotation	0.98784	0.44975	0.14324	0.13456	0.96071	0.15557
	segmented							
Isolation	further	rotation	0.95799	0.22589	0.20555	0.20416	0.28734	0.19214
Forest	segmented							
Envelope	further	current	0.71897	0.85229	0.90524	0.64950	0.90401	0.89313
	segmented							
Isolation	further	current	0.96040	0.59488	0.71184	0.12800	0.20479	0.96324
Forest	segmented							
Envelope	further	torque	0.36768	0.44624	0.16427	0.13694	0.96300	0.11151
	segmented							
Isolation	further	torque	0.96531	0.19042	0.15131	0.15256	0.29835	0.13549
Forest	segmented							
Autoencoder	normal	all	0.91089	0.68969	0.76112	0.50108	0.66805	0.78669
Autoencoder	normal	rotation	0.98259	0.43955	0.44159	0.40066	0.47983	0.42776
Autoencoder	normal	current	0.96036	0.67013	0.73249	0.41882	0.65452	0.79908
Autoencoder	normal	torque	0.98386	0.48701	0.49377	0.47027	0.48845	0.48885
Autoencoder	further	all	0.69262	0.71102	0.73501	0.57457	0.75605	0.76290
	segmented							
Autoencoder	further	rotation	0.67877	0.60241	0.54521	0.52901	0.75527	0.53894
	segmented							
Autoencoder	further	current	0.88036	0.70483	0.74124	0.59869	0.73317	0.73531
	segmented							
Fuzzy sys- tem	normal	all	0.98232	0.75718	0.80021	0.05202	0.82626	0.96861
Fuzzy + bma	normal	all	0.97958	0.82577	0.93815	0.06268	0.96125	0.97788

Table A.2: Other metrics of every algorithm

Algorithm	Dataset	Signal	Accuracy	Precision	Recall	Artificial Anomaly Accuracy	Artificial Anomaly Precision	Artificial Anomaly Recall
baseline	normal	current	0.99380	0.98931	0.99862	0.71583	0.97502	0.44917
modified	normal	current	0.99345	0.98804	0.99931	0.76667	0.94205	0.57917
zscore								
baseline								
Envelope	normal	all	0.98036	0.98627	0.97795	0.84792	0.91723	0.76667
Isolation	normal	all	0.96003	0.94588	0.97795	0.75708	0.82788	0.65833
Forest								
Envelope	normal	rotation	0.94728	1.00000	0.89456	0.50042	0.51323	0.07250
Isolation	normal	rotation	0.94211	0.91390	0.97795	0.56458	0.63602	0.32667
Forest								
Envelope	normal	current	0.49483	0.43633	0.07099	0.50000	0.50137	0.08500
Isolation	normal	current	0.94521	0.92099	0.97795	0.81000	0.86026	0.74583
Forest								
Envelope	normal	torque	0.98656	0.99296	0.97795	0.61292	0.81454	0.29833
Isolation	normal	torque	0.95555	0.94025	0.97795	0.59125	0.63379	0.48167
Forest								
Envelope	further	all	0.71245	0.79810	0.51870	0.76458	0.85862	0.62167
	segmented							
Isolation	further	all	0.97878	0.97623	0.98213	0.61667	0.80296	0.31250
Forest	segmented							
Envelope	further	rotation	0.98827	0.99889	0.97767	0.61458	0.79164	0.31583
	segmented							
Isolation	further	rotation	0.95729	0.93951	0.97767	0.51083	0.53862	0.14667
Forest	segmented							
Envelope	further	current	0.80737	0.92014	0.65438	0.84042	0.82861	0.89333
	segmented							
Isolation	further	current	0.95812	0.94139	0.98269	0.68958	0.85917	0.45917
Forest	segmented							
Envelope	further	torque	0.57677	0.58478	0.33948	0.61542	0.79684	0.31000
	segmented							
Isolation	further	torque	0.96454	0.94516	0.98660	0.51583	0.57866	0.11500
Forest	segmented							
Autoencoder	normal	all	0.92074	0.98705	0.85320	0.63958	0.61748	0.79167
Autoencoder	normal	rotation	0.98484	0.99239	0.97726	0.50417	0.50536	0.41333
Autoencoder	normal	current	0.96210	0.98112	0.94280	0.65208	0.66585	0.71833
Autoencoder	normal	torque	0.98587	0.99518	0.97450	0.50125	0.50130	0.49500
Autoencoder	further	all	0.77275	0.94974	0.56784	0.63000	0.58350	0.91000
	segmented							
Autoencoder	further	rotation	0.79034	0.93207	0.59296	0.54333	0.54059	0.70917
	segmented							
Autoencoder	further	current	0.91318	0.93689	0.87940	0.62458	0.58281	0.89833
	segmented							
Fuzzy sys- tem	normal	all	0.98346	0.99245	0.97795	0.80625	0.95224	0.64833
Fuzzy + bma	normal	all	0.98070	0.98689	0.97795	0.84625	0.95051	0.73333

