Deep autoencoder for condition monitoring of wind turbines

- Detecting and diagnosing anomalies

Master's thesis in Complex Adaptive Systems

JOHANNA RENMAN

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Deep autoencoder for condition monitoring of wind turbines

Detecting and diagnosing anomalies

JOHANNA RENMAN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2019

Deep autoencoder for condition monitoring of wind turbines
– Detecting and diagnosing anomalies
JOHANNA RENMAN

Supervisor: Pramod Bangalore, Greenbyte AB
Examiner: Kristian Gustafsson, Department of Physics, University of Gothenburg

Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Wind turbine

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Abstract

Over the last decade, energy production from wind turbines has grown by 400%, prompted by public investments and climate change awareness as well as advances in technology. With subsidies offered to wind farms dwindling, the owners and operators of wind farms are forced to cut operational cost to stay profitable. This has lead to a renewed focus on predictive and preventive maintenance, targeting not only the traditionally well monitored large components in the wind turbine, but also including smaller, more easily replaced components. Advances in the use of machine learning to model complex system, combined with the growing access of data have allowed advanced methods for condition monitoring and anomaly detection to be developed. This has been applied to the field of condition monitoring for wind turbines in various research projects, utilizing data from the Supervisory Control and Data Acquisition (SCADA) system available in modern wind turbines. Most of these systems have modeled just one component at a time and are therefore not able to provide a complete condition monitoring system. Using autoencoders for condition monitoring of wind turbines enables the whole wind turbine to be modeled as one system, by learning the internal connections between the SCADA signals. This has previously been studied, but most studies focus on the anomaly detection step, and leave out the important part of diagnosing which signals are most affected by the anomaly. By finding these signals, the source of the fault can be found, which is important to allow for recommendation on where to do maintenance. This thesis investigates the application of deep autoencoders to detect and diagnose developing faults. The autoencoder has been used to produce a residual, taken as the error between the input to the autoencoder and its reconstructed signal. For the fault detection, the Mahalanobis distance has been used on the residual. For fault diagnosis, the residual for each signal has been standardized and analyzed to examine which signals are mostly affected by the fault. This was tested on eight known faults found in five different components: gearbox, cooling system, hydraulic system, yaw encoder issue and generator slipring. The proposed condition monitoring system was successful in detecting and diagnosing all faults but one.
This thesis also presents an approach to understanding what the autoencoder has learned, with the use of simulated faults. The study provides a good method for discovering what connections between the SCADA signals the autoencoder has learned as well as information about how the residual is affected when one signal is experiencing a fault.

Keywords: *Wind turbine, condition monitoring, anomaly detection, preventive maintenance, SCADA, autoencoder, neural networks, fault diagnosis, anomaly diagnosis, fault detection*

Acknowledgements

I would like to thank the whole of Geeenbyte for making this thesis possible. In particular I want to thank my supervisor, Pramod Bangalore, for his guidance during all stages of the process: Thank you for always taking time to answer my questions and your support when I needed it. I would also like to thank Niklas Renström, who did not only provide me with his research in the area, but who was also open for discussing ideas and results. Thank you Edmund for offering help and for your expertise in English grammar and thank you Thomas for your support and openness. I would also like to thank Kristian Gustafsson for being my examiner and for his interest in the research.
Finally, a big thank you to all of my friends and family, not only for this thesis but for being there during my whole education. Without you this would not have been possible and I am forever grateful.

Johanna Renman, Gothenburg, December 2019

Contents

1

Introduction

1.1 Background and problem overview

Over the last decade, energy production from wind turbines has grown by 400% [1]. In 2018, wind power stood for a large part of the world's current renewable electricity with 24 % of the total capacity [2]. This growth is prompted by public investments and climate change awareness as well as advances in technology. Wind farms are increasingly being built offshore for several reasons, including less disturbance in urban areas, stronger and more stable wind and possibility for larger units, which in turn may produce more energy [3]. The cost of maintenance for offshore wind farms is however significant, which increases the need for good condition monitoring systems to allow for preventive maintenance. A good condition monitoring system is of course not only good for offshore wind farms but also onshore wind farms can greatly benefit from predicting and planning maintenance. In fact, over an operating life of 20 years, maintenance costs for wind farm may reach 15% of the total income for onshore wind farms and 30% of total income for offshore wind farms. This cost can be greatly reduced with a good condition monitoring system that allows for predictive and preventive maintenance [4].

Condition monitoring involves observing the components of a wind turbine in order to identify changes in the system that could indicate a developing fault. Traditionally condition monitoring rely on e.g. visual inspection, vibration analysis, strain measurement, thermography and acoustic emissions. Some of these techniques are intrusive and impose a wear on the component being monitored [5]. The advances in data-driven modelling using machine learning and the recent developments in sensors and signal processing systems have allowed for new types of data-driven condition monitoring systems, often utilizing SCADA (Supervisory Control And Data Acquisition) data. Data-driven models are an effective way of creating a model without having to consider the mathematical model of the physical system, instead the internal connections between measured input and output signals are learned to model the system.

This thesis aims to design such a condition monitoring system for wind turbines with the use of SCADA data. The system should (i) be able to detect developing faults and (ii) be able to diagnose the fault, i.e. finding the degrading component(s). In particular, the use of deep autoencoders, which is a type of Artificial Neural Network, for condition monitoring purpose is examined.

1.2 Previous work

Various research projects have investigated the use of SCADA-data to model a condition monitoring system for wind turbines. Many of the methods aim to create a model that learns the complex internal relationships between SCADA signals, such as component temperatures, rotational speed, power produced, electrical quantities etc. These types of models typically reconstructs a SCADA signal from a subset of the other signals, and the residual between the original data and the reconstructed data can be used to detect faults in the wind turbine. The capability to model highly nonlinear relationships makes Deep Neural Networks (DNN) a popular choice. In Reference [4] and [6] DNN models are used for condition monitoring of the gearbox by learning how temperature and oil pressure signals in the gearbox behave. Since these models are focused on the gearbox they will not react to a fault somewhere else in the wind turbine.

In Reference [7] and [8] an Artificial Neural Network (ANN) model for each subsystem of the wind turbine was proposed to get a more complete coverage. The system was successful in detecting a variety of faults, but a deep knowledge of the physical connection of the parts in the wind turbine was needed for a correct signal mapping. Since the wind turbines on the market today are operating in a variety of ways, with different signals being monitored for different turbines, this method is difficult to employ across different types of wind turbines.

The use of a different kind of ANN model called autoencoder (AE) has shown success in detecting developing faults, as shown in Reference [9] and [10]. By learning to reproduce all its inputs, the AE has the capability to model complex relations and to monitor the whole system at once. AE's have successfully been applied in the field of condition monitoring for wind turbines as shown in Reference [11], where the AE was able to detect blade issues in the wind turbine and in Reference [12] and [13] where AE's detected various types of faults, like sensor faults and yaw encoder faults. While these studies have applied the AE to a variety of faults in the wind turbine and successfully shown that the AE can be used to detect the faults, the focus lies in the detection step; to find out if the wind turbine is experiencing a fault. The next step is to identify what signals are responsible for causing the alarm and thereby help determining the root cause of the fault and provide recommendation on where to do maintenance. In Reference [14], the Mahalanobis distance, D^2, on the residual between the input and the output of the AE was used find periods with a potential fault in the wind turbine. In order to diagnose the fault, a method of determining the relative contribution for each separate residual signal to the Mahalanobis distance was proposed. This method calculated the partial Mahalanobis distance, $D^2_{(i)}$, on all residual signals except the i^{th} signal and used the difference $d_i = D^2 - D^2_{(i)}$ to determine the contribution of signal i to the overall Mahalanobis distance. A large contribution suggested that the fault affected that signal a lot. The result of this study suggests that this method might give some information about what the fault in the turbine is, but due to lack of information in the turbine service logs it is unclear whether some of the results are false positives. The authors suggests alternative approaches to dealing with the problem of fault diagnosis and this thesis proposes one such approach.

1.3 Aim and limitations

This thesis aims to develop a condition monitoring system based on an AE that is able to both find periods when the wind turbine operates with a fault and to diagnose what components are faulty, by finding which measured signals are most affected by the fault. Since most of the previous studies based on AE have focused on the fault detection step [11] [12] [13], the main focus in this thesis is the fault diagnosis.

ANNs, and hence AEs, are seen as *black box* models, meaning that it is difficult to analyze why an ANN comes to a certain conclusion or gives a certain result. To understand what the autoencoder has learned, this thesis presents a novel method for examination of what the AE has learned with the help of simulated faults. This provides information about what internal connections between the SCADA data signals the AE has learned as well as information about how the residual (the difference between the input signal and the reconstructed signal) is affected when there is a fault in a certain signal.

The aim with the thesis is as follows:

1. Design a condition monitoring system based on an autoencoder that (i) detects developing faults and (ii) is able to diagnose the fault, i.e. finding the degrading component(s).
2. Examine what internal connections the autoencoder has found in order to understand how the residual between the input and output of the autoencoder behaves.

This thesis will utilize previous studies for the design of the AE, in particular the studies made in Reference [13].

1.4 Structure of the thesis

The thesis is structured as follows:

- **Chapter 2** provides the theoretical background to the methods and models used in this thesis. Basic theory about wind turbines and SCADA data is provided as well as information about artificial neural networks in general and autoencoders in particular. The chapter also provides theoretical information regarding pre- and post-processing of data, such as data transformation and the Mahalanobis distance for multivariate distance measurement.
- **Chapter 3** describes in detail the proposed method of using deep autoencoders for condition monitoring of wind turbines. It explains how the data was pre- and post-processed as well as the design and training of the autoencoder. It also provides the proposed method for condition monitoring of wind turbines with the use of an autoencoder, which is tested in the following chapter.
- **Chapter 4** presents the validation tests that were done to examine the performance of the condition monitoring system proposed in the previous chapter. The first part of the chapter describes the usage of simulated faults to understand what the autoencoder has learned and the knowledge gained from the following discussion is used on real world faults in the second part of the chap-

ter, in which eight faults are examined with the proposed condition monitoring system.

- **Chapter 5** discusses the findings in the thesis and proposes future work. Finally, the thesis is concluded.

2
Theory

This chapter provides the theoretical background to the methods and models used in this thesis. The chapter starts with basic theory about wind turbines and the Supervisory Control and Data Acquisition (SCADA) system and its usage. This is followed by an overview of Artificial Neural Networks (ANN), starting with an explanation of one of the most basic ANNs, the Single Hidden Layer Feedforward Network. This leads to an explanation of the Autoencoder, which is the type of ANN used in this thesis. Theory for preprocessing data, including data transformation, is presented in the following section. After this, two tools that are used to postprocess the data are explained.

2.1 Wind turbines

A wind turbine has multiple parts functioning together. Here follows an overview of how a conventional horizontal-axis wind turbine is operating and a description of its main components. Next follows a description of the SCADA system and its usage.

2.1.1 Basics about wind turbines

The conventional horizontal-axis wind turbine consists of a rotor and a nacelle, which are placed on top of a tower that rests on a foundation. The tower is usually 70 - 120 meters high, to allow for long blades and to capture the faster, less turbulent wind that is present at higher altitudes. The blades are placed on the rotor and their lenght ranges between 20 to 80 meters. The rotor is connected to the low-speed shaft, within the nacelle, which in turn is connected to the high-speed shaft via the gearbox. This increases the rotational speed from about 30-60 RPM to about 1000-1800 RPM, which is the rotational speed required by the generator to produce electricity. The generator is connected to a converter that transforms the electricity to lie within the grid frequency. The electricity is then transported through the grid. To avoid unnecessary strain on the components, the wind turbine operates within the wind speed range of 4-25 m/s. See Figure 2.1 for a sketch of the turbine, with the most important parts marked.
The most high-maintenance part of the wind turbine is the gearbox, which is also the part that is causing most downtime [15]. The many wheels and bearings in the gearbox suffer from great stress because of wind turbulence and a fault in any part of the gearbox may lead to a halt. Due to this, wind turbines without a gearbox

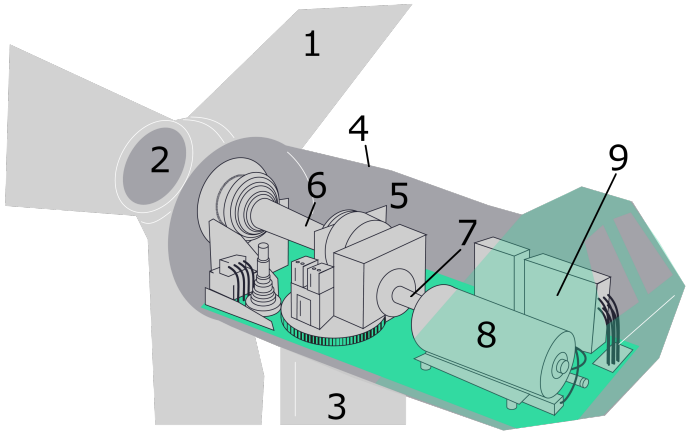


Figure 2.1: A sketch showing many of the important parts of the wind turbine. 1) Blades, 2) Rotor, 3) Tower, 4) Nacelle, 5) Gear box, 6) Low-speed shaft, 7) High-speed shaft, 8) Generator, 9) Converter

are being developed. These wind turbines instead uses direct drive and are said to improve reliability. In this thesis, all the wind turbines examined have a gear box.

2.1.2 SCADA system for wind turbines

The Supervisory Control and Data Acquisition (SCADA) system is an important part of the wind turbine. It is used on one hand to control the wind turbine remotely and on the other hand to collect data to monitor both the current and the historical behaviour of the wind turbine. The data recorded is sensor data collected as 10 minute averages.
Signals that are recorded for the wind turbine are for example wind speed, wind direction and power as well as component specific signals like bearing temperatures and lubrication oil temperatures and pressures. Signals from electrical components, like currents and voltages, are also recorded.
By monitoring these signals, it is possible to gain knowledge about how the turbine is currently operating and to compare it to how it historically has behaved. There are multiple ways to monitor these signals, from setting static thresholds on signals to developing a model of how the data looks for a healthy turbine. This thesis will describe one such model.

2.2 Artificial Neural Networks

In this section a brief overview of artificial neural networks is given, with the aim to give the right background for discussing the autoencoder, which is a type of artificial neural network that is used in this thesis for condition monitoring of wind turbines.

Readers already familiar with the basics of neural networks can skip to Section 2.2.3 to read about the autoencoder.

2.2.1 Introduction

Artificial neural networks are inspired by how the biological brain learns. The brain contains neurons, that are wired together with synapses in complex patterns that are always evolving, allowing the brain to learn. Instead of having to specify a set of rules, the brain learns by examples and experience. This is the mechanism that artificial neural networks are trying to copy, why they consists of artificial neurons that can be wired together to learn. The value of one neuron depends on the value of the neurons connected to it as well as the weight of the connections between the neurons. For a better overview and easier understanding, artificial neural networks are usually visualized as directed graphs. One of the simplest versions of a neural network is the single hidden layer feedforward network, which structure is shown as a directed graph in Figure 2.2 and will be explained in more detail in the following subsection.

2.2.2 Feedforward neural networks

This section aims to explain the basics of one of the most simple forms of neural networks, the single hidden layer feedforward network. Many of the concepts are very similar or identical to how the autoencoder network is used and trained, but is easier explained by the use of this simple neural network.
A graph of a single hidden layer feedforward network is shown in Figure 2.2. This network consists of one input layer, one hidden layer and one output layer. Each layer consists of multiple neurons. The neurons in the input layer are all connected to the neurons in the hidden layer and all the neurons in the hidden layer are connected to the neurons in the output layer. This is called a *fully connected* network. Neurons within a layer are not connected, which is an important aspect of a feedforward network. What connects the neurons are weighted edges, usually just called weigths. When talking about *training* a network, what is meant is the method of finding the optimal values of the weights, and one of the most common ways of doing this is explained in Section 2.2.2.2. When the optimal values of the weights are found, the weights are fixed and will not change.
Information in the network can flow in two ways; forward, which is used for calculating the output of the network, and backward, which is only used during training of the network. In the following two sections both of these will be explained in more detail.

2.2.2.1 Forward propagation

Calculating the output of the network involves first calculating the values of the neurons in the hidden layer, Equation (2.1),

$$V_i = g^{(h)}(\sum_j W_{i,j}^{(h)} x_j + b_i^{(h)}) \tag{2.1}$$

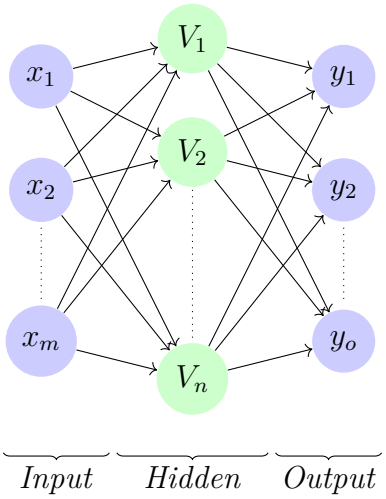


Figure 2.2: Architecture of a single hidden layer feedforward neural network.

where V_i is the value of hidden neuron i, $W_{i,j}^{(h)}$ is the weight between neuron i in the hidden layer and neuron j in the input layer, x_j is the value of input neuron j, $b_i^{(h)}$ is a bias for neuron i in the hidden layer and $g^{(h)}(\dots)$ is the activation function for the hidden layer (see 2.2.2.3 for explanation of activation functions). The superscript $^{(h)}$ specifies that the weights, biases and activation function are specific for the hidden layer.
The output of the network is then calculated in a similar way, Equation (2.2),

$$y_i = g^{(o)}(\sum_j W_{i,j}^{(o)} V_j + b_i^{(o)}) \tag{2.2}$$

where y_i is the value of output neuron i and the superscript $^{(o)}$ specifies that the weights, biases and activation function are specific for the output layer.
If there is more than one hidden layer, the procedure is the same; the values of the neurons in the layers are calculated layer by layer until reaching the output layer.

2.2.2.2 Backpropagation

Backpropagation is a common way of training the network by changing the weights of the network to reach optimal results. The optimal result is reached by trying to minimize a function called the *loss function*. The loss function measures how close the output of the network is to the wanted output of the network (called the *target values*, t_i). In this thesis the mean squared error, Equation (2.3), is used as loss function.

$$L = \frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2 \tag{2.3}$$

The goal is to minimize the loss function, which is done with the use of an optimizer. One common optimizer is gradient descent, which requires the activation functions used on the layers to be differentiable. Gradient descent is used to repeatedly update the weights by adding increments calculated on the error, see Equation (2.4),

where η is the learning rate and the superscript $^{(\mu)}$ specifies the layer. The name *backpropagation* refers to the process of updating the weights closest to the output layer and then propagating the loss *backwards*, updating weights in each layer until reaching the input layer [16].

$$W_{m,n}^{(\mu)} \leftarrow W_{m,n}^{(\mu)} - \eta \frac{\partial L}{\partial W_{m,n}^{(\mu)}} \tag{2.4}$$

2.2.2.3 Activation function

There are many types of activation functions, but for networks that are trained using backpropagation it is important that the activation function is differentiable. Here three types of activation functions are presented; linear, sigmoid and ReLU. The simplest activation function is linear, see Equation (2.5).

$$g(x) = x \tag{2.5}$$

The sigmoid activation function, usually implemented as the logistic function, see Equation (2.6), is often used when the goal is to calculate probabilities, since it outputs a value between 1 and 0.

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

The ReLU activation function [17] is defined as the positive part of its argument, see Equation (2.7). It is the most used activation function in deep learning. ReLU is not differentiable when $x = 0$, but a common convention is to set the derivate to zero at $x = 0$.

$$g(x) = \max(0, x) \tag{2.7}$$

2.2.3 Autoencoder

The autoencoder is the type of artificial neural network that is used in this thesis. It is a feed-forward neural network with a layout that allows the network to learn the most important features of the input signals, to allow for dimensionality reduction and reconstruction of the data. The training of an autoencoder can be seen as an unsupervised or self-supervised training algorithm, meaning that no labeled dataset is used as targets of the network. Instead the network is trained to reproduce its inputs as correctly as possible, which is done by using the input values as targets. The network is trained with the use of back-propagation, with a loss-function depending on both the input and the output of the network; $\mathrm{L}(x, \hat{x})$.
In Figure 2.3 a graph of a *deep* autoencoder is shown. A deep neural network is a network that contains multiple hidden layers, which allows the network to learn both linear and non-linear relationships. The autoencoder can be seen to consist of two sub-networks, where one is called the encoder, since it lowers the dimension of the data. The other sub-network is called the decoder since it reconstructs the data into the original dimension. The encoder and decoder contain the same amount of layers and neurons, but in opposite order. The middle-layer, which is the layer with

the least amount of neurons, is called the code-layer. The bottleneck structure of the autoencoder forces it to learn the most significant features from the input data, to be able to reconstruct it well.
Among other things, autoencoders can be used for non-linear dimensionality reduction and for anomaly detection. In this thesis, they are used for the latter.