Table A.3: F1-score metric of every algorithm

Algorithm	Dataset	Signal	F1_score	Artificial Anomaly F1_score	Current peak easy F1_score	Current peak hard F1_score	Motor stop F1_score	Increased current F1_score
CNN FC L	raw 10000	all	1.00000	0.74684	0.79730	0.56883	0.79685	0.78668
CNN FC L	raw 10000	current	1.00000	0.88825	0.94242	0.70802	0.91682	0.95567
CNN FC L	raw 14000	current	1.00000	0.85537	0.90911	0.69666	0.87871	0.91474
CNN FC L	raw 2500	all	1.00000	0.74422	0.79987	0.52682	0.83587	0.76835
CNN FC L	raw 2500	current	1.00000	0.86564	0.90686	0.72002	0.89151	0.92624
CNN FC L	raw 5000	all	1.00000	0.73654	0.77369	0.58065	0.80688	0.76104
CNN FC L	raw 5000	current	1.00000	0.85490	0.90290	0.70494	0.87277	0.91933
CNN FC L	raw 14000	current	1.00000	0.86781	0.95154	0.88374	0.61874	0.93818
+ Envelope CNN FC L	raw 2500	all	1.00000	0.76029	0.91565	0.58236	0.93180	0.46163
+ Envelope CNN FC L	raw 2500	current	1.00000	0.90973	0.92756	0.89615	0.86162	0.94844
+ Envelope CNN FC L	raw 5000	all	1.00000	0.66060	0.73876	0.33469	0.94961	0.39133
+ Envelope CNN FC L	raw 5000	current	1.00000	0.90800	0.93796	0.91375	0.83417	0.93801
+ Envelope CNN FC S	raw 10000	current	1.00000	0.86221	0.91676	0.68607	0.89708	0.92112
CNN FC S	raw 2500	current	1.00000	0.87688	0.92060	0.72430	0.90099	0.93524
CNN FC S	raw 10000	current	1.00000	0.87197	0.94418	0.88284	0.66823	0.95270
+ Envelope CNN FC S	raw 2500	current	1.00000	0.89790	0.93792	0.88709	0.80096	0.95448
+ Envelope FC NET	raw 10000	current	1.00000	0.87949	0.91242	0.73455	0.91710	0.93504
FC NET	raw 14000	current	1.00000	0.88389	0.91810	0.74421	0.91993	0.93493
FC NET	raw 2500	all	1.00000	0.80356	0.88922	0.49196	0.91096	0.84010
FC NET	raw 2500	current	1.00000	0.89649	0.93568	0.73757	0.93303	0.95252
FC NET	raw 5000	all	1.00000	0.79922	0.89599	0.45831	0.91266	0.82886
FC NET	raw 5000	current	1.00000	0.89170	0.92517	0.73258	0.93877	0.94383
FC NET + Envelope	raw 10000	current	0.99965	0.88586	0.93532	0.87408	0.78012	0.93744
FC NET + Envelope	raw 14000	current	0.99965	0.86847	0.94983	0.87873	0.64537	0.93954
FC NET + Envelope	raw 2500	all	1.00000	0.80416	0.93921	0.73011	0.93912	0.49886
FC NET + Envelope	raw 2500	current	1.00000	0.91572	0.93782	0.89682	0.87490	0.94976
FC NET + Envelope	raw 5000	all	1.00000	0.76281	0.90382	0.47828	0.95710	0.56034
FC NET + Envelope	raw 5000	current	1.00000	0.90822	0.93776	0.89871	0.84275	0.94664
Fully Conv	raw 10000	current	0.98897	0.72922	0.77964	0.63503	0.70800	0.77177
Fully Conv	raw 2500	current	0.94709	0.73934	0.75858	0.69027	0.73499	0.76567
Fully Conv	raw 10000	current	0.88078	0.74474	0.92267	0.63021	0.55425	0.79398
+ Envelope Fully Conv	raw 2500	current	0.99551	0.77991	0.94353	0.66104	0.55314	0.86240
+ Envelope U Net	raw 10000	current	0.88744	0.72390	0.86314	0.58333	0.49827	0.84616
U Net	raw 2500	current	0.99659	0.74843	0.79233	0.68254	0.69365	0.80295
U Net + Envelope	raw 10000	current	0.99402	0.56661	0.81200	0.31915	0.27210	0.68128
U Net + Envelope	raw 2500	current	0.99237	0.66364	0.94177	0.44074	0.22525	0.78225

Table A.4: Other metrics of every algorithm

Algorithm	Dataset	Signal	Accuracy	Precision	Recall	Artificial Anomaly Accuracy	Artificial Anomaly Precision	Artificial Anomaly Recall
CNN FC L	raw 10000	all	1.00000	1.00000	1.00000	0.68625	0.64855	0.90583
CNN FC L	raw 10000	current	1.00000	1.00000	1.00000	0.88833	0.88502	0.89250
CNN FC L	raw 14000	current	1.00000	1.00000	1.00000	0.84667	0.81575	0.90167
CNN FC L	raw 2500	all	1.00000	1.00000	1.00000	0.70292	0.66252	0.86167
CNN FC L	raw 2500	current	1.00000	1.00000	1.00000	0.85833	0.82447	0.91167
CNN FC L	raw 5000	all	1.00000	1.00000	1.00000	0.68042	0.62852	0.89417
CNN FC L	raw 5000	current	1.00000	1.00000	1.00000	0.84750	0.81406	0.90083
CNN FC L	raw 14000	current	1.00000	1.00000	1.00000	0.86917	0.86858	0.87083
+ Envelope								
CNN FC L	raw 2500	all	1.00000	1.00000	1.00000	0.78125	0.83481	0.70083
+ Envelope								
CNN FC L	raw 2500	current	1.00000	1.00000	1.00000	0.90667	0.87790	0.94500
+ Envelope								
CNN FC L	raw 5000	all	1.00000	1.00000	1.00000	0.71542	0.79310	0.57500
+ Envelope								
CNN FC L	raw 5000	current	1.00000	1.00000	1.00000	0.90542	0.88282	0.93500
+ Envelope								
CNN FC S	raw 10000	current	1.00000	1.00000	1.00000	0.85458	0.83519	0.89667
CNN FC S	raw 2500	current	1.00000	1.00000	1.00000	0.87292	0.85328	0.90500
CNN FC S	raw 10000	current	1.00000	1.00000	1.00000	0.87292	0.87577	0.86917
+ Envelope								
CNN FC S	raw 2500	current	1.00000	1.00000	1.00000	0.89542	0.87489	0.92333
+ Envelope								
FC NET	raw 10000	current	1.00000	1.00000	1.00000	0.87417	0.84628	0.91667
FC NET	raw 14000	current	1.00000	1.00000	1.00000	0.87917	0.85193	0.91917
FC NET	raw 2500	all	1.00000	1.00000	1.00000	0.79917	0.78669	0.82417
FC NET	raw 2500	current	1.00000	1.00000	1.00000	0.89458	0.88219	0.91333
FC NET	raw 5000	all	1.00000	1.00000	1.00000	0.79792	0.79417	0.80917
FC NET	raw 5000	current	1.00000	1.00000	1.00000	0.88958	0.87385	0.91167
FC NET +	raw 10000	current	0.99965	0.99930	1.00000	0.88458	0.87345	0.89917
Envelope								
FC NET +	raw 14000	current	0.99965	0.99930	1.00000	0.86958	0.87034	0.87000
Envelope								
FC NET +	raw 2500	all	1.00000	1.00000	1.00000	0.81458	0.85096	0.76333
Envelope								
FC NET +	raw 2500	current	1.00000	1.00000	1.00000	0.91292	0.88481	0.94917
Envelope								
FC NET +	raw 5000	all	1.00000	1.00000	1.00000	0.78417	0.83523	0.70750
Envelope								
FC NET +	raw 5000	current	1.00000	1.00000	1.00000	0.90542	0.88042	0.93833
Envelope								
Fully Conv	raw 10000	current	0.98660	0.98138	0.99929	0.66625	0.61866	0.90167
Fully Conv	raw 2500	current	0.93759	0.90864	0.99929	0.65667	0.61878	0.94917
Fully Conv	raw 10000	current	0.82405	0.81867	1.00000	0.77458	0.83864	0.67833
+ Envelope								
Fully Conv	raw 2500	current	0.99471	0.99266	0.99929	0.80417	0.84970	0.73167
+ Envelope								
U Net	raw 10000	current	0.94147	0.90973	0.88575	0.69917	0.70558	0.77667
U Net	raw 2500	current	0.99542	0.99577	0.99859	0.66500	0.66539	0.91833
U Net +	raw 10000	current	0.99401	0.98886	0.99929	0.65708	0.73844	0.47000
Envelope								
U Net +	raw 2500	current	0.99048	0.98881	0.99788	0.72083	0.81828	0.56333
Envelope								