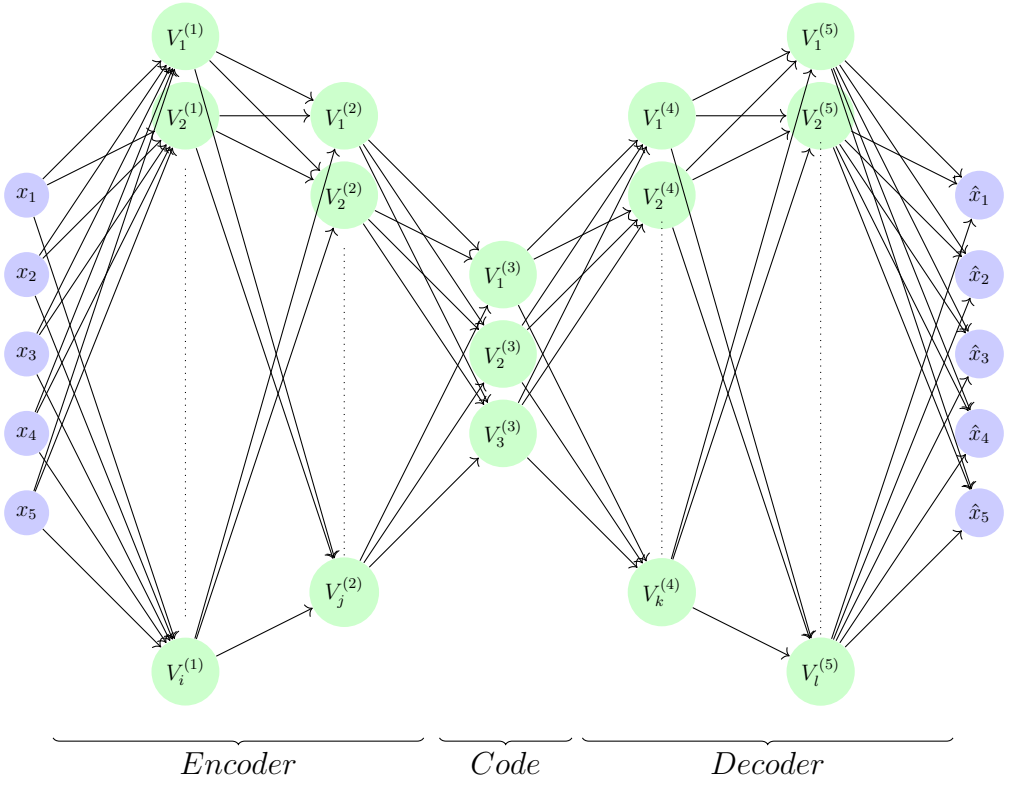


Figure 2.3: Architecture of a deep autoencoder with 5 hidden layers and code size 3.

2.2.3.1 Anomaly detection using autoencoder

When training an autoencoder the model is forced to find the most important features and relations of the input data, to be able to reconstruct the data well. This is due to the bottle neck structure of the autoencoder, where as much information as possible about the data has to be kept in a lower dimension. When training the autoencoder on data from healthy operating conditions it will learn to reconstruct normal data very well, but when faced with data from abnormal working conditions it will fail to do so. By looking at the reconstruction error, called residual, i.e the error between the input data and the reconstructed data, see Equation (3.3), it is possible to find anomalies. More information about how this is done in the application for wind turbines is presented in Section 3.5.1.

2.3 Preprocessing data

Raw data is often incomplete and in need of preprocessing to transform it into a more fitting format. There are multiple important steps included in data prepro-

cessing, such as data cleaning, feature selection and data transformation. During the data cleaning step missing values and outliers are handled, either by removal or by for example interpolation of nearby values. Feature selection is the process of selecting which features to use for the model. This can be done manually, by applying knowledge of which features are important for the model. It can also be done automatically, by applying feature selection techniques such as *Lasso* [18], which forces the coefficients for some of the features to zero, or by feature fusion techniques like *PCA* [19], which transforms the data into a set of uncorrelated *principal components*, allowing a subset of the components to capture a large part of the variance. In this thesis, the feature selection is a combination of manual and automatic selection: Multiple features are manually selected with knowledge of what signals are available and important in the wind turbine. The autoencoder is then performing feature selection and feature fusion automatically, as the architecture of it forces it to only keep important information.

2.3.1 Transforming data

Data transformation is used to ensure that the data is in wanted format, which can differ between applications. This can include removing biases and trends and ensuring that all features have the same amplitude. When features are measured with different units, the amplitude of the values can differ by many orders of magnitude, why it is good to scale it to a particular range. In this thesis, two types of data transformation have been examined; Standardizing data and ZCA whitening.

2.3.1.1 Standardizing data

Standardizing the data ensures that the features are scaled and that some bias is removed. This is done per feature by removing the mean value from the feature and by dividing it by its standard deviation, as seen in Equation (2.8), where $\vec{x}$ represent the original feature and $\vec{y}$ the transformed feature.

$$\begin{aligned} \vec{y} &= \frac{\vec{x} - \mu}{\sigma} \\ \mu &= \text{Mean}[\vec{x}] \\ \sigma^2 &= \text{Var}[\vec{x}] \end{aligned} \tag{2.8}$$

2.3.1.2 ZCA whitening

Whitening is a linear transformation that converts the vector $\vec{x}$ with mean $\vec{\mu}$ and a positive definite covariance matrix Σ_x into a new vector $\vec{y}$ of the same dimension as $\vec{x}$ with identity covariance matrix $\Sigma_y = \text{I}$ according to Equation (2.9), where W is the transformation matrix.

$$\vec{y} = \text{W}\vec{x} \tag{2.9}$$

The transformation matrix needs to fulfill $\text{W}^{\text{T}}\text{W} = \Sigma_x^{-1}$, to fulfill the requirement of $\Sigma_y = \text{I}$. As long as this condition is met, there are infinite many transformation matrices that will whiten $\vec{x}$. ZCA whitening gives one example of a transformation

matrix, that minimizes the total squared distance between the original and whitened variables [20]. The transformation matrix for ZCA is shown in Equation (2.10).

$$\mathrm{W}^{\mathrm{ZCA}} = \Sigma_x^{1/2} \tag{2.10}$$

2.4 Data postprocessing

This section provides the theory of two data postprocessing tools that are used in this thesis: The multivariate distance measure *Mahalanobis distance* and the signal smoothing *Exponentially Weighted Moving Average*. The application of these will be further explained in the next chapter.

2.4.1 Mahalanobis distance

Mahalanobis distance [21] is a multivariate distance measure which measures the distance between a point, $\vec{x}$, and a distribution. It is calculated as shown in Equation (2.11), where $\vec{\mu}$ is a vector of mean values of independent variables from the distribution and Σ^{-1} is the inverse covariance matrix of the independent variables from the distribution. If the true distribution is not known, μ and Σ can be estimated from data.

$$D(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^{\mathrm{T}} \Sigma^{-1} (\vec{x} - \vec{\mu})} \tag{2.11}$$

The Mahalanobis distance can be used as a test statistic for determining the likelihood that a point comes from the distribution. A high Mahalanobis distance means that it is less likely that the point comes from the distribution. When the features are uncorrelated, Mahalanobis distance becomes Euclidean distance with respect to the mean of the data, since the covariance matrix is then the identity matrix.

2.4.2 Exponentially Weighted Moving Average

The exponentially weighted moving average (EWMA) can be used on time series data to lower the importance of temporary spikes and noise and allow for underlying trends to be more easily detected [22]. The EWMA is defined as shown in Equation (2.12), where x is the original data, z is the EWMA of the data and $\lambda \in (0, 1]$ is a constant. A starting value for z is needed for the first iteration and is in this thesis set to the starting value of $\vec{x}$; $z_0 = x_0$.

$$z_i = \lambda x_i + (1 - \lambda) z_{i-1} \tag{2.12}$$

The EWMA is a weighted average of all previous samples where the value of λ decides how important the past values are in relation to the present values. If $\lambda = 0.2$, the weight to the current value is 0.2 and the weights given to the preceding values are 0.16, 0.128, 0.1024 etc. This exponential decay in the weights is the reason why the method is called *exponentially* weighted moving average.

3

Design of a condition monitoring system for wind turbines

This chapter describes in detail the proposed method of using deep autoencoders for condition monitoring of wind turbines. It starts with a description of which SCADA signals that were used for training and testing of the method, followed by an explanation of how the data was cleaned and transformed in the preprocessing step. Thereafter comes a section describing the design and the training of the autoencoder. Next, the postprocessing of the data is explained: How the residual is taken and standardized and how Exponentially Weighted Moving Averaged can be used to find trends in noisy data. The final section explains in detail how the residual is used to find faults in the wind turbine with the use of the Mahalanobis distance as well as an description on how the residual for individual signals can be used for fault diagnosis.

3.1 Data description

The data used to train and test the models in this thesis was SCADA data collected from multiple wind turbines in two different wind farms, see Section 2.1.2 for more information about SCADA data. The data was collected as 10 minute averages where each feature is a signal from a sensor. The signals that were used are listed in Table 3.1.

3.1.1 Training and validation data

For each turbine 12 months of data was collected as training data and the following 6-9 months were used as validation data. This provided a training dataset of about 50000 data points and a validation dataset of $25000 - 40000$ data points per turbine. The data for training and validation should ideally come from a healthy turbine, since the goal with the autoencoder is to learn to correctly reconstruct data during healthy condition. A *healthy* turbine refers to a turbine with no faulty component that is operating as expected during normal conditions. To ensure that the data used in this thesis came from healthy turbines service logs were examined to see if the turbine had experienced any fault during the period.

3.1.2 Inference data

The goal with the training of the autoencoders was to be able to evaluate new data, called inference data. This data was preprocessed the same way as the training and validation data.

Signal name	**Unit**
Power	kW
Wind speed	m/s
Wind direction	°
Generator bearing front temperature	°C
Generator bearing rear temperature	°C
Generator phase 1 temperature	°C
Generator phase 2 temperature	°C
Generator phase 3 temperature	°C
Generator slip ring temperature	°C
Hydraulic oil temperature	°C
Gear oil temperature	°C
Gear bearing temperature	°C
Nacelle temperature	°C
Ambient temperature	°C
Grid inverter temperature L1	°C
Top controller temperature	°C
Hub controller temperature	°C
Spinner temperature	°C
Rotor inverter temperature L1	°C
Rotor inverter temperature L2	°C
Rotor inverter temperature L3	°C
Grid busbar temperature	°C
Gear bearing temperature	°C
Voltage L1	V
Voltage L2	V
Voltage L3	V
Current L1	A
Current L2	A
Current L3	A
Generator RPM	RPM
Rotor speed	RPM
Blade angle (pitch position)	°

Table 3.1: The 32 SCADA-data signals that were used for the turbines examined.

3.2 Data preprocessing

This section explains how the data was preprocessed, the theory behind this process can be found in Section 2.3. The data was handled as tables, in the format of dataframes, meaning that the columns represented the signals and each timestamp was represented as a row. Note that the data was cleaned and transformed *per turbine*, this since it was later used to train or test autoencoders, which is also done *per turbine*.

3.2.1 Data cleaning

If a signal could not be recorded it would show up as a NaN-value in the data. This could happen for longer periods, which could lead to the certain signal not having enough values to be used at all. If any signal would have more than 20% NaN-values, the column containing this signal would be dropped and that signal would be ignored for this turbine. It could also happen temporarily, which would result in few, less than 20%, NaN-values. In this case all data on these timestamps were dropped, to ensure that every timestamp had valid data for all signals.
It was possible to see when the turbine was shut down by looking at the timestamps where power was zero. Since the turbine behaves in a different way when it is shut down, the signals for these timestamps were unwanted. Therefore all timestamps where power was zero was removed, together with 30 minutes before and after the shut down of the turbine, to avoid abnormal signals due to start/stop behaviour.

3.2.2 Transforming data

Two types of transformations were initially examined in this thesis, standardizing data and ZCA whitening. The theory behind these transformations can be found in 2.3.1. Standardizing data was tried since this is one of the most common ways of doing data transformation for neural networks. It ensures that the data is centered around zero and that the signals are scaled appropriately. ZCA whitening takes this one step further, by also decoupling the signals from each other, i.e removing the correlation between the signals. The idea is that removing correlation between signals can help the autoencoder separate the signals from each other and thereby facilitate training.
When transforming the data, the transformation was done with parameters calculated on the training data. For example, when standardizing the data the mean and the variance were needed. These parameters were taken from the training data and when transforming both training, validation and inference data, the same parameters were used.
To examine whether the data should be standardized or ZCA whitened, both types of transformation were used on the simulated faults and the real faults presented in Section 4.1 and 4.2. The results showed that ZCA whitening was less fit to use when the goal is to perform fault diagnosis. The autoencoders trained and applied on ZCA transformed data provided less information about the fault than what standardizing the data did. When trying to analyze what the autoencoder

had learned by creating the *connection map* explained in Section 4.1.2, it was clear that the autoencoder trained on ZCA whitened data had learned relationships that made it difficult to use for fault diagnosis. One explanation for these results is that when ZCA whitening the data, information is actually removed from the data and is stored in the transformation matrix instead, which means that there is less information in the ZCA whitened data to draw conclusions from. Appendix A.1 provides more details about test cases, results and further discussion of this. From these conclusions it was clear that the data should be standardized for best result, which is the transformation that has been used in this thesis.

3.3 Autoencoder

The thesis *Condition monitoring system for wind turbines* by Renström [13] examined the use of autoencoders for anomaly detection in wind turbine data. Different autoencoder designs and hyperparameters[1] were examined and the best performing model has been reproduced in this thesis, for continued examination. This section describes the design and training of the autoencoder, including what hyperparameters were used. For a thorough examination of the design process the reader is referred to Reference [13].

3.3.1 Model design

The layout of the autoencoder is shown in Equation (3.1), where the numbers represent neurons per layer. $n_{signals}$ is the number of available signals from Table 3.1 (see Section 3.2.1 for an explanation why not all signals might be available). If all signals were available $n_{signals} = 32$.

$$\underbrace{n_{signals} \times 144 \times 96 \times 64}_{\text{Encoder}} \underbrace{\times 18 \times}_{\text{Code}} \underbrace{64 \times 96 \times 144 \times n_{signals}}_{\text{Decoder}} \tag{3.1}$$

The activation function used between each layer was ReLU [17], except for the last layer which had a linear activation function. Batch normalization was used, which is a way of standardizing the output of each layer to increase the stability of the neural network [23]. The weights were initialized from a uniform distribution. The autoencoder was implemented and trained using the programming language Python 3.7 with the package PyTorch (`torch` version 1.2.0) [24].

3.3.2 Training the autoencoder

When training the autoencoder the standardized training and validation data for the turbine of interest was used. It was trained using backpropagation with Mean Squared Error (MSE) as loss function. This meant that the output of the autoencoder was calculated using the training data, then the MSE between the reconstructed and the training data was used to update the weights, using a version of

[1] Hyperparameters in Machine Learning are parameters that are set before training, for example the size of the network or the learning rate.

gradient descent called ADAM [25]. The MSE for the validation data was also calculated, but this error was never used to update weights. Instead it was used as a criteria for early stopping, meaning that the training would stop if the autoencoder started to overfit to the training data and thereby worsen the reconstruction of the validation data. The training was done using batch training, with batch size 256, which combined with the optimizer ADAM is a preferred way of training deep neural networks since it lowers the risk of getting stuck in a local minima.

3.4 Data postprocessing

When the autoencoder has been trained it can be used to examine new data, called inference data, coming from the same turbine as it was trained on. The inference data, $\vec{x}_{inf}$, needs to be transformed using the same transformation as was done on the training data, producing $\vec{y}_{inf}$. It is thereafter used as input to the autoencoder which in turn produces a reconstruction, $\hat{\vec{y}}_{inf}$. This reconstructed data is then inverse-transformed to its original format, $\hat{\vec{x}}_{inf}$. This process is described in Equation (3.2).

$$\vec{x}_{inf} \longrightarrow \vec{y}_{inf} \longrightarrow \text{Autoencoder} \longrightarrow \hat{\vec{y}}_{inf} \longrightarrow \hat{\vec{x}}_{inf} \tag{3.2}$$

The difference between the original data and the reconstructed data is called the *residual* and is examined further with the goal of using it for fault detection and diagnosis.

3.4.1 Residual

The residual is the difference between the original data and the reconstructed data, see Equation (3.3).

$$\text{res}(\vec{x}, \hat{\vec{x}}) = \vec{x} - \hat{\vec{x}} \tag{3.3}$$

The residual is close to zero for signals that are reconstructed well and large for signals not reconstructed well. Some signals might always be better or worse reconstructed than the other signals due to what the autoencoder has learned. This makes it hard to compare the residual from one signal to another signal. To make comparison of residuals between the signals possible, the residual is standardized with respect to the validation data.

3.4.1.1 Standardizing residual

To make it possible to compare the residual for each signal, the residual has to be standardized with respect to how well the signal is usually reconstructed. For this reason, the validation data is used as a benchmark for how well the reconstruction usually is for each signal. The residual for the inference data is transformed according to Equation (3.4), where $\hat{\vec{\text{res}}}_{inf}$ is the transformed residual for the inference data. When discussing the residual in the following sections and chapters of this thesis, it is always the standardized residual that is used.

$$\begin{aligned} \hat{\vec{\text{res}}}_{inf} &= \frac{\vec{\text{res}}_{inf} - \mu}{\sigma} \\ \mu &= \text{Mean}[\vec{\text{res}}_{val}] \\ \sigma^2 &= \text{Var}[\vec{\text{res}}_{val}] \end{aligned} \tag{3.4}$$

3.4.2 Exponentially Weighted Moving Average

The residual from the autoencoder can be noisy even when it is calculated on data from a healthy wind turbine. Instabilities in the grid, wind gusts and other environmental events are usually very short lived, but can still cause high spikes in the residual. To not let these temporary events affect the condition monitoring system it is necessary to process the residual. Real faults in the wind turbine are more long-lived and should not be affected by a method that mainly targets short lived faults. One way of doing this is to use the exponentially weighted moving average (EWMA), that was described in Section 2.4.2. This method removes noise and allows for easier detection of underlying trends. If the underlying trend shows a large residual, it is more likely that this is due to a real fault in the wind turbine. In Figure 3.1b, the Mahalanobis distance of a residual is shown before and after applying EWMA.
To calculate the EWMA the parameter λ is needed, which decides how important previous samples are in relation to the current sample. It was set to $\lambda = 0.004$, which corresponds to a response time of approximately 42 hours.

3.5 Condition monitoring for wind turbines

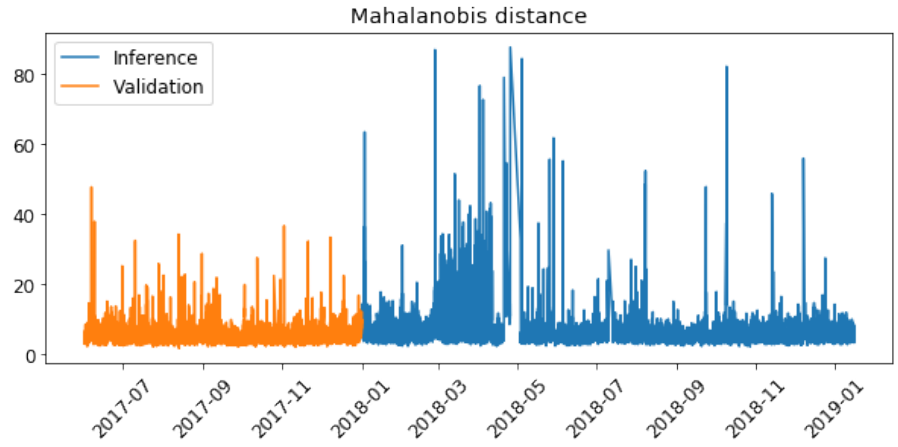
The condition monitoring system for wind turbines that is proposed in this thesis consists of two parts; detecting time periods when the wind turbine is operating with a developing fault and diagnosing which signals are affected by the fault. This is examined in the next chapter, first on simulated faults in Section 4.1 and then on data from turbines with known faults in Section 4.2.

3.5.1 Fault detection

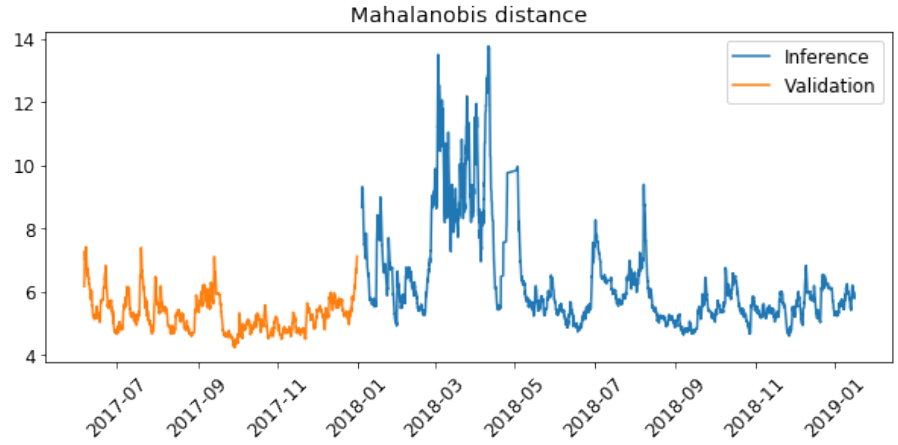
To detect when the wind turbine is operating in an abnormal way is crucial for a good condition monitoring system. The residual produced from the autoencoder can be used to find out how well the autoencoder has reproduced the signals. Since the autoencoder is trained on healthy data it has learned the relations between the signals for healthy conditions. If the residual is larger for new data than it is for known healthy data, there is potentially a fault in the wind turbine. To detect if the residual is larger than normal, the Mahalanobis distance can be used as a metric.

3.5.1.1 Mahalanobis distance

The Mahalanobis distance (described in 2.4.1) measures the distance between a point and a distribution and can be used to determine the likelihood that a point is from



(a) The Mahalanobis distance of the validation residual and inference residual before applying EWMA.



(b) The Mahalanobis distance of the validation residual and inference residual after applying EWMA with $\lambda = 0.004$.

Figure 3.1: A comparison of how a signal can look before and after applying EWMA with $\lambda = 0.004$.

the distribution. When applied to anomaly detection this translates to determining if a point is an anomaly or not.
When the Mahalanobis distance is applied to the residual of an autoencoder, the residual of the validation data is used to estimate the distribution, since this is how the residual is expected to look when it comes from healthy conditions. The Mahalanobis distance of the inference data is calculated for each timestamp according to Equation (2.11), with the covariance matrix and mean vector estimated from the residual of the validation data.
By setting a threshold on the Mahalanobis distance it is possible to classify timestamps from the inference data as anomalies. Since the Mahalanobis distance is noisy, mainly due to noise in the original data, the threshold might be passed for short time periods even when there is no fault present in the data. To ensure the threshold is only passed when there is an underlying trend of a large Mahalanobis distance, the Mahalanobis distance is averaged using EWMA (see Section 2.4.2) before applying

the threshold, as seen in Figure 3.1. The value of the threshold can be determined by using datasets with known healthy and faulty time periods for the wind turbine. Thereafter the threshold can be varied to find the threshold that gives the highest amount of true positives (alarms for real faults), while minimizing the number of false positives (alarms for no fault). The best result is often task-specific, where the importance of finding all faults is weighted against having few false alarms. This was examined extensively in Reference [13], and setting thresholds on the Mahalanobis distance will not be further examined in this thesis.

3.5.2 Fault diagnosis

It is usually not enough to only detect that the wind turbine is operating with a developing fault. Without knowledge about where in the turbine the fault is located it is difficult to give recommendations on how to solve the problem. When a component is operating with a fault, the signals measured from, or in connection to, that component are affected. This section presents the proposed way of diagnosing which signals are affected the most by the fault, which in turn can help in finding the source of the fault.
When examining which signals are affected the most by the fault, the residual of the signals are kept separated, as opposed to using the Mahalanobis distance to merge them together. To allow for comparison between the residual signals, the residual is standardized as described in 3.4.1.1. The idea is that signals not affected by the fault will have a small residual, while signals affected should have a large residual. This assumes that the residual for data from a healthy wind turbine approximately follows a Gaussian distribution, which was tested and shown to be a valid approximation, as can be seen in Appendix B.

3.5.2.1 Boxplot for fault diagnosis

How the separate signals are affected by the fault can be seen by looking at the standardized residual for each separate signal. In order to compare the affect on the different signals, the mean value of each residual is taken over a time period defined as the test period. These mean values can then be compared to find out which signals were affected the most. To make sure that the results gained from such a study is not due to something just one autoencoder has learned, multiple autoencoders with the same hyperparameters were trained on the same data. Due to randomness in the initialization of the weights and in the training of the network, each autoencoder could produce slightly different results. Each autoencoder was therefore used to produce a residual, on which a mean value was taken per signal. These sets of mean values per signal were then compared by using a boxplot to plot the result. The resulting boxplot both show how similar the results are between different autoencoders as well as give a majority vote to what signals were most affected by the fault.
To clarify how a boxplot is created, here follows an example of how to calculate the values needed for one box. In this example 10 autoencoders with the same hyperparameters have been trained for the same turbine and all 10 autoencoders have been used to create 10 sets of residuals. The specific residual signal examined

in this example is the residual for *Power*. There are 10 residuals for the power-signal, where each residual is a time series over how the residual for the power varies for the specified test period. For each residual, take the mean value of the power over the test period, resulting in 10 mean values for the power, one for each autoencoder. These 10 values can now be used to create one box, as in Figure 3.2. The box is created by using a function in the programming language Python (the function comes from the package Pandas; `pandas.DataFrame.boxplot` [26]) and the box is defined as follows: The line in the middle of the box is the median of the values — yes, that is a median of means in this case— and the outer edges of the box are the 25th and 75th percentile of the values, which are called Q1 and Q3 respectively. The lines extending from the box are called whiskers and are calculated by using the length of the box, the *Inner Quartile Range*: $IQR = Q3 - Q1$. The upper line extends to $Q3 + 1.5IQR$ and the lower line to $Q1 - 1.5IQR$. If there are any values outside of the whiskers, they are represented as circles and are considered outliers. This is done for all signals, to create multiple boxes and to be able to compare residual signals between each other.

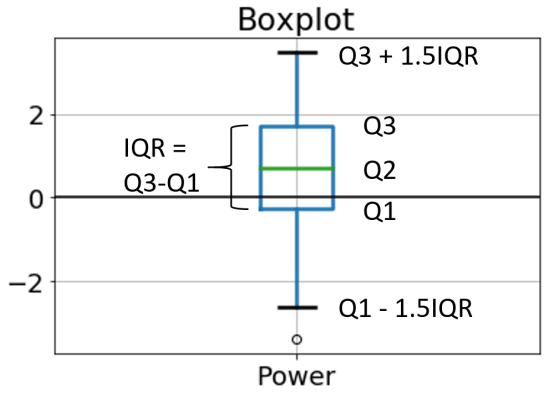


Figure 3.2: Example of a boxplot with one signal. The median, Q2, is represented by the green line inside the box. The lower and upper edge of the box, Q1 and Q3 are determined by the 25th and 75th percentile of the values used to create the box. The whiskers are calculated from the values of Q1 and Q3, as displayed in the figure. The circle represents an outlier.

4

Validation study

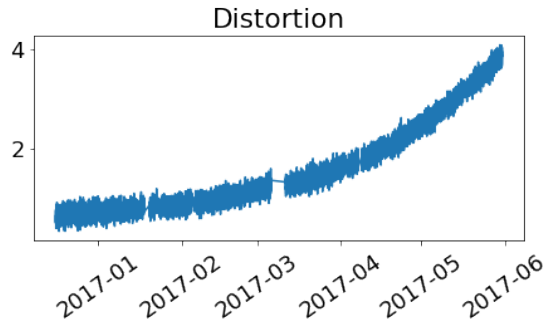
This chapter consists of two main sections, both aiming to test the method for condition monitoring that was described in the previous chapter. The first section describes the method and results of using simulated faults to examine what the autoencoder has learned. The results in this section explains what connections between data signals the autoencoder has found and provides information about how to interpret the residual for each signal. This information is valuable for the next section, in which eight test cases from real world faults are examined with the proposed condition monitoring method.

4.1 Decoding the autoencoder

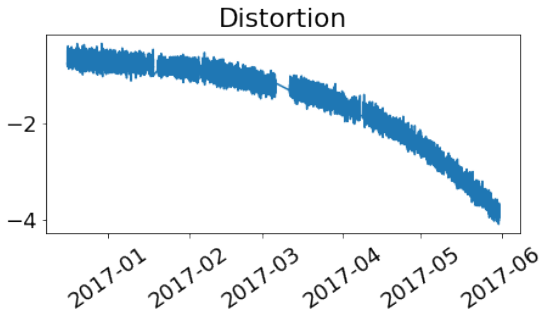
The tests described in this section aim to examine how the autoencoder reacts when one signal experiences a fault and how that in turn affects the reconstruction of the data. This helps interpreting the residual and understanding which internal connections between the signals the autoencoder has learned. To do this, simulated faults were introduced to data from a healthy turbine. The data used for this was the same data as the training data, which in the following subsections is called the *original* data. Simulated faults are distortions manually added to the data in the preprocessing step. An advantage of manually adding faults is that the exact fault is known, which is not the case for the real world faults examined in the following section, 4.2.

Two types of simulated faults were examined, one was generated as an exponential function and was used to determine what happens when the fault added is positive or negative, see Section 4.1.1. The other type of fault was a positive constant, proportional to the standard deviation of the signal to be distorted. This fault was used to compare how adding the fault to different signals would affect the other signals and is described further in Section 4.1.2.

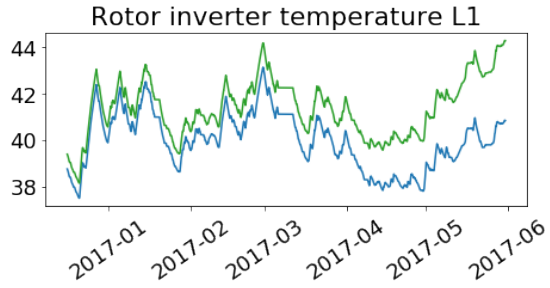
The simulated faults used in this section only affect one signal at a time and thereby differ from the types of faults a turbine would normally experience, which usually affect multiple signals. The exception would be sensor faults, where one sensor would produce a faulty signal that could be similar to the simulated faults discussed here.



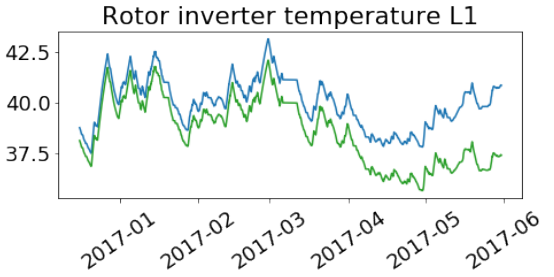
(a) Positive simulated fault, generated as shown in Equation (4.1).



(b) Negative simulated fault, generated by multiplying the distortion in Equation (4.1) by -1.



(c) The original signal (blue) and the distorted signal (green). The distorted signal is created by adding the distortion shown in Figure 4.1a to the original signal.



(d) The original signal (blue) and the distorted signal (green). The distorted signal is created by adding the distortion shown in Figure 4.1b to the original signal.

Figure 4.1: A positive and a negative simulated fault added to the original signal *Rotor inverter temperature L1*. The signals have been smoothed with EWMA for visualisation.

4.1.1 Positive vs negative fault

This section describes how the residual is affected by adding a positive or a negative distortion to one signal, making the signal either higher or lower than original. This is done in order to examine if it is possible to find out what signal was distorted by looking at the residual and to see if the autoencoder has learned to find connections between signals.

4.1.1.1 Method

A positive distortion was generated as shown in Equation (4.1), where z_i is the distortion at timestamp i and x_i is taken from a set of evenly spaced values between 5 and 10 (this set is generated using the function `numpy.linspace` in the programming language Python [27]). The positive distortion is displayed in Figure 4.1a. A negative distortion was generated by multiplying the positive distortion by -1 and is displayed in Figure 4.1b.

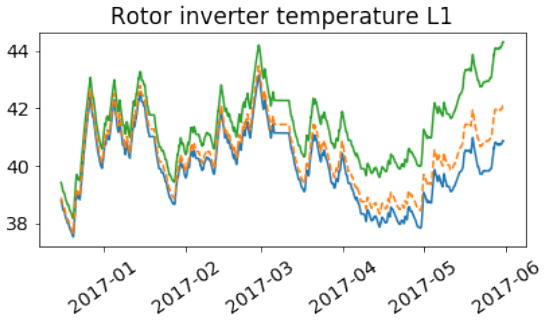
$$z_i = 0.3 + \frac{2^{x_i}}{300} + \mathcal{N}(0.2, 0.1) \tag{4.1}$$

The positive distortion was added to the signal *Rotor inverter temperature L1* in the original data. This would symbolize a turbine operating with too high *Rotor inverter temperature L1*. For comparison, the negative distortion was also added to the signal. The original and the distorted signal for both faults are shown in Figure 4.1c and 4.1d.
After the data was distorted, it was standardized and inserted into a trained autoencoder. The autoencoder reconstructed the data and after the reconstructed data was inverse transformed back to original shape, the residual was calculated and standardized as explained in Section 3.4.1.
The results for the distorted signal and two other signals, *Rotor inverter temperature L2* and *Rotor inverter temperature L3*, were then analyzed together by plotting the original, distorted and reconstructed signals as well as the residual, for the three signals. These signals were analyzed together since they are measured on the same component and thereby vary together when everything is working as expected. By just distorting one of the three signals, it is possible to see if the autoencoder has learned the underlying relationship between the signals and how that affects the residual of the signals.
Due to randomness in the training of the autoencoder, two autoencoders with the same hyperparameters trained on the same data could produce different results. Therefore, to see that the results were not unique to just one autoencoder, 10 autoencoders, with the same hyperparameters, were trained on the same training data. The data was then distorted as described above and reconstructed using the 10 autoencoders. This allowed for a boxplot for the positive and the negative fault to be created, as described in 3.5.2.1. The boxplot also provides an overview of how the distortion affects all signals and not just the rotor inverter temperatures.

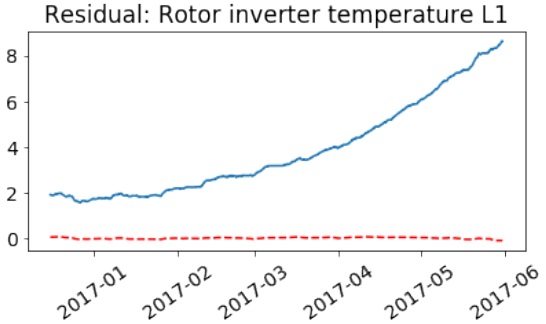
4.1.1.2 Results and discussion

Figure 4.2 shows the affect on the signals *Rotor inverter temperature L1, L2* and *L3* when distorting *Rotor inverter temperature L1* by a positive distortion. The residual is clearly positive for the distorted signal, while it is negative for the other two signals. This result could be interpreted as a large positive residual implies a fault connected to this signal. To test this assumption, the same signal was distorted with a negative residual, with the result shown in Figure 4.3. From the result in this figure it is clear that *Rotor inverter temperature L2* and *L3* have large positive residuals while the distorted signal has a large negative residual. If the assumption is correct, this result would mean that *Rotor inverter temperature L2* and *L3* are the distorted signals, which is wrong. This means that it is hard to draw conclusion on what signal is behaving in an abnormal way by just looking at which signal has the largest positive residual.
Another way to interpret the residuals is to view all residuals, positive and negative, as a result of the fault the wind turbine is experiencing. A faulty component will affect all the signals directly or indirectly connected to it, which will be seen in the residual if the autoencoder has correctly learned how the signals are connected. The results in Figure 4.2 and 4.3 show that the autoencoder has learned that the signals *Rotor inverter temperature L1, L2* and *L3* vary together during normal conditions. The autoencoder reconstructs each rotor inverter temperature signal from the input

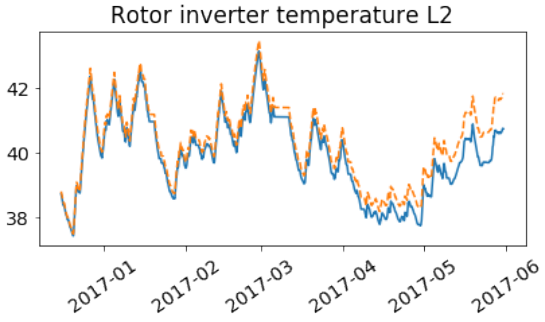
of all three signals. As can be seen in Figure 4.2 the reconstructed signal for *Rotor inverter temperature L1* is higher than the original signal, due to it trying to reconstruct the distorted signal, but lower than the distorted signal, since it is created from a function also including *Rotor inverter temperature L2* and *L3*, which are lower than the distorted signal. The same is seen by looking at the reconstructed signals for *Rotor inverter temperature L2* and *L3*, which are higher than their respective original signal, due to the reconstruction also including the higher value of the distorted signal. This is why the residual is positive for the distorted signal and negative for the other signals. A result as the one shown in Figure 4.2 could be interpreted as *Rotor inverter temperature L1* being too high compared to *Rotor inverter temperature L2* and *L3* according to what the autoencoder knows. The same reasoning can be used on the negative distortion in Figure 4.3.
To compare the results from multiple autoencoders and to visually show how the residual for all signals, not just the rotor inverter temperatures, are affected, the boxplot method explained in Section 3.5.2.1 was used on the data from the two distortions as well as on the original healthy, data for comparison. The result for the positive distortion is shown in Figure 4.4, for the negative distortion in Figure 4.5 and for the original healthy data in Figure 4.6. These figures show, apart from the same result as discussed above, that the distortion also affects *Power* and *Current L1, L2* and *L3*. For the positive distortion the residual for these signals is slightly lower than zero and for the negative distortion it is slightly higher than zero . This implies that the autoencoder expects the wind turbine to produce more power when the rotor inverter temperature is high and less when it is low. More power means higher currents, which the autoencoder also have learned.
This discussion leads to two important conclusions: (i) The autoencoder can learn relationships between signals which can be explained by how the signals are physically connected and (ii) the residual for the signals connected to the distorted signal, and the distorted signal itself, were affected the most by the distortion. This means that the signals with the largest residual should be analyzed when trying to diagnose the fault. A fault in a component will likely show as large residuals for all signals measured directly on the faulty component, but could also show on the residual for signals that are indirectly connected through relationships learned by the autoencoder.



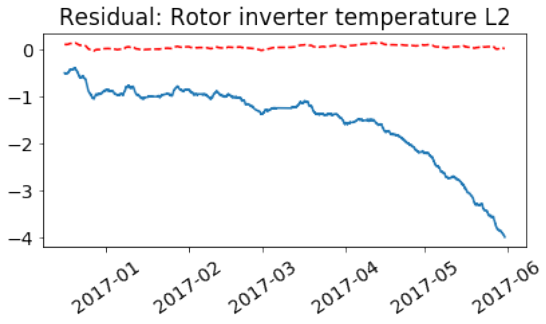
(a) Original signal (blue), distorted signal (green), reconstructed signal by autoencoder (dashed orange).



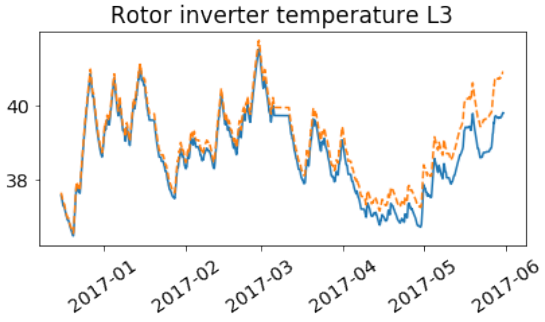
(b) Residual between the distorted signal and the reconstructed signal, with distortion (blue) and without distortion (dashed red).



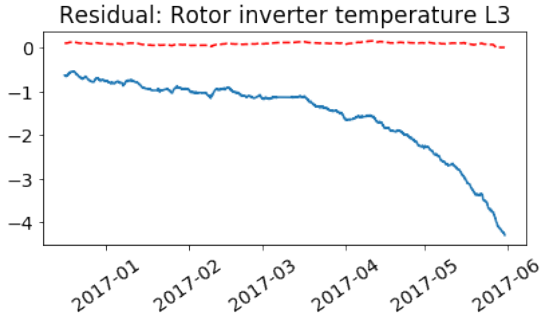
(c) Original signal (blue), reconstructed signal by autoencoder (dashed orange).



(d) Residual between the original signal and the reconstructed signal, with distortion (blue) and without distortion (dashed red).

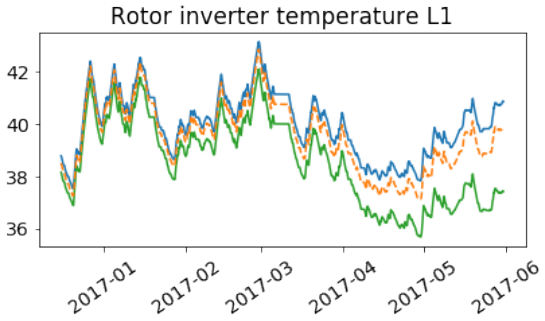


(e) Original signal (blue), reconstructed signal by autoencoder (dashed orange).