Bibliography

- [1] Peter J. Rousseeuw and Katrien Van Driessen. “A Fast Algorithm for the Minimum Covariance Determinant Estimator”. In: *Technometrics* 41.3 (1999), pp. 212–223. DOI: 10 . 1080 / 00401706 . 1999 . 10485670. eprint: <https://www.tandfonline.com/doi/pdf/10.1080/00401706.1999.10485670>. URL: <https://www.tandfonline.com/doi/abs/10.1080/00401706.1999.10485670>.
- [2] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation Forest”. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.
- [3] Leo Breiman. “Random Forests”. English. In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 0885-6125. DOI: 10 . 1023 / A : 1010933404324. URL: <http://dx.doi.org/10.1023/A%3A1010933404324>.
- [4] Kunihiro Fukushima. “Cognitron: A self-organizing multilayered neural network”. In: *Biological Cybernetics* 20 (1975), pp. 121–136.
- [5] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400 –407. DOI: 10 . 1214 / aoms / 1177729586. URL: <https://doi.org/10.1214/aoms/1177729586>.
- [6] J. Kiefer and J. Wolfowitz. “Stochastic Estimation of the Maximum of a Regression Function”. In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462 –466. DOI: 10 . 1214 / aoms / 1177729392. URL: <https://doi.org/10.1214/aoms/1177729392>.
- [7] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10 . 48550 / ARXIV . 1412 . 6980. URL: <https://arxiv.org/abs/1412.6980>.