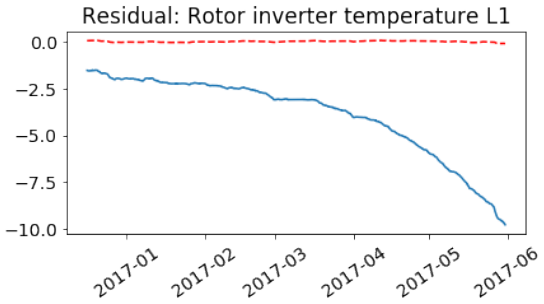


(f) Residual between the original signal and the reconstructed signal, with distortion (blue) and without distortion (dashed red).

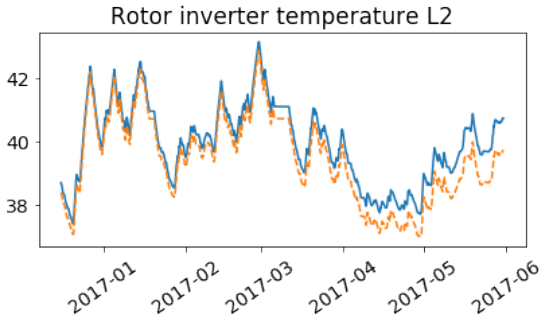
Figure 4.2: The effect on the rotor inverter temperatures from adding a positive exponential distortion to the signal *Rotor inverter temperature L1*, as shown in Figure 4.1c. The residual for the simulated fault is compared with the residual when there was no distorted signal in figure b), d) and f). All signals have been smoothed with EWMA for visualisation.



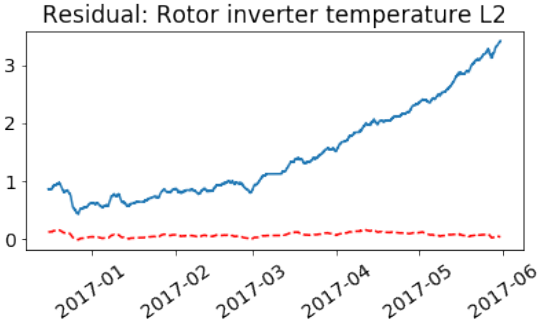
(a) Original signal (blue), distorted signal (green), reconstructed signal by autoencoder (dashed orange).



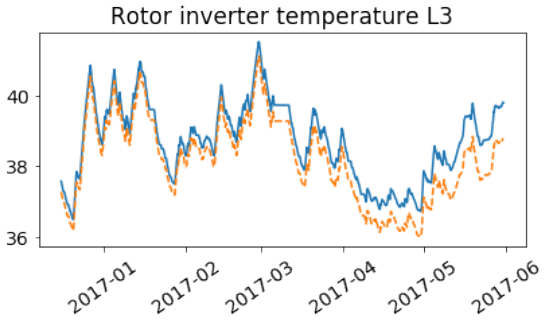
(b) Residual between the distorted signal and the reconstructed signal, with distortion (blue) and without distortion (dashed red).



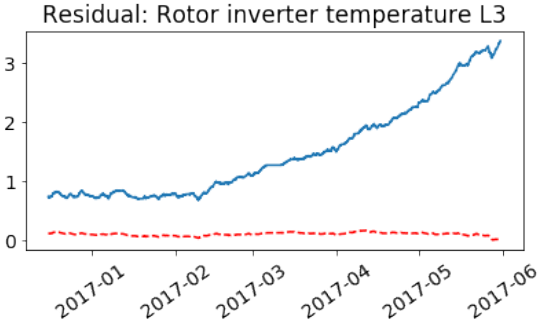
(c) Original signal (blue), reconstructed signal by autoencoder (dashed orange).



(d) Residual between the original signal and the reconstructed signal, with distortion (blue) and without distortion (dashed red).



(e) Original signal (blue), reconstructed signal by autoencoder (dashed orange).



(f) Residual between the original signal and the reconstructed signal, with distortion (blue) and without distortion (dashed red).

Figure 4.3: The effect on the rotor inverter temperatures from adding a negative exponential distortion to the signal *Rotor inverter temperature L1*, as shown in Figure 4.1d. The residual for the simulated fault is compared with the residual when there was no distorted signal in figure b), d) and f). The signals have been smoothed with EWMA for visualisation.

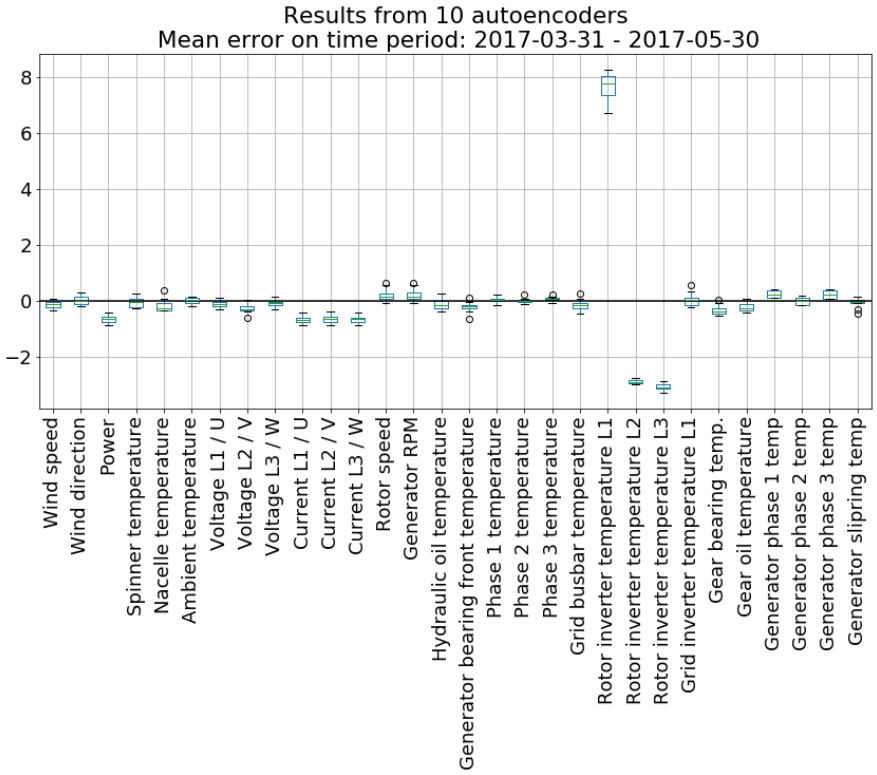


Figure 4.4: Boxplot from the results of 10 autoencoders on the positive exponential distortion of the original signal *Rotor inverter temperature L1*, as shown in Figure 4.1c.

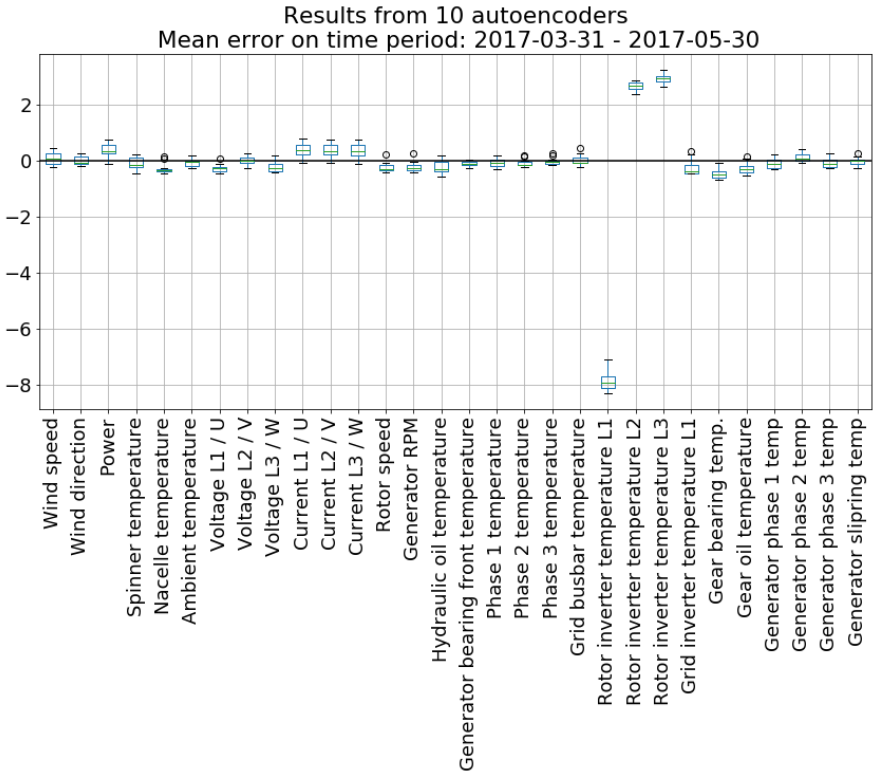


Figure 4.5: Boxplot from the results of 10 autoencoders on the negative exponential distortion of the original signal *Rotor inverter temperature L1*, as shown in Figure 4.1d

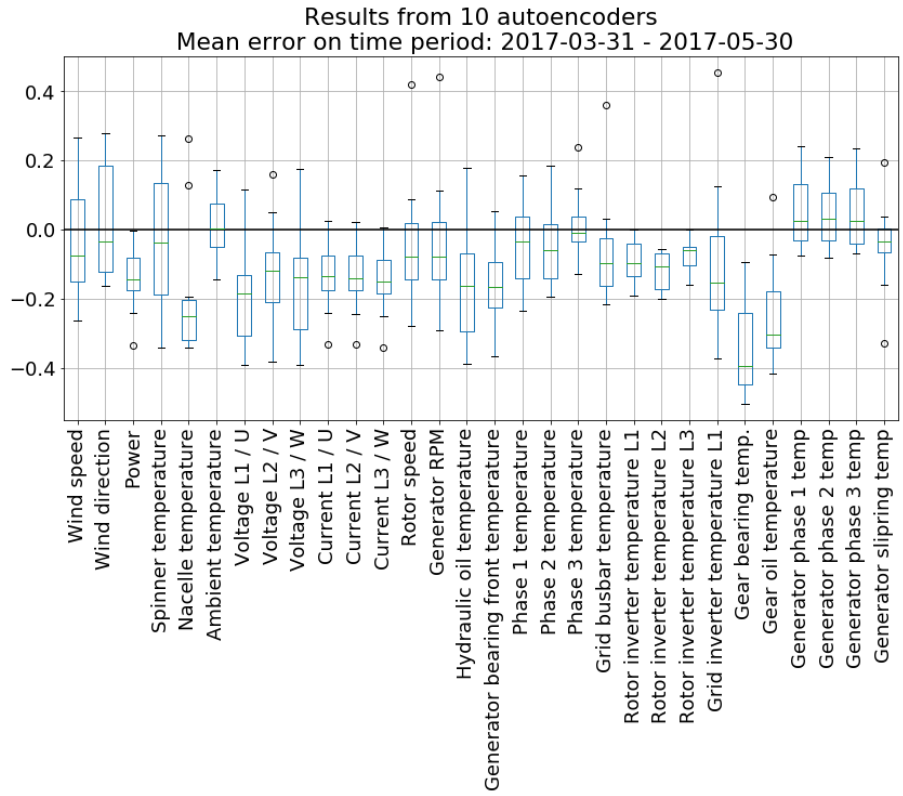


Figure 4.6: Boxplot from the results of 10 autoencoders on healthy data.

4.1.2 Connection map

The results in the previous section showed that by distorting one signal, it is possible to see what other signals are affected. It showed that the autoencoder had learned that *Rotor inverter temperature L1, L2* and *L3* change together and that they in turn also affect the signals *Power* and *Current L1, L2, L3*. The aim of this section is to expand this into understanding how each signal affect all the other signals. As before, the data distorted was the data that was used to train the autoencoder and is referred to as the *original data* in the following text and plots.

4.1.2.1 Generating the connection map

In a similar manner as when distorting the signal *Rotor inverter temperature L1*, in Section 4.1.1, a simulated fault was added to one signal while the other signals were unchanged. The effect on the residual for this simulated fault was recorded as the mean value of the residual for each signal over a certain time period, resulting in one mean value per signal. The data was then reset to original and another signal was distorted. This was repeated until all signals were distorted once. The result of this is multiple sets of mean values, where each set belong to what signal was distorted. This can be displayed as a matrix, where the sets are inserted as column vectors, resulting in a $n \times n$ matrix (where n is the number of signals) were each column tells what signal was distorted and the rows tell how the distortion affected the signal on each row. To make sure the results are not unique to one autoencoder, this was done for the 10 autoencoders that were trained on the original data and the results is a matrix like the one described above, where the values are the means of the result from the 10 autoencoders.
The fault that was added to each signal was a positive translation of the signal that depended on the standard deviation σ of the signal, see Equation (4.2), where $\vec{x}_i$ is the original data for signal i and $\hat{\vec{x}}_i$ is the distorted signal i. In Figure 4.7 such a distortion is shown.

$$\begin{aligned} \hat{\vec{x}}_i &= \vec{x}_i + \sigma_i \\ \sigma_i &= \sqrt{\mathrm{Var}[\vec{x}_i]} \end{aligned} \tag{4.2}$$

4.1.2.2 Analysis of the connection map

The resulting matrix, called a *connection map*, can be seen in Figure 4.8. It is coloured according to what values are high (red) or low (blue). The colouring is done per column, meaning that the highest value in each column is dark red and the lowest is dark blue. Since the colouring is done per column, the colours should not be compared between the columns. In the connection map in Figure 4.8 it is clear that the largest value in each column belongs to the signal that was distorted, which is shown as a diagonal of dark red values in the map. This result means that when adding a positive distortion to one signal, the same signal has the largest positive residual. It is also possible to see what other signals are affected by a distortion.

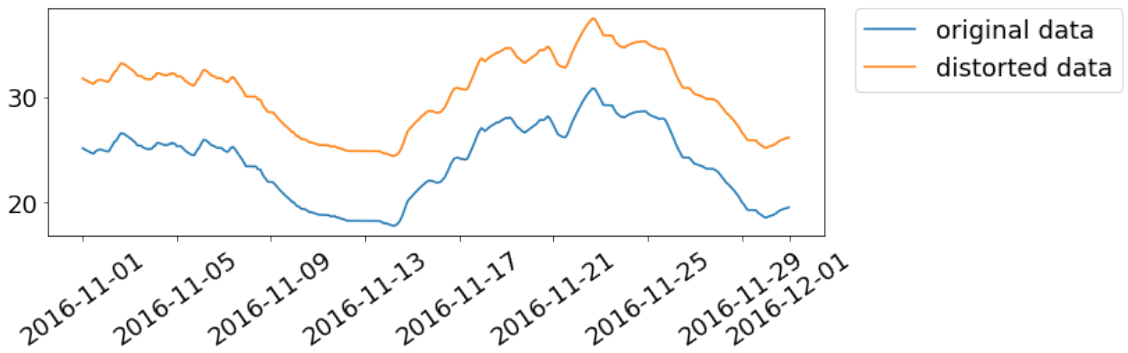


Figure 4.7: Distortion of a signal by using Equation (4.2). The lower blue line is the original signal and the upper orange line is the distorted signal.

If the mean value for a residual from a certain signal is non-zero it means that the autoencoder has found a connection between this signal and the distorted signal. When the value is negative, it means that the autoencoder expects the two signals to be proportionally connected (high together or low together) when the wind turbine is healthy, as was shown and discussed in the previous section for the signals connected to the rotor inverter. The opposite is true when the value is positive, meaning that the signals are inversely proportional.
The result that was shown in the previous section can be seen by looking at column *Rotor inverter temperature L1*. The mean residual value is large and positive for *Rotor inverter temperature L1* and large but negative for *Rotor inverter temperature L2* and *L3*. Note that the values differ compared to the values shown in the previous section, since the signals were distorted with different distortions.
By looking at the columns for *Current L1, L2* and *L3* it is visible that they are affecting both each other and *Power*. The same is seen in the column for *Power*; when distorting *Power*, the currents are also affected. For these distortions, the mean residual values for the discussed affected signals are negative, except for the distorted signal. This means that they are proportionally connected, a result that confirms physical relations.
The *Generator bearing front temperature* seems to be related to the *Generator phase 1-3 temperatures*, which is also expected since they are all measured on the generator. Column *Ambient temperature* shows that when the ambient temperature is high, the residual for *Generator slipring temperature* is negative, meaning that those two temperatures usually vary together. What is notable in this column is that the residuals for *Generator phase 1-3 temperature* are also positive when the ambient temperature is high which is probably due to their relation to the *Generator slipring temperature*. This relationship can be seen in the column for generator slipring temperature where a high *Generator slipring temperature* results in negative residual for *Generator phase 1-3 temperature*.
As shown by this discussion, the connection map can be used to analyze what the autoencoder has learned. It shows which signals are connected in the autoencoder and shows that it seems to have learned relations that can be explained physically in the turbine. The connection map could be further investigated, by for example trying out different distortions, changing the amplitude of the distortion or distorting

multiple signals at once. Since the autoencoder is highly non-linear, the relations shown in this sections might change for other types of distortions. This would help in understanding how stable the results in the connection map is as well as potentially give more knowledge about what the autoencoder has learned. This will not be further investigated in this thesis, but is left as a suggestion for future work.

4.1.3 Conclusion on simulated faults

The analysis of simulated faults helps in understanding how the autoencoder reacts to faults in one signal and how that affects the reconstruction of the data. It has been shown that a fault in one signal affects the reconstruction of multiple signals, which is seen as large amplitude of the residual for affected signals. It has also been shown that when a signal is too high, the residual for that signal is positive and large, while the residual for proportionally connected signals usually is large but negative. The opposite is also true, a too low signal results in a large negative residual for the distorted signal and a large positive residual for proportionally connected signals. This shows that it is hard to know if it is one signal being too low or another signal being too high that is the source of the fault. It suggests that rather than telling that one signal is faulty, the analysis of the residual tells which signals are affected by the fault.
The *connection map* in Section 4.1.2 helps understanding what internal connections between the signals the autoencoder has learned. Many of the connections can be explained by physical relations, which shows that the autoencoder has learned real relationships. This method of explaining what the autoencoder has learned does give some insight, but it is important to remember that the autoencoder is highly non-linear because of its deep architecture. When distorting signals together, or by other amounts, other relationships could arise, that are not visible in the current connection map.
The simulated faults discussed have been added to only one signal at a time, which is usually not how a fault would look like in the real world, unless it is a sensor fault. To know if the autoencoder can be used to find real world faults, and how those would affect the residual, data from known faults needs to be examined. This will be done in the following section, 4.2.

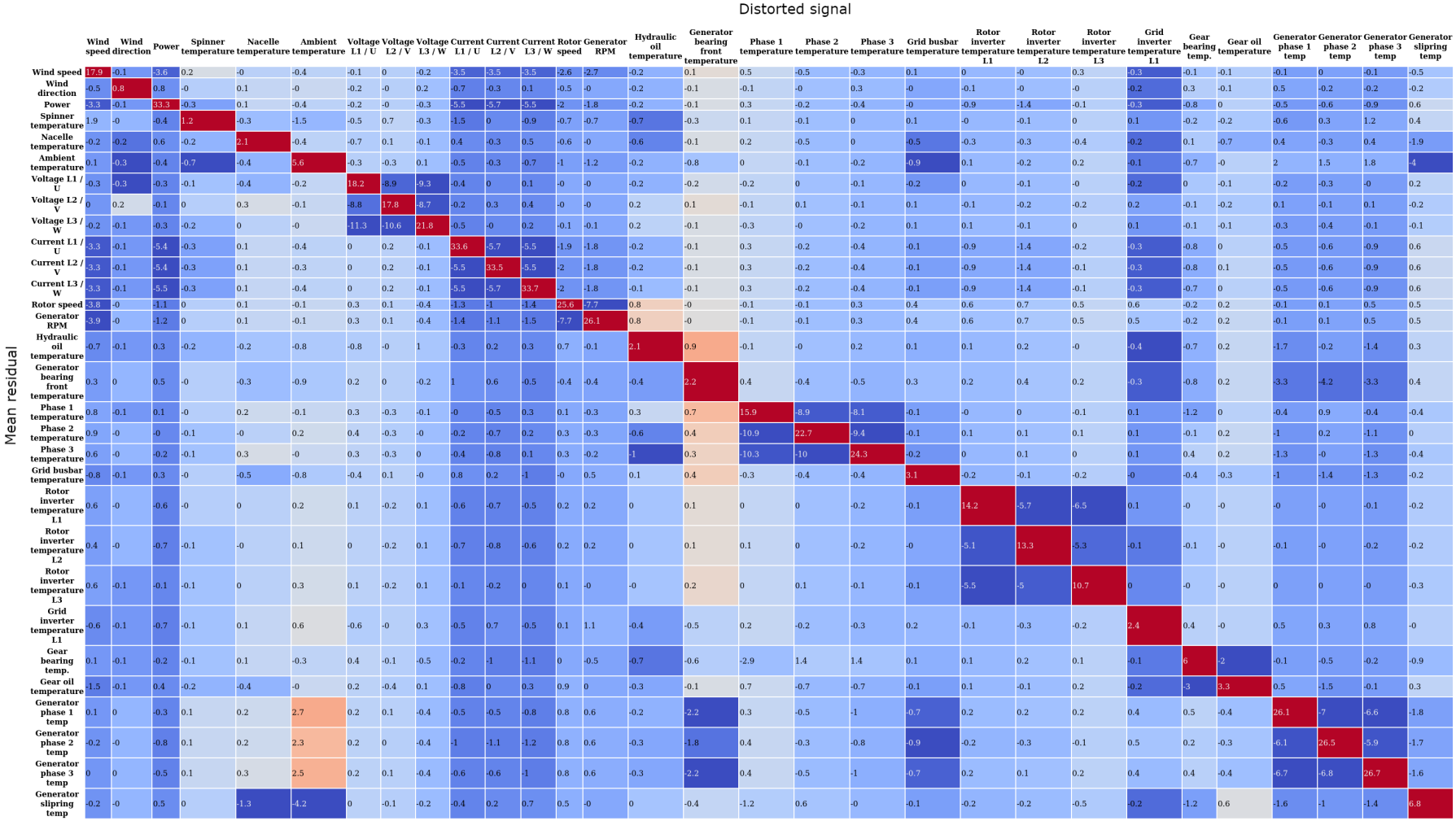


Figure 4.8: The connection map created by adding a distortion to each signal one by one. The columns represent the distorted signal and the rows are the effect on the signals. The colouring is done per column, where the highest value in each column is coloured dark red and the lowest value dark blue. The colours are not comparable between columns. The connection map is created by using 10 autoencoders and the map is the mean from the result from the 10 autoencoders.

4.2 Validation of the condition monitoring system

In this section, the proposed method for condition monitoring of wind turbines presented in Section 3.5 was tested on data from wind turbines with known faults. The condition monitoring system consists of two parts, the *fault detection* part, described in Section 3.5.1, and the *fault diagnosis* part, described in 3.5.2.

4.2.1 Test cases

The condition monitoring system was tested on data from turbines with known faults. Most of the faults were found by an existing condition monitoring system that is based on the work presented in [28], and were confirmed by looking at service logs. The other faults were found only by looking at service logs, since the existing condition monitoring system did not produce a warning for these faults. The existing condition monitoring system uses one small feed-forward neural network per signal it monitors, which allows it to produce warnings when the monitored signals are higher than expected, but does not incorporate the connections between the different signals. In the following, the existing condition monitoring system is called ANN-CMS while the novel condition monitoring system proposed in this thesis is called AE-CMS.
The test cases are presented in Table 4.1. The column *Warning signals* shows the signals that were causing warnings according to the ANN-CMS. If the cell is empty the ANN-CMS did not give warning for the fault. In the column *Cause* the cause for the alarm is specified. This information, and the information in *Component breakdown*, which specifies when the component broke down, both come from service logs. The time period for the tests are specified as 60 days per fault, and is shown in the column *Test time period*. As seen in the table there are eight faults of which five are unique. The faults came from six different turbines where two turbines experienced two different faults at different times. There is also one test case for healthy data, to compare the results from healthy data to results from a faulty turbine.

4.2.2 Method

The proposed method for condition monitoring of wind turbines is described in Section 3.5 and includes *fault detection* and *fault diagnosis*. The fault detection part is represented by one plot per test case that shows the Mahalanobis distance calculated on the residual from the validation data and inference data from one autoencoder. In the Mahalanobis distance plot, the test period for the fault is marked with a blue square. To tell if the fault could be detected, the Mahalanobis distance must be larger for the data within the marked test period than it is for the validation data, otherwise the fault could not be found by setting a threshold on the Mahalanobis distance.
The fault diagnosis is done by comparing the residual for each individual signal and this is done by examining a boxplot showing the mean residual over the test period. The boxplots were created as described in Section 3.5.2.1, with the use of

Wind farm	Wind turbine	Warning signal (ANN-CMS)	Cause	Component breakdown	Test time period
A	3	No fault	No fault		2018-04-01 – 2018-06-01
A	1	Rotor inverter temp L1, L2, L3, Grid inverter temperature L1	Cooling system issue		2018-12-03 – 2019-02-01
A	5	Rotor inverter temp L1, L2, L3, Grid inverter temperature L1	Cooling system failure		2019-01-19 – 2019-03-20
A	5	Generator slipring temperature	Generator slipring hose failure		2018-07-14 – 2018-09-12
A	2	Generator slipring temperature	Slipring brush failure		2019-09-01 – 2019-10-31
A	2	Hydraulic oil temperature	Hydraulic system component failure		2018-09-16 – 2018-11-15
A	3		Gearbox	2019-01-17	2018-11-18 – 2019-01-17
A	4		Gearbox	2018-11-15	2018-09-16 – 2018-11-15
B	1		Yaw encoder failure		2018-03-02 – 2018-05-01

Table 4.1: Table over the test cases. The column *Warning signal (ANN-CMS)* shows the signals that were causing a warning according to an existing condition monitoring system. The column *Cause* shows the cause of the fault according to service logs and the column *Component breakdown* show, if known, when the component broke down and needed to be replaced.

10 autoencoders per turbine. The result of the fault diagnosis is compared to what signals caused warning according to the ANN-CMS. If the same signals seemed to be responsible for the fault, the fault diagnosis proposed here was seen as successful for the specific fault. For each fault, the results are also compared with knowledge about what component the signals were measured on. This was especially important for the two types of faults that the ANN-CMS had not warned for, since in these cases there was no known faulty signal to compare the result to.

4.2.3 Results and discussion per fault type

Here the results from the test cases are presented. The results are displayed as plots and are described and discussed under the subsection for each fault type. Note that both the range of the plotted data and the values on the y-axises differ between plots. In Table 4.2, the largest positive and negative residuals are shown in comparison with the result from the ANN-CMS and the service logs.

4.2.3.1 Healthy data

The result of the AE-CMS on healthy data can be seen in Figure 4.9. The Mahalanobis distance is low, and the amplitudes of the residuals shown in the boxplot are also low. It looks as if the wind turbine might be producing less power than expected, by looking at the low residual for power and current, but since the amplitude overall is so low this is not bad enough to be classified as a fault.

4.2.3.2 Cooling system failures

Two turbines experienced faults in the cooling system connected to the rotor inverter, which resulted in too high temperatures in the rotor inverter and the grid inverter according to the ANN-CMS. The result for Wind farm A: Turbine 1 and Turbine 5 can be seen in Figure 4.10 and 4.11 respectively. The Mahalanobis distance is high for both turbines at the faulty time period, showing that the fault is detected. The boxplots are very similar for the two faults, both showing that the residual for the signals *Rotor inverter temperature L1, L2, L3* and *Grid inverter temperature L1* is large and positive, which means that these temperature are higher than expected. This matches the result from the ANN-CMS. The residual for *Power* and for the currents are negative, which is likely due to the produced power being expected to be higher when the wind turbine is operating at the current temperatures for grid and rotor inverter. The connection between the grid and rotor inverter temperatures and the signals for power and current can also be seen in the connection map, Figure 4.8, which confirms the suspicion that the autoencoder expects more power to be produced at these high temperatures.

4.2.3.3 Generator slipring failures

There were two turbines that experienced problems with the generator slipring, Wind farm A: Turbine 5 and Turbine 2, with results shown in Figure 4.12 and 4.13, respectively. Both result show a clear rise in the Mahalanobis distance during the test period, meaning that the faults can be found. Both also show a large residual for the signal *Generator slipring temp*, which is the same signal as the ANN-CMS warned for. This signal is also the only signal that is measured directly on the generator slipring, and a rise in this signal is expected when there is a fault in the generator slipring.
In both boxplots it is also visible that the residuals for the signals *Generator phase 1-3 temp* are negative, which could mean that they are either lower that they should be, or that the autoencoder has learned that they are usually more similar to the *Generator slipring temperature*. The latter would mean that the autoencoder has found a relationship between these signals and this relationship can in fact be seen in the connection map in Figure 4.8. This relationship can also be explained by these temperatures all being measured on the generator.

4.2.3.4 Hydraulic system issue

Wind farm A: Turbine 2 experienced a problem in the hydraulic oil system and the result from the AE-CMS can be seen in Figure 4.14. As seen in the figure, the Mahalanobis distance is large during the test period, meaning the fault detection system can find the fault. The boxplot shows a large positive residual for the signal *Hydraulic oil temperature*, which match the results from the ANN-CMS. The residual for *Gear bearing temperature* is also large and positive. In the connection map, Figure 4.8, these two signals are shown to vary together. The reason for this can be explained physically since the hydraulic oil system and the gearbox cooling system use the same oil. A fault in a component in the hydraulic system may therefore

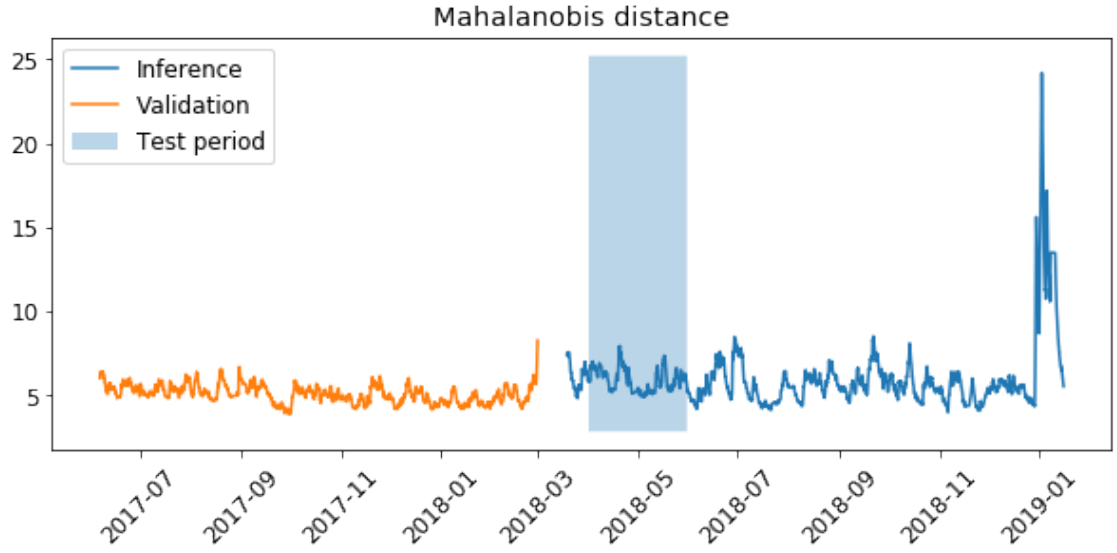
affect both the *Hydraulic oil temperature* and the *Gear bearing temperature*.

4.2.3.5 Gearbox failure

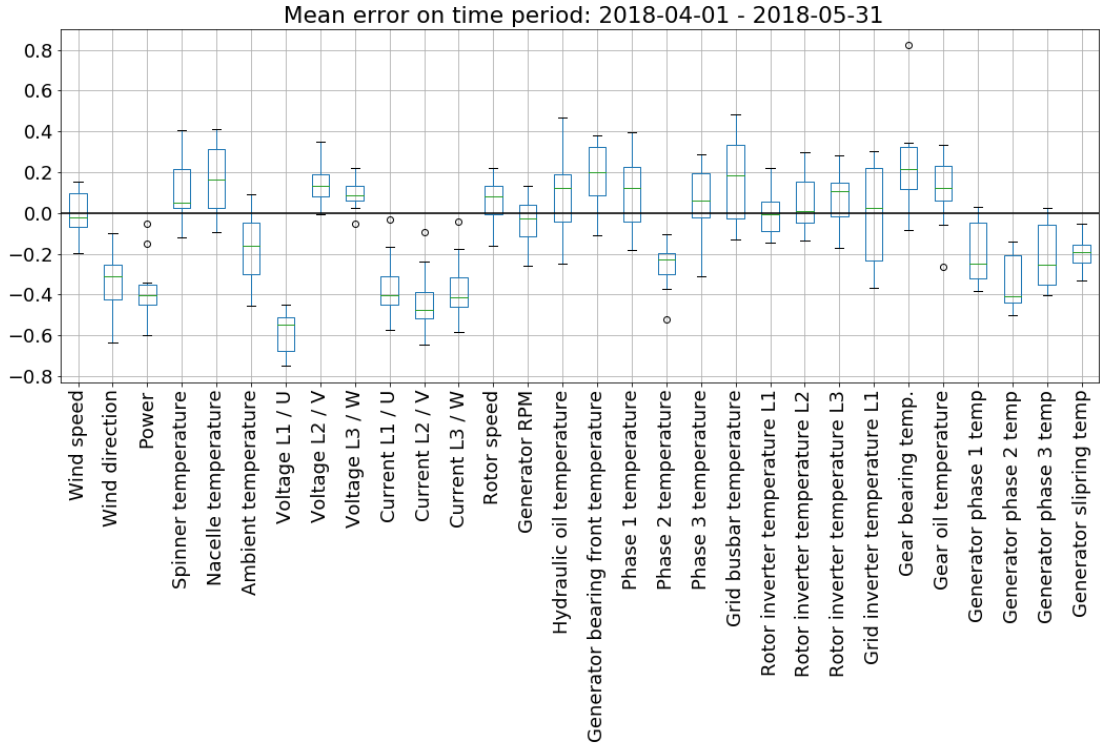
Two wind turbines in the test set experienced gearbox failures according to service logs. These faults were not found by the ANN-CMS, so there is no ground truth telling what signals were high or abnormal. Figure 4.15 shows the result for Wind farm A: Wind turbine 3. The Mahalanobis distance shows a clear peak during the test period and hence the fault can be detected. The boxplot shows large positive residuals for *Gear oil temperature* and *Gear bearing temperature*. In Figure 4.16, the result for Wind farm A: Wind turbine 4 is shown. The Mahalanobis distance plot shows one large, but thin, spike during the test period, but for the main part of the test period the Mahalanobis distance is relatively low. This means that the fault is difficult to detect by using the Mahalanobis distance. The boxplot shows a large positive residual for *Gear bearing temperature* and a large negative residual for *Gear oil temperature*.
As seen by comparing the results from the two wind turbines, both have a large positive residuals for *Gear bearing temperature*, while the residual for *Gear oil temperature* is positive for the first turbine and negative for the second. These temperatures are both measured in the gearbox, why a fault in the gearbox is likely to affect these signals. The difference in the sign for *Gear oil temperature* could possibly be due to different components in the gearbox being responsible for the fault. The exact faults were not specified in the service logs. Both boxplots also show that the power and the currents are having negative residuals, which is probably due to the wind turbine producing more power at this high *Gear bearing temperature* when it is operating in a healthy condition.

4.2.3.6 Yaw encoder failure

The yaw encoder turns the wind turbine in the direction of the wind, to maximize power production and ensure optimal operation conditions for the turbine. If it is not operating correctly the wind turbine cannot turn in the right direction, which Wind farm B: Wind turbine 1 experienced. The ANN-CMS did not produce a warning for this issue. The result from the AE-CMS can be seen in Figure 4.17. The Mahalanobis distance is high for the test period, while the amplitude of the residuals shown in the boxplot are as low as they were for the healthy wind turbine shown in Figure 4.9. From the boxplot it is hard to by eye diagnose what fault happened in the wind turbine. There may be a pattern from the residuals that could be found and used to classify Yaw encoder failures, for example with the use of a neural networks classifier.

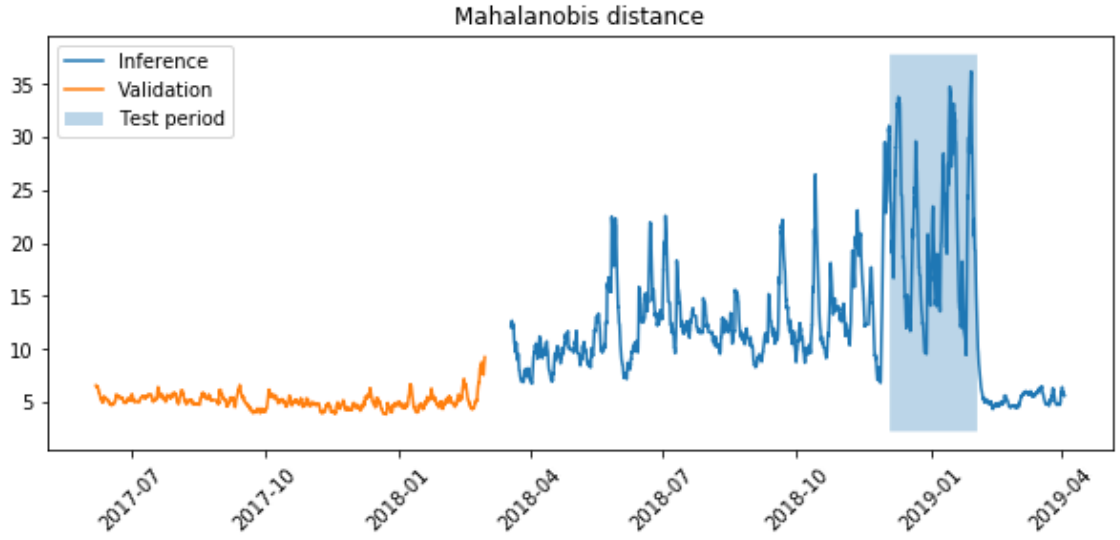


(a) Mahalanobis distance on validation and inference residual. Test period marked.

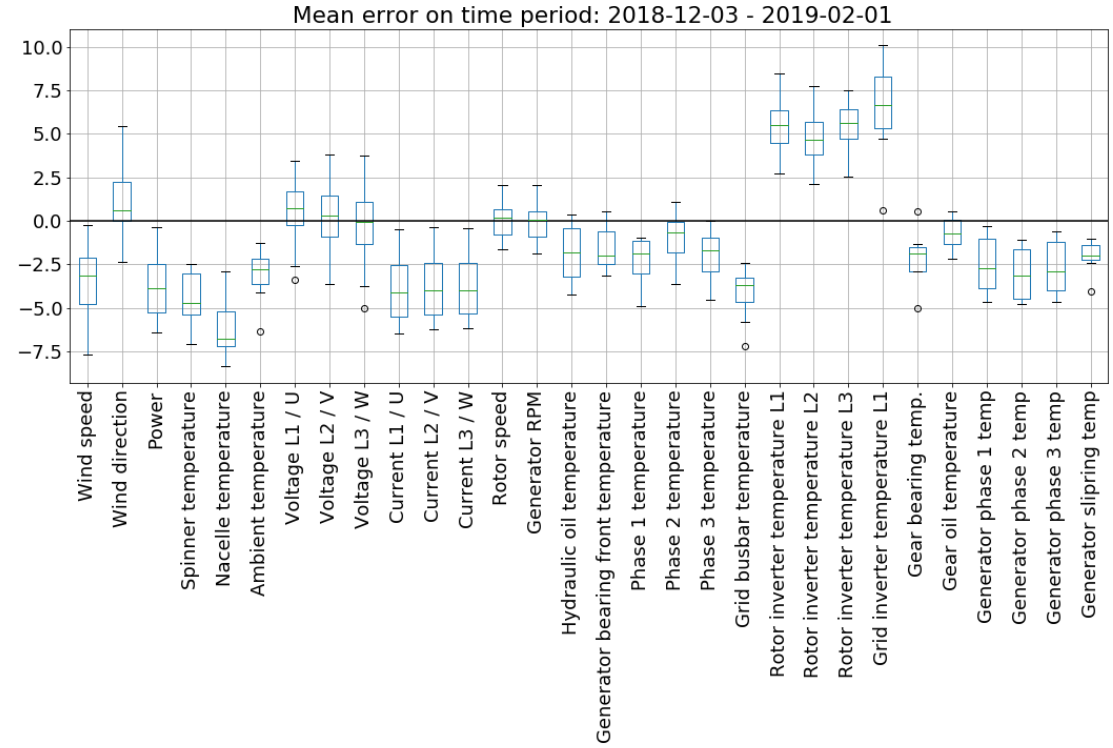


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.9: Data from **Wind farm A: Turbine 3**, when it is operating under **healthy conditions**. In (a), the test period for this test is marked. The gearbox failure described in Figure 4.15 can also be seen as the rise in the Mahalanobis distance at 2019-01.