- [8] Andries P. Engelbrecht. *Computational intelligence: An introduction*. CRC Press, 2008.
- [9] E.H. Mamdani and S. Assilian. “An experiment in linguistic synthesis with a fuzzy logic controller”. In: *International Journal of Man-Machine Studies* 7.1 (1975), pp. 1–13. ISSN: 0020-7373. DOI: [https://doi.org/10.1016/S0020-7373\(75\)80002-2](https://doi.org/10.1016/S0020-7373(75)80002-2). URL: <https://www.sciencedirect.com/science/article/pii/S0020737375800022>.
- [10] John H. Holland. *Adaptation in Natural and Artificial Systems*. second edition, 1992. Ann Arbor, MI: University of Michigan Press, 1975.
- [11] Pablo Moscato. “On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms”. In: 1989.
- [12] János Botzheim, László Gál, and Laszlo Koczy. “Fuzzy Rule Base Model Identification by Bacterial Memetic Algorithms”. In: vol. 222. May 2009, pp. 21–43. ISBN: 978-3-642-02186-2. DOI: 10.1007/978-3-642-02187-9_3.
- [13] J. Botzheim et al. “Estimating fuzzy membership functions parameters by the Levenberg-Marquardt algorithm”. In: *2004 IEEE International Conference on Fuzzy Systems (IEEE Cat. No.04CH37542)*. Vol. 3. 2004, 1667–1672 vol.3. DOI: 10.1109/FUZZY.2004.1375431.
- [14] Kenneth Levenberg. “A METHOD FOR THE SOLUTION OF CERTAIN NON – LINEAR PROBLEMS IN LEAST SQUARES”. In: *Quarterly of Applied Mathematics* 2 (1944), pp. 164–168.
- [15] Donald W. Marquardt. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters”. In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441. ISSN: 03684245. URL: <http://www.jstor.org/stable/2098941> (visited on 01/11/2023).
- [16] S. Kaisler. “Expert systems: An overview”. In: *IEEE Journal of Oceanic Engineering* 11.4 (1986), pp. 442–448. DOI: 10.1109/JOE.1986.1145205.
- [17] Charles C. Phiri et al. “Fuzzy rule-based model for outlier detection in a Topical Negative Pressure Wound Therapy Device”. In: *ISA Transactions* 117 (2021), pp. 16–27. ISSN: 0019-0578. DOI: <https://doi.org/10.1016/j.isatra.2021.01.046>. URL: <https://www.sciencedirect.com/science/article/pii/S0019057821000562>.

- [18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: <http://arxiv.org/abs/1505.04597>.
- [19] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2019. DOI: 10 . 48550 / ARXIV . 1909 . 11942. URL: <https://arxiv.org/abs/1909.11942>.
- [20] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10 . 48550 / ARXIV . 1706.03762. URL: <https://arxiv.org/abs/1706.03762>.
- [21] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10 . 48550 / ARXIV . 1810 . 04805. URL: <https://arxiv.org/abs/1810.04805>.

List of Figures

2.1	Current signal over multiple cycles	4
2.2	Motor rotation signal over multiple cycles	5
2.3	Current signal over one cycle in the correct operating mode	5
2.4	Motor rotation signal over one cycle in the correct operating mode	6
2.5	Current signal over one atomic cycle	6
2.6	Motor rotation signal over one atomic cycle	7
2.7	Several identified natural anomalies	8
2.8	Current peak error	8
2.9	Motor stop error	9
2.10	Increased current error	9
2.11	Current averages of a motor without anomalous behavior	11
2.12	Current averages of a motor with anomalous behavior	11
2.13	Current averages histogram of a motor without anomalous behaviour . . .	12
2.14	Current averages histogram of a motor with anomalous behaviour	12
3.1	Reconstruction error over time	19
3.2	Sorted reconstruction error values	19
3.3	Trapezoidal membership function	20
3.4	The Mamdani inference process	21
3.5	Changed rule	24
3.6	Unchanged rule	24
3.7	The rule base before training	24
3.8	The rule base after training	25
3.9	Reconstruction error over time	29
3.10	Sorted reconstruction error values	29
3.11	Reconstruction of a normal signal	30
3.12	Reconstruction of a signal with a documented natural anomaly	30
3.13	Reconstruction of a signal with a current peak error	31

3.14 Reconstruction of a signal with a more difficult current peak error	31
3.15 Reconstruction of a signal with a simulated motor stop error	31
3.16 Reconstruction of a signal with an increased current error	32

List of Tables

4.1	Confusion matrix	34
4.2	Best performing algorithms according to the F1-score metric	38
4.3	Best performing algorithms according to the F1-score metric	41
A.1	F1-score metric of every algorithm	48
A.2	Other metrics of every algorithm	49
A.3	F1-score metric of every algorithm	50
A.4	Other metrics of every algorithm	51