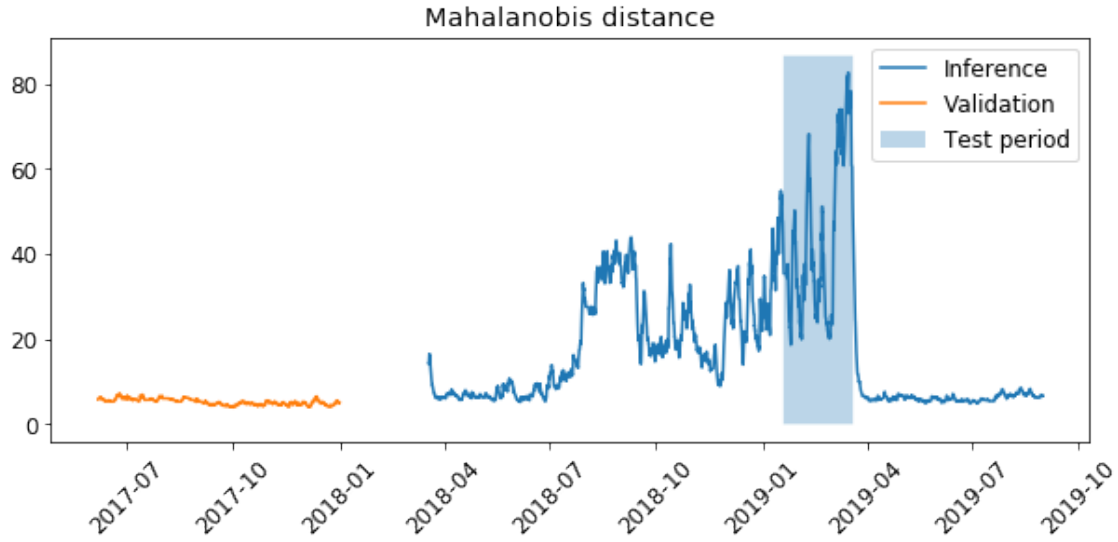


(a) Mahalanobis distance on validation and inference residual. Test period marked.

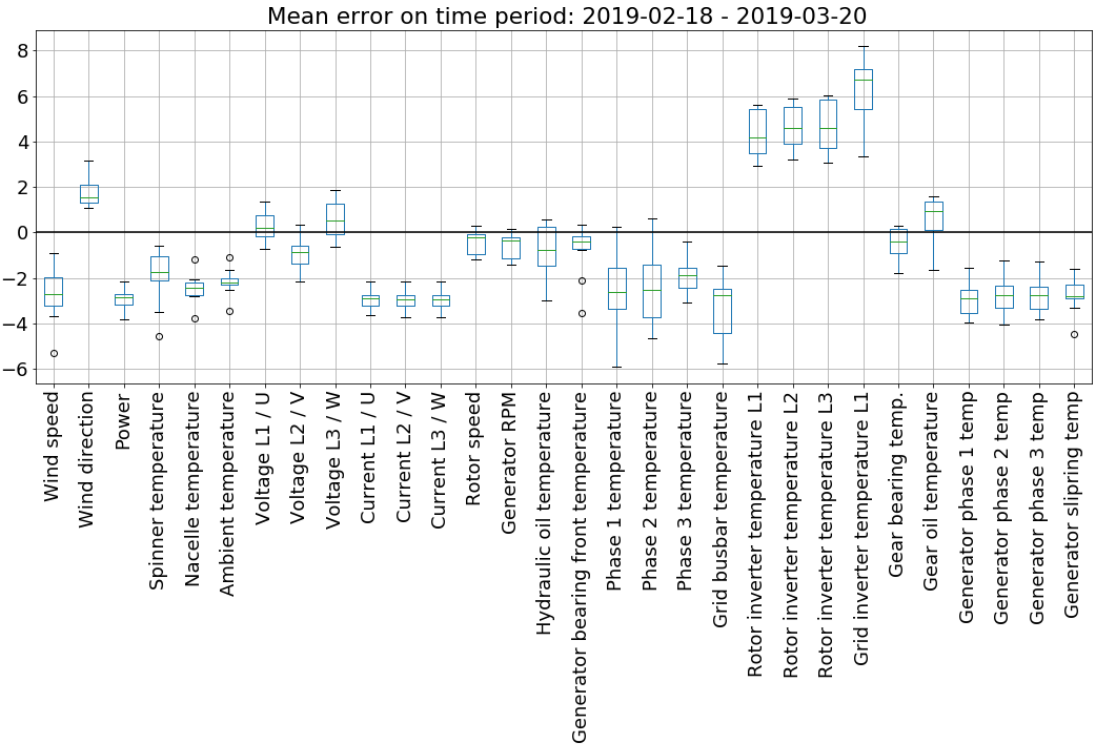


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.10: Data from **Wind farm A: Turbine 1**. The turbine was experiencing **cooling system issues** according to service logs. According to the existing anomaly detection method the signals *Rotor inverter temperature L1, L2, L3* and *Grid inverter temperature L1* were high. In (a), the test period for this test is marked.

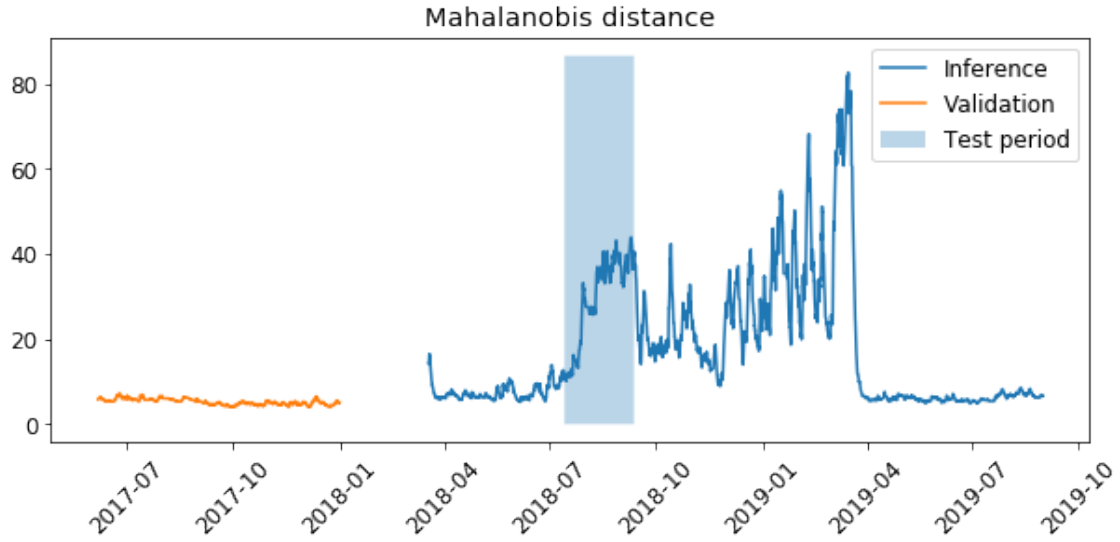


(a) Mahalanobis distance on validation and inference residual. Test period marked.

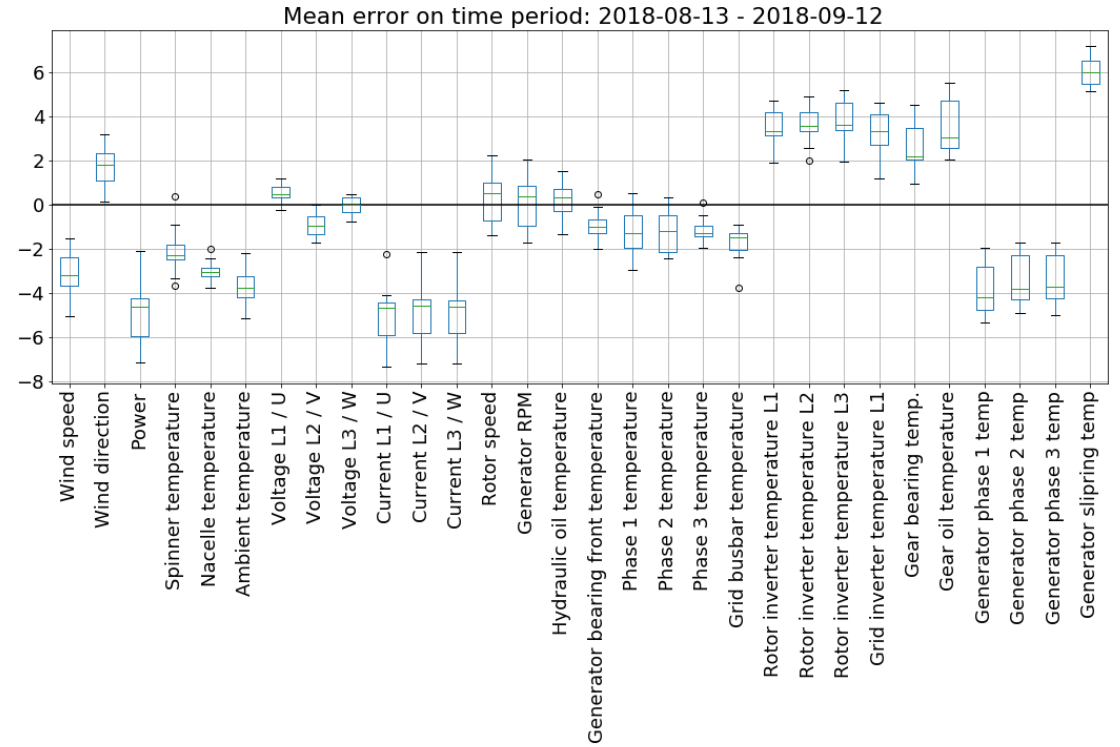


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.11: Data from **Wind farm A: Turbine 5**. The turbine was experiencing **cooling system issues** according to service logs. According to the existing anomaly detection method the signals *Rotor inverter temperature L1, L2, L3* and *Grid inverter temperature L1* were high. In (a), the test period for this test is marked. The fault described in Figure 4.12 can also be seen as the rise in the Mahalanobis distance at 2018-07 – 2018-09.

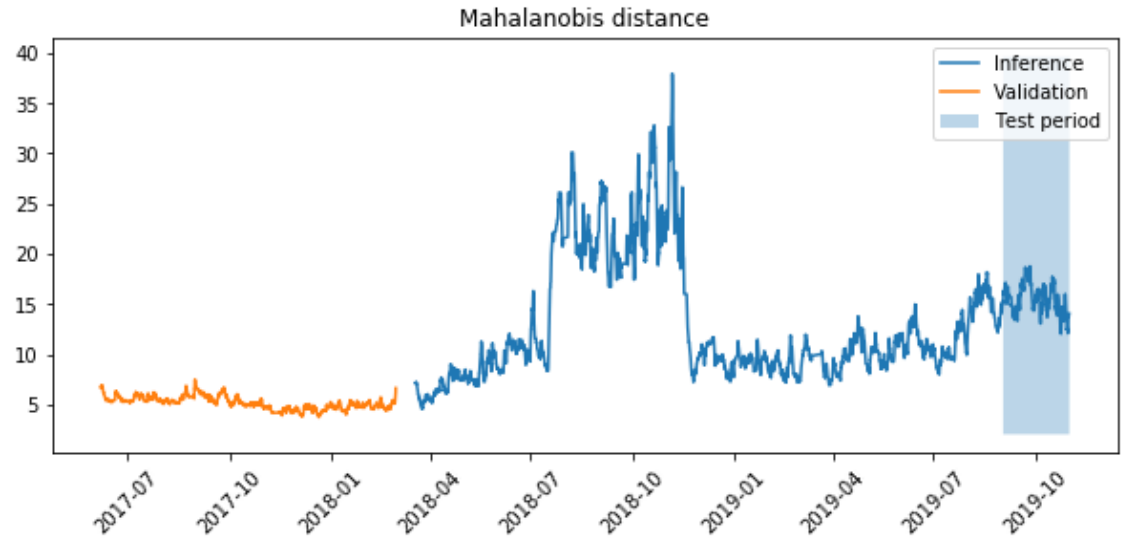


(a) Mahalanobis distance on validation and inference residual. Test period marked.

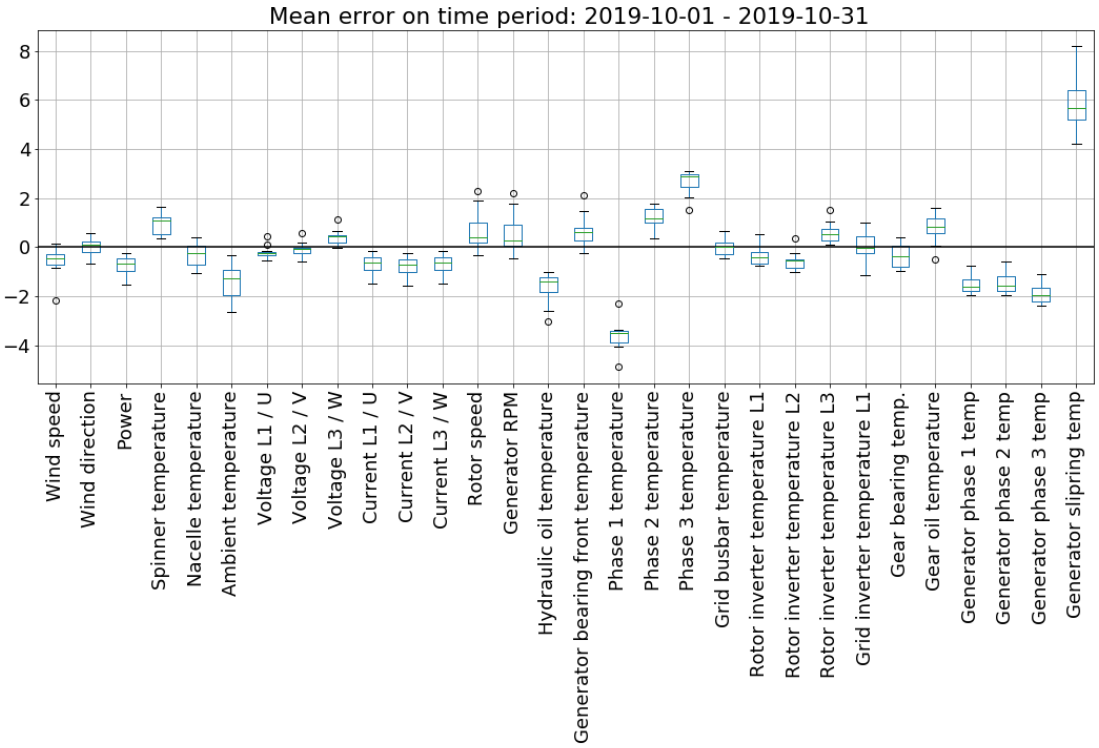


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.12: Data from **Wind farm A: Turbine 5**. The turbine was experiencing a **failure in the generator slipring hose** according to service logs. According to the existing anomaly detection method the signal *Generator slipring temperature* was high. In (a), the test period for this test is marked. The fault described in Figure 4.11 can also be seen as the rise in the Mahalanobis distance at 2019-01 – 2019-04.

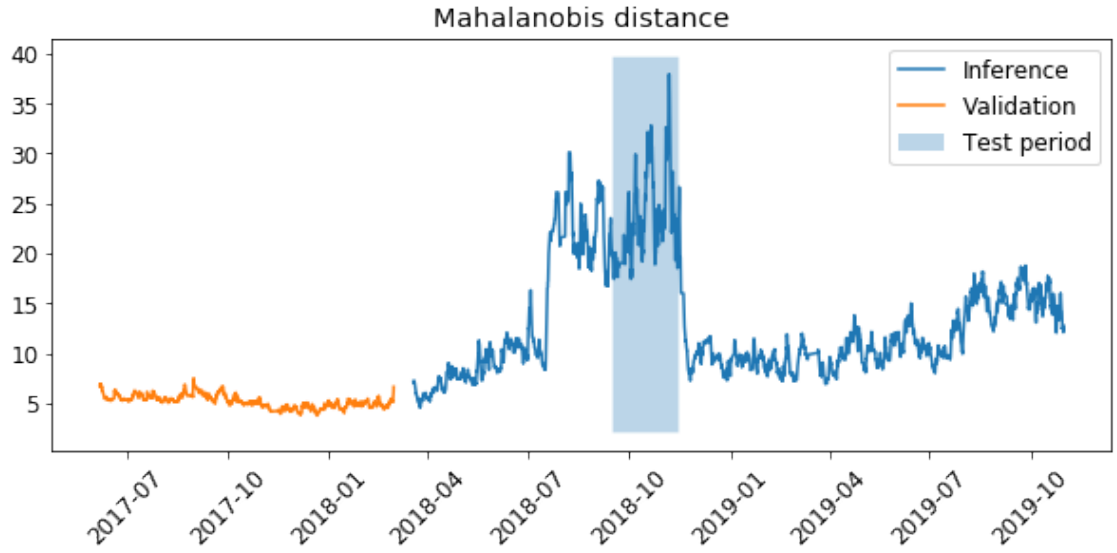


(a) Mahalanobis distance on validation and inference residual. Test period marked.

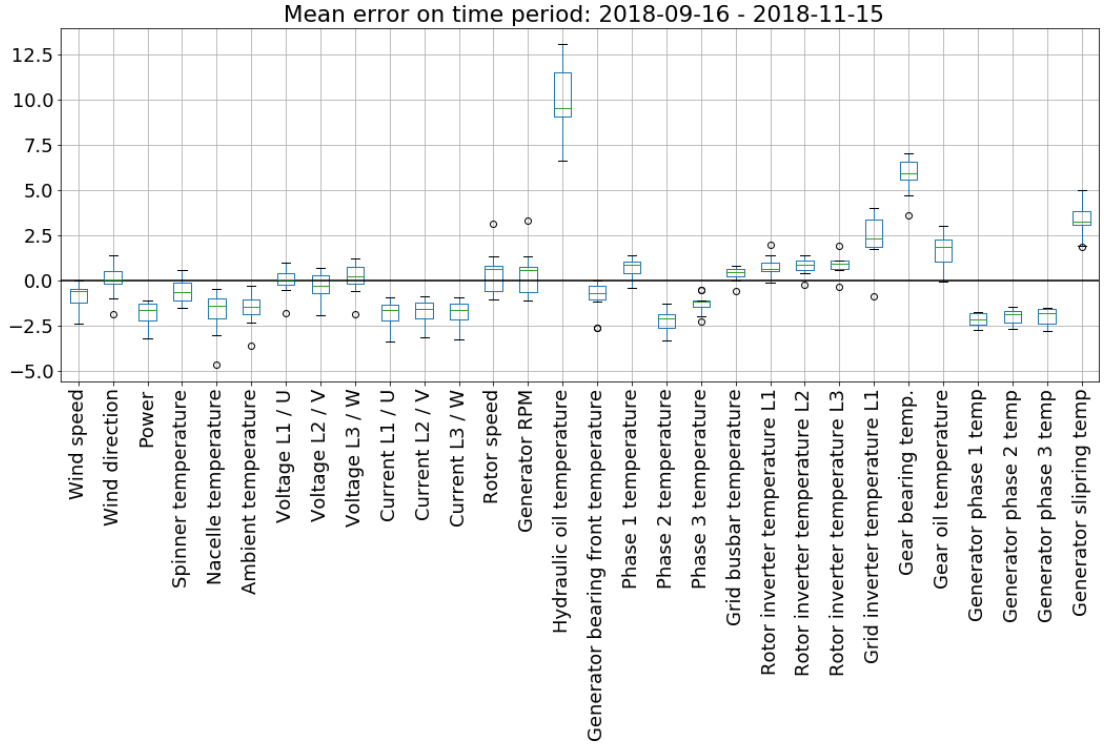


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.13: Data from **Wind farm A: Turbine 2**. The turbine was experiencing a **failure in the generator slipring brush** according to service logs. According to the existing anomaly detection method the signal *Generator slipring temperature* was high. In (a), the test period for this test is marked. The fault described in Figure 4.14 can also be seen as the rise in the Mahalanobis distance at 2018-07 – 2018-11.

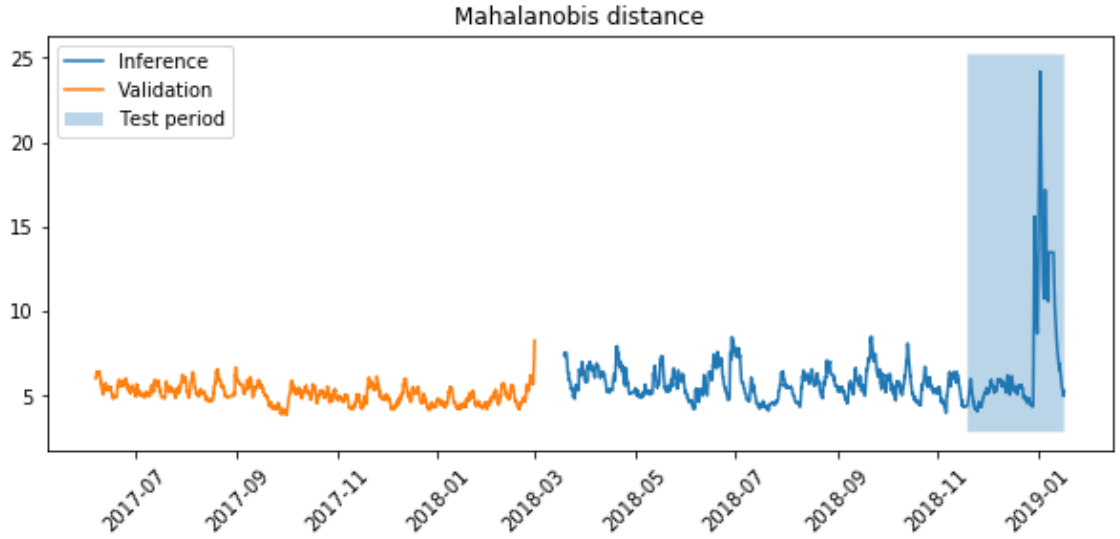


(a) Mahalanobis distance on validation and inference residual. Test period marked.

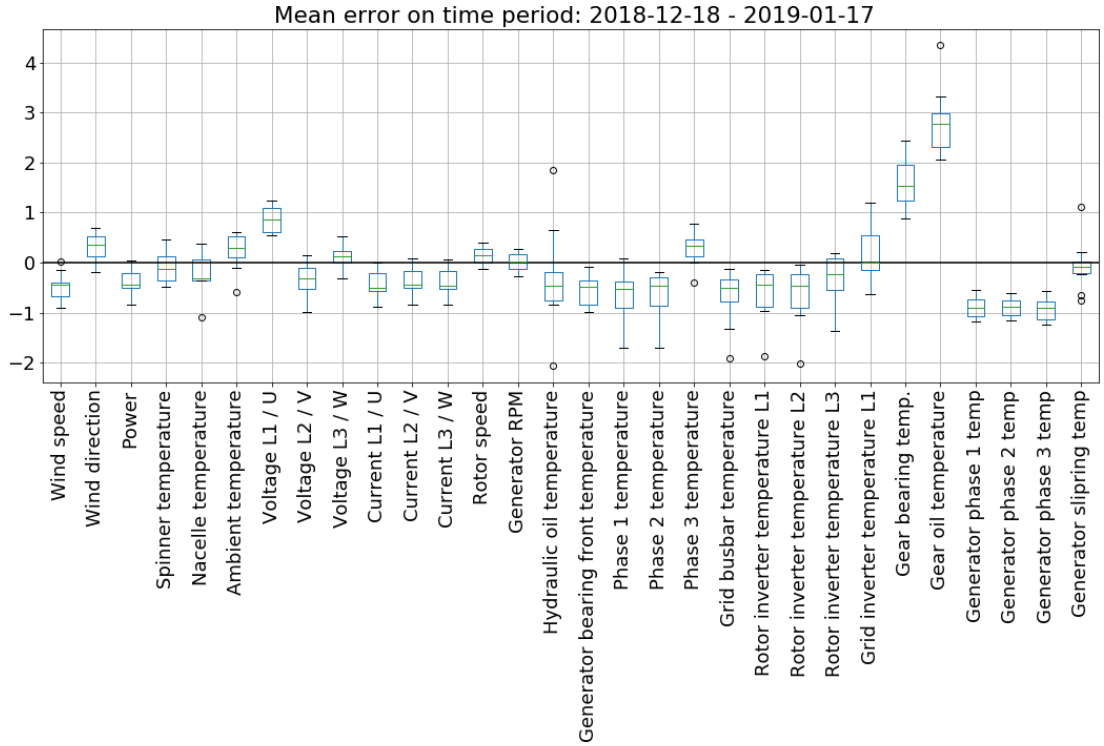


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.14: Data from **Wind farm A: Turbine 2**. A **component in the hydraulic system failure** according to service logs. According to the existing anomaly detection method the signal *Hydraulic oil temperature* was high. In (a), the test period for this test is marked. The fault described in Figure 4.13 can also be seen as the rise in the Mahalanobis distance at 2019-08 – 2019-11.

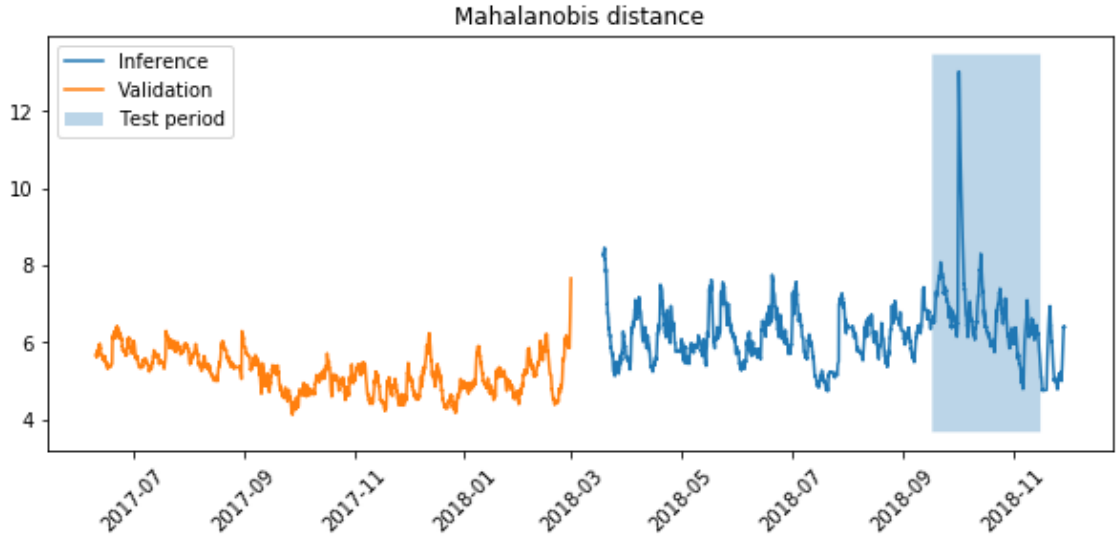


(a) Mahalanobis distance on validation and inference residual. Test period marked.

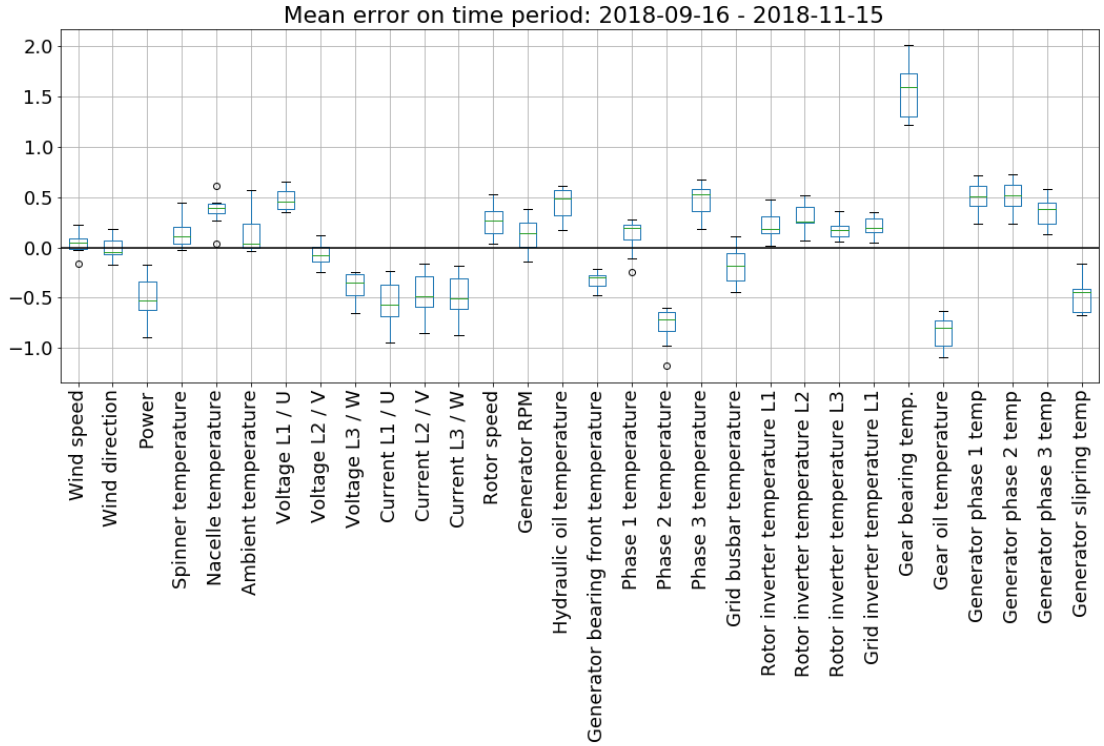


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.15: Data from **Wind farm A: Turbine 3**. **Gearbox failure** according to service logs. In (a), the test period for this test is marked.

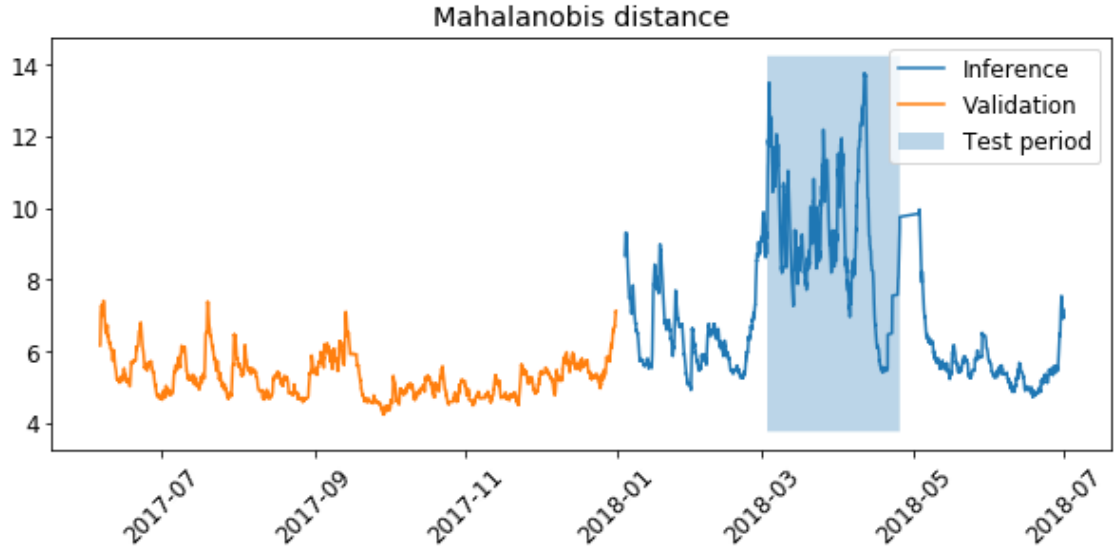


(a) Mahalanobis distance on validation and inference residual. Test period marked.

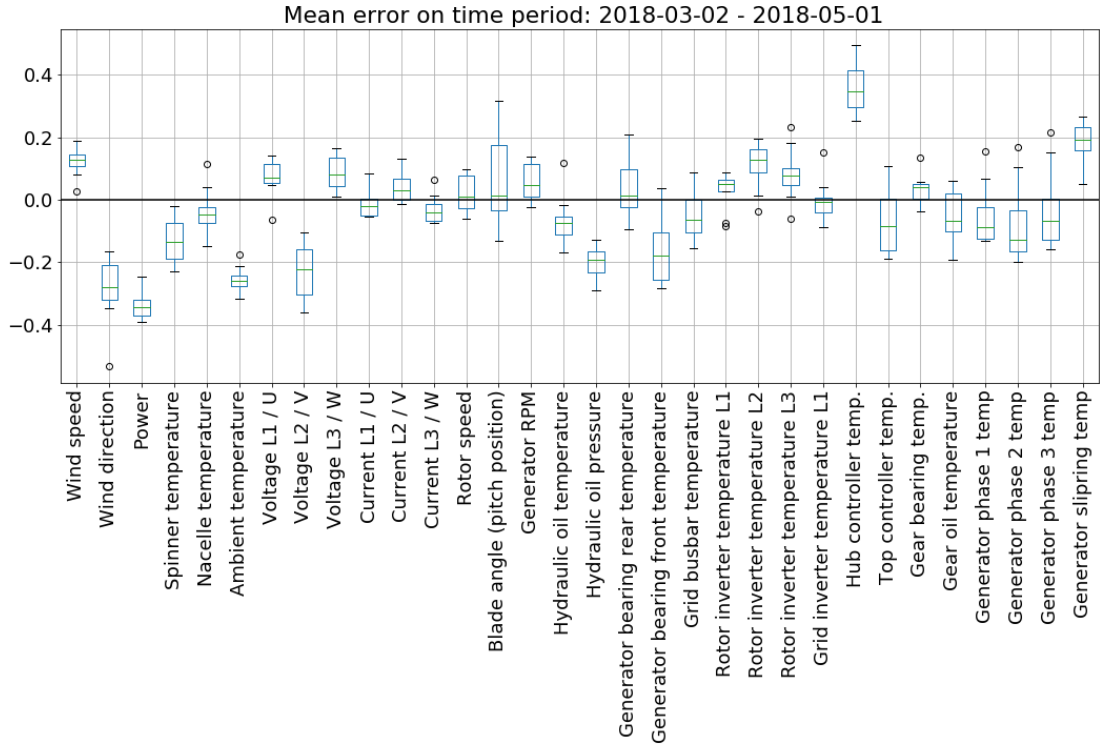


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.16: Data from **Wind farm A: Turbine 4**. **Gearbox failure** according to service logs. In (a), the test period for this test is marked.



(a) Mahalanobis distance on validation and inference residual. Test period marked.



(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure 4.17: Data from **Wind farm B: Turbine 1**. **Yaw encoder failure** according to service logs. In (a), the test period for this test is marked.

Wind farm	Wind turbine	Warning signal (ANN-CMS)	Cause	Result figure	Large positive residual (median val.) (AE-CMS)	Large negative residual (median val.) (AE-CMS)
A	3	No fault	No fault	4.9	-	
A	1	Rotor inverter temp L1, L2, L3, Grid inverter temperature L1	Cooling system issue	4.10	Rotor inverter temp L1 (5.2), L2 (4.8), L3 (5.3), Grid inverter temperature L1 (6.2)	Nacelle temperature (-6.6)
A	5	Rotor inverter temp L1 (10.2), L2 (11), L3, Grid inverter temperature L1	Cooling system failure	4.11	Rotor inverter temp L1 (4.2), L2 (4.6), L3 (4.6), Grid inverter temperature L1 (6.7)	
A	5	Generator slipring temperature	Generator slipring hose failure	4.12	Generator slipring temperature (6.0)	
A	2	Generator slipring temperature	Slipring brush failure	4.13	Generator slipring temp (5.7)	Phase 1 temperature (-3.5)
A	2	Hydraulic oil temperature	Hydraulic system component failure	4.14	Hydraulic oil temperature (9.5), Gear bearing temperature (6.1)	
A	3		Gearbox	4.15	Gear oil temperature (2.8), Gear bearing temperature (1.5)	
A	4		Gearbox	4.16	Gear bearing temperature (1.6)	Gear oil temperature (-0.8), Phase 2 temperature (-0.7)
B	1		Yaw encoder failure	4.17	Hub controller temp (0.3)	Power (-0.4)

Table 4.2: Table over the test cases showing faults and warnings according to the ANN-CMS and service logs and the corresponding results gained with the AE-CMS. In the two columns *Large positive / negative residual (median val.) (AE-CMS)* the largest positive and negative residuals are presented for each fault, with the median values displayed within parenthesis.

4.2.4 Conclusion from the validation on real world faults

Out of eight test cases it was clear that seven faults could be detected by monitoring the rise in the Mahalanobis distance, which is two more faults than what the ANN-CMS had detected. The fault that could not be detected by the AE-CMS (nor the ANN-CMS) was the gearbox failure for Wind farm A: Wind turbine 4, since the Mahalanobis distance was low for almost the whole test period, as seen in Figure 4.16. This fault might be detectable if the separate residuals were monitored instead, since it was clear in the boxplot for the fault that the separate residuals for the gear temperatures were large.
It was possible to diagnose the fault for seven of the eight test cases by looking at the boxplots. The boxplots showed large residuals for all the signals the ANN-CMS had produced warnings for and could also be used to diagnose the two gearbox failures, since the residual for the signals measured in the gearbox were large compared to the other signals. The fault that the AE-CMS failed to diagnose was the Yaw encoder issue, Figure 4.17, as the amplitude of the separate residuals in the boxplot were as low as for a healthy wind turbine and showed no clear pattern. Since there was no signal that directly measured the yaw encoder, it might be hard to see the fault on the residual for a specific signal. But, since the Mahalanobis distance was large for the test period, there must be some fault in the separate residuals that should be possible to find, but maybe not by just analysing the boxplot by eye. An algorithm could potentially be trained to find patterns in the residual and thereby help in diagnosing this and other types of faults. This is suggested as a continuation of the work presented in this thesis.
The *connection map*, Section 4.1.2, helped when interpreting the result seen in the boxplots. For example it helped explaining why the residual for the power and currents was low for the cooling issues shown in Figure 4.10 and 4.11. It also helped in explaining the low residual for *Generator phase 1-3 temperatures* when there was a fault in the generator slipring, as discussed in Section 4.2.3.3.

5
Closure

5.1 Conclusion

The aim of this thesis was to (i) design a condition monitoring system based on an autoencoder that can detect and diagnose developing faults, (ii) examine what internal connections the autoencoder has found in order to understand how the residual between the input and output of the autoencoder behaves.
The first aim was met by the proposed condition monitoring system, as was shown in the validation study of eight known faults found in five different components: gearbox issue, cooling issue, hydraulic oil issue, yaw encoder issue and generator slipring issue. The autoencoder was used to produce a residual, taken as the error between the input to the autoencoder and its reconstructed signal. For fault detection, the Mahalanobis distance was used on the residual and it was shown that when the wind turbine experienced a fault, the Mahalanobis distance was large for seven of the eight faults. For fault diagnosis, the residual for each signal was standardized with respect to how large the residual is for healthy data. The standardized residual was analyzed to examine which signals are mostly affected by the fault. The results showed that seven of eight faults could be diagnosed by examining the largest residuals, while the eighth fault, the yaw encoder issue, did not show a pattern that could be interpreted easily by eye.
The second aim was met by the usage of simulated faults, *distortions*, added to healthy data. By adding a distortion to just one signal and then inserting the data containing the simulated fault into the autoencoder, a residual was produced. From examining the residual it was clear that the residual for multiple signals was affected, not only the residual for the distorted signal. The affected signals were interpreted as being connected to the distorted signal, according to what the autoencoder had learned. The relationships that were found with this method could be explained by physical relationships, like two temperatures being measured on the same component, but also more advanced relationships, as temperatures being connected to how much power the autoencoder expected the wind turbine to produce. The study of simulated faults both helped in interpreting the residual and in understanding what complex relationships the autoencoder had found between the signals. The connection map also provided help when diagnosing the real faults since it helped explaining why the residual for some signals was affected by the known fault in the wind turbine.

5.2 Future work

Further investigations to improve the condition monitoring system could be done in relation to both the fault detection step and the fault diagnosis step. To be able to find more faults, it would be interesting to examine the possibility of monitoring each residual by itself, instead of summarizing them with the Mahalanobis distance. In this thesis it was shown that the Mahalanobis distance was not large for a known fault in the gearbox, but the residual for a signal measured in the gearbox was large. This means that a fault that is missed by the Mahalanobis distance potentially could be found by individual signal monitoring.
When it comes to fault diagnosis, it was seen that all faults but the yaw encoder issue was possible to diagnose. The yaw encoder issue had large Mahalanobis distance, while the individual residuals were low and showed no clear pattern. Since the Mahalanobis distance was large, one can assume that the individual residuals does have information about the fault. To examine if there indeed is a pattern in the residual, it would be interesting to test this on more turbines with a yaw encoder issue. By examining multiple faults and fault types, it would be possible to train a classifier on the results from the fault diagnosis. This classifier could potentially use the residual for each signal as a *fingerprint* and classify what fault such a fingerprint means.
The *connection map*, Section 4.1.2, that was created by distorting one signal at a time and recording the affect of the distortion on the residual for all signals in a matrix, could be investigated further. In this thesis only one type of distortion has been investigated, but different connections between the signals might be found by examining different distortions, changing the amplitude of the distortion or distorting multiple signals at once. Doing this would also help in understanding how sensitive the results in the connection map are to other types of distortions. This would give more knowledge about what the autoencoder has learned and be of help when analyzing the results from real faults.

Bibliography

[1] H. Ritchie and M. Roser, "Renewable energy," *Our World in Data*, 2019. `https://ourworldindata.org/renewable-energy`.

[2] I. (2019), "Renewable capacity statistics 2019," tech. rep., International Renewable Energy Agency (IRENA), 2019.

[3] L. Bergström, L. Kautsky, T. Malm, R. Rosenberg, M. Wahlberg, N. Å. Capetillo, and D. Wilhelmsson, "Effects of offshore wind farms on marine wildlife—a generalized impact assessment," *Environmental Research Letters*, vol. 9, p. 034012, mar 2014.

[4] P. Qian, X. Ma, and P. Cross, "Integrated data-driven model-based approach to condition monitoring of the wind turbine gearbox," *IET Renewable Power Generation*, vol. 11, no. 9, pp. 1177–1185, 2017.

[5] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic, "Machine learning methods for wind turbine condition monitoring: A review," *Renewable energy*, vol. 133, pp. 620–635, 2019.

[6] H. Long, L. Wang, Z. Zhang, Z. Song, and J. Xu, "Data-driven wind turbine power generation performance monitoring," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6627–6635, 2015.

[7] M. Schlechtingen, I. F. Santos, and S. Achiche, "Wind turbine condition monitoring based on scada data using normal behavior models. part 1: System description," *Applied Soft Computing*, vol. 13, no. 1, pp. 259–270, 2013.

[8] M. Schlechtingen and I. F. Santos, "Wind turbine condition monitoring based on scada data using normal behavior models. part 2: Application examples," *Applied Soft Computing*, vol. 14, pp. 447–460, 2014.

[9] M. Roy, S. K. Bose, B. Kar, P. K. Gopalakrishnan, and A. Basu, "A stacked autoencoder neural network based automated feature extraction method for anomaly detection in on-line condition monitoring," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1501–1507, IEEE, 2018.

[10] G. Michau, Y. Hu, T. Palmé, and O. Fink, "Feature learning for fault detection in high-dimensional condition-monitoring signals," *arXiv preprint arXiv:1810.05550*, 2018.

[11] L. Wang, Z. Zhang, J. Xu, and R. Liu, "Wind turbine blade breakage monitoring with deep autoencoders," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2824–2833, 2016.

[12] G. Jiang, P. Xie, H. He, and J. Yan, "Wind turbine fault detection using a denoising autoencoder with temporal information," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 89–100, 2017.

[13] N. Renström, "Condition monitoring systems for wind turbines – based on deep autoencoders," Master's thesis, Chalmers University of Technology, May 2019.
[14] H.-H. Yang, M.-L. Huang, and S.-W. Yang, "Integrating auto-associative neural networks with hotelling t2 control charts for wind turbine fault detection," *Energies*, vol. 8, no. 10, pp. 12100–12115, 2015.
[15] P. Tavner, J. Xiang, and F. Spinato, "Reliability analysis for wind turbines," *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 10, no. 1, pp. 1–18, 2007.
[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press.
[17] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.
[18] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
[19] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.
[20] A. L. Agnan Kessy and K. Strimmer, "Optimalwhitening and decorrelation," *The American Statistician*, vol. 72, pp. 309–314, Dec 2018.
[21] P. C. Mahalanobis, "On the generalised distance in statistics," *Proceedings National Institute of Science, India*, vol. 2, pp. 49–55, Apr 1936.
[22] D. C. Montgomery, *Introduction to statistical quality control*. John Wiley Sons, 6 ed., 2009. Page 419-420.
[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.
[24] "PyTorch documentation." `https://pytorch.org/docs/stable/index.html`. Accessed: 2010-12-18.
[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec 2014.
[26] "Pandas boxplot." `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.boxplot.html`. Accessed: 2010-12-18.
[27] "Numpy linspace." `https://docs.scipy.org/doc/numpy/reference/generated/numpy.linspace.html`. Accessed: 2010-12-18.
[28] P. Bangalore, *Load and risk based maintenance management of wind turbines*. PhD thesis, Chalmers University of Technology, Sept 2016.

A

Appendix 1

A.1 Comparing data transformers

Data transformation is used to ensure that the data is in the wanted format, which can differ between applications. In this thesis, two types of data transformation have been examined: Standardizing data and ZCA whitening. Standardizing data is one of the most common ways of transforming data. It removes bias from the data and ensures all features have variance one. ZCA whitening takes this one step further, by also decoupling the signals from each other, i.e removing the correlation between the signals. The idea is that removing correlation between signals might help the autoencoder separate the signals from each other and thereby facilitate training.
In the main thesis, the method of Standardizing data was used in the preprocessing step. The reason for using this method instead of ZCA whitening is due to ZCA not being as suitable for fault diagnosis, as is shown in by the results of this chapter.
Multiple autoencoders were trained on ZCA whitened data from different turbines and was tested on simulated faults as well as a subset of the real world faults examined in section 4.2. Note that the data was transformed in the preprocessing step, meaning that the autoencoder needed to be trained on either the Standardized data or the ZCA whitened data. So when comparing results between the result from the Standardized data and the ZCA whitened data, the autoencoder(s) that produced the results are different autoencoders (same architecture but different values on the trained wights), trained with data from the specified transformer.

A.1.1 Connection map for ZCA whitened data

In section 4.1 simulated data was used to examine how the autoencoder reacts faults similar to sensor faults, meaning that one signal is faulty while the rest is healthy. This was done to examine what the autoencoder had learned and to analyze how the residual behaves when one signal is distorted. By creating a connection map, explained and shown in section 4.1.2, it was possible to analyze both which connections the autoencoder had found and how a fault in one signal affects the other signals. A connection map for ZCA whitened data was created in the same way, with the exception that the transformer in the preprocessing step was different. The result is shown in figure A.1.
The connection map for the standardized data, seen in figure 4.8, showed a clear diagonal, meaning that distorting one signal resulted in a large residual for that signal. This is wanted behaviour, otherwise it is difficult to find which signal is

experiencing the fault. In figure A.1 the connection map for the ZCA transformed data is shown. The diagonal is interrupted in places, for example for *Phase 2 temperature*. This means that if *Phase 2 temperature* is distorted with a positive distortion the result is a negative residual for the same signal, while the largest positive residual is shown for *Gear oil temperature*. This means that a fault in one signal can be shown in the residual for another signal. Another example is that when *Power* is distorted *Ambient temperature* gives the largest positive residual. From these results, it seems as if the ZCA whitening of data is not a good option when trying to diagnose which signals are most affected by a fault.
The simulated faults are different from how most normal faults behave, why ZCA whitening is also examined on real faults in the following section.

A.1.2 Test on real faults using ZCA whitened data

In this section, three of the eight test cases presented in section 4.2 are analyzed when the data is ZCA whitened instead of standardized. The test cases are presented in table A.1 and the result and discussion is presented under each subsection. The results are compared to the result from using standardized data and, when applicable, to the existing condition monitoring system based on the work presented in [28], called the ANN-CMS in the following.

Wind farm	**Wind turbine**	**Warning signal (ANN-CMS)**	**Cause**	**Component breakdown**	**Test time period**
A	1	Rotor inverter temp L1, L2, L3, Grid inverter temperature L1	Cooling system issue		2018-12-03 – 2019-02-01
A	5	Generator slipring temperature	Generator slipring hose failure		2018-07-14 – 2018-09-12
A	4		Gearbox	2018-11-15	2018-09-16 – 2018-11-15

Table A.1: Table over the test cases for ZCA whitening. The column *Warning signal (ANN-CMS)* shows the signals that were causing a warning according to an existing condition monitoring system. The column *Cause* shows the cause of the fault according to service logs.

A.1.2.1 Cooling system issue

In figure A.2 the result for using ZCA transformed data from Wind farm A: Turbine 1 is seen. It shows high Mahalanobis distance for the fault period and the time before, and it has a large positive residual for the signals *Rotor inverter temperature L1-L3* and *Grid inverter temperature L1*, which are the same signals as the ANN-CMS warns for. This shows that this type of fault can still be found and diagnosed using ZCA whitened data. When comparing this result to the same fault analyzed with standardized data, shown in figure 4.10, it is clear is that there seem to be less effect on the other residuals when using ZCA whitened data on this fault. When standardizing the data, the result in the boxplot shows a large residual for the rotor- and grid inverter signals, as well as an affect on e.g. the power and the currents. When whitening the data, this relationships does not seem to be learned,

which means that the autoencoder based on the ZCA whitened data is missing an important connection.

A.1.2.2 Generator slipring issue

In figure A.3 the result when using ZCA whitened data from Wind farm A: Turbine 5 is shown. The fault for this turbine was an issue with the generator slipring hose which affected the *Generator slipring temperature* according to the ANN-CMS. The result for using ZCA whitened data shows a high Mahalanobis distance for the signal. The boxplot shows a large positive residual for the same signal as the ANN-CMS was warning for, the *Generator slipring temperature*. For many of the other signals, the size of the boxplot is larger than the median value for the signal, which means that the value should not be trusted since the result between the autoencoders vary a lot. Compared with the result for standardized data on the same fault, seen in figure 4.12, the result for ZCA whitened data is less trustworthy due to the large boxes.

A.1.2.3 Gearbox failure

Figure A.4 shows the result for using ZCA whitened data from Wind farm A: Turbine 4 during a time period when the wind turbine experienced gearbox issues. The Mahalanobis distance is low, except for one large spike, which is a similar result as for standardized data, figure 4.16. This means that the fault would not be detected by using the Mahalanobis distance. Multiple boxes in the boxplot are larger than their median value and are crossing zero. The boxplot does not seem to give any indication that there is a signal that is affected by a fault.

A.1.3 Conclusion

When using simulated faults to examine the usage of ZCA whitened data it was shown that the signal that was distorted did not always produce the largest residual. This means that it is hard to draw conclusion on which signal is experiencing a fault by looking at the residuals.
The examination of using ZCA whitening on three test cases from known faults showed that the Mahalanobis distance is similar to the result from using standardized data. It is possible that ZCA whitening can be used to train an autoencoder that is used only for anomaly detection. When analyzing the boxplots in order to diagnose the fault the ZCA whitening gave less information about the fault than what was given when using standardized data. Two of the faults could potentially still be diagnosed, but the test with a failing gearbox could not be diagnosed when using ZCA whitened data.
From the results and discussion in this chapter it has been shown that ZCA whitened data is less fit for fault diagnosis than standardized data.

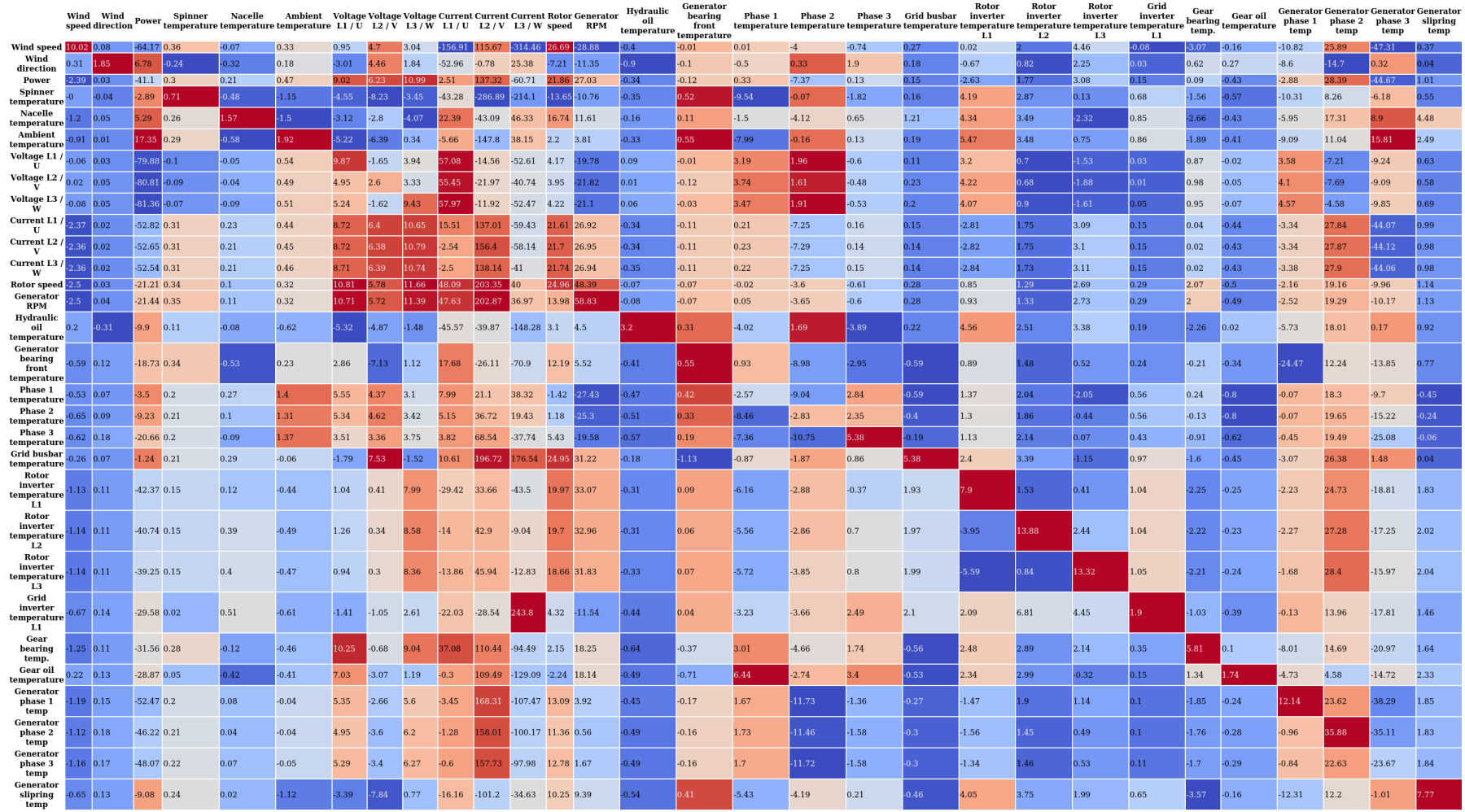
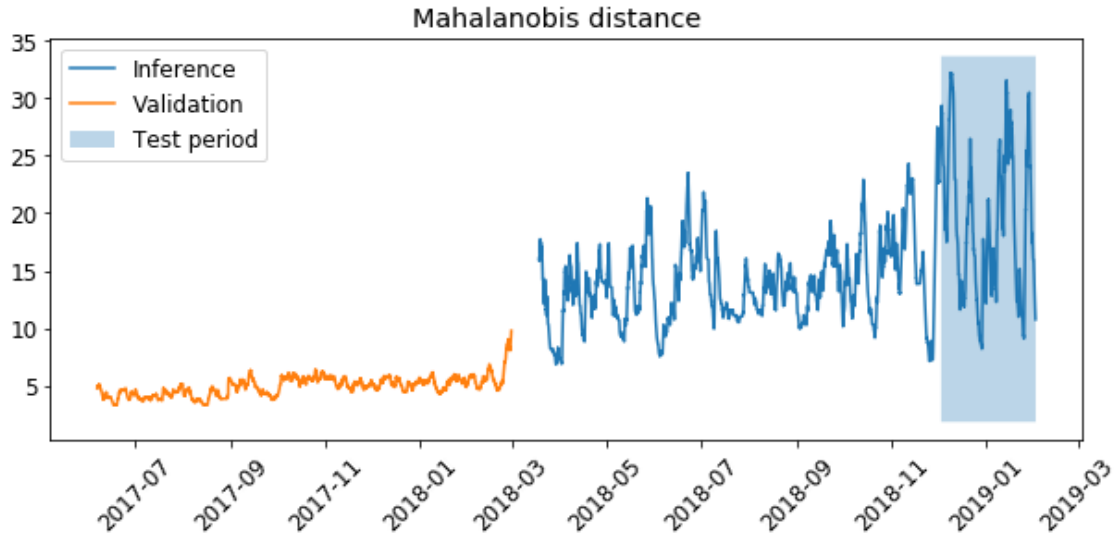
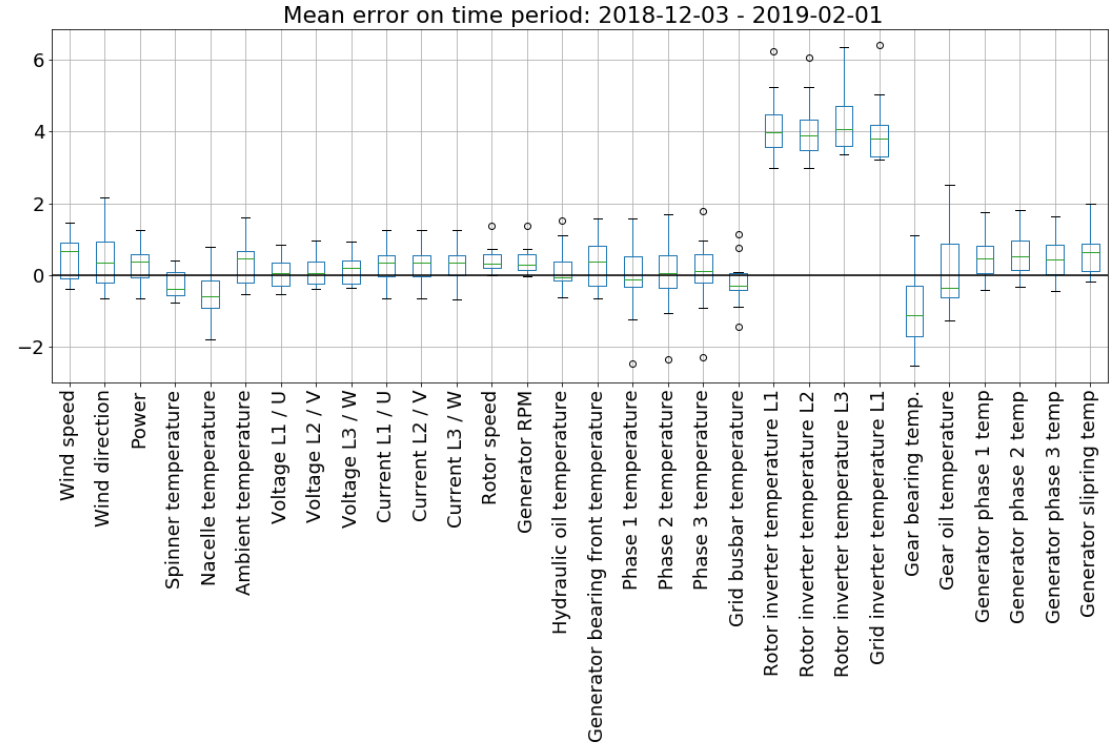


Figure A.1: The connection map for ZCA whitened data, created by adding a distortion to each signal one by one. The columns represent the distorted signal and the rows are the effect on the signals. The colouring is done per column, where the highest value in each column is coloured dark red and the lowest value dark blue. The colours are not comparable between columns.

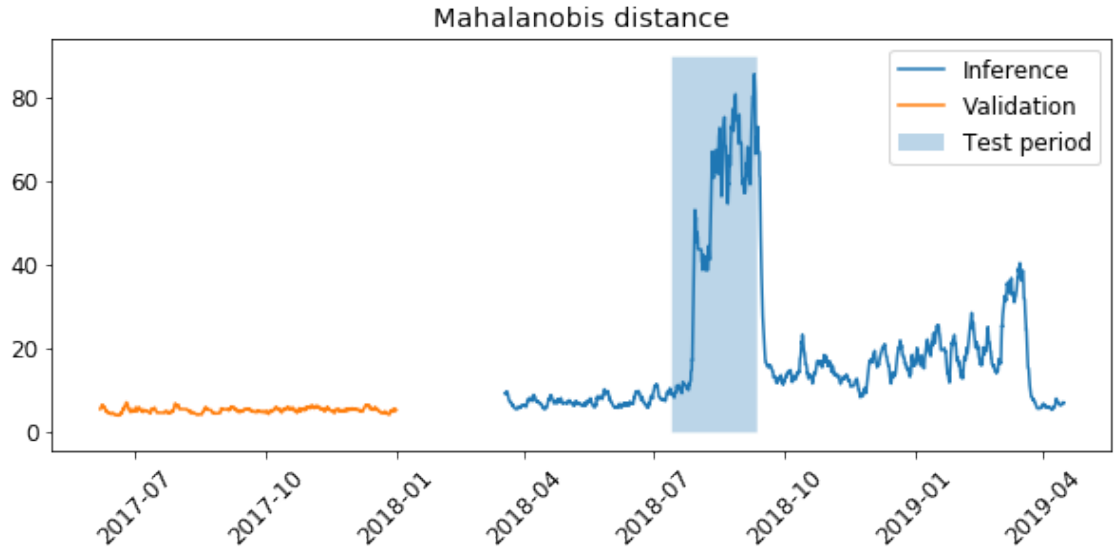


(a) Mahalanobis distance on validation and inference residual. Test period marked.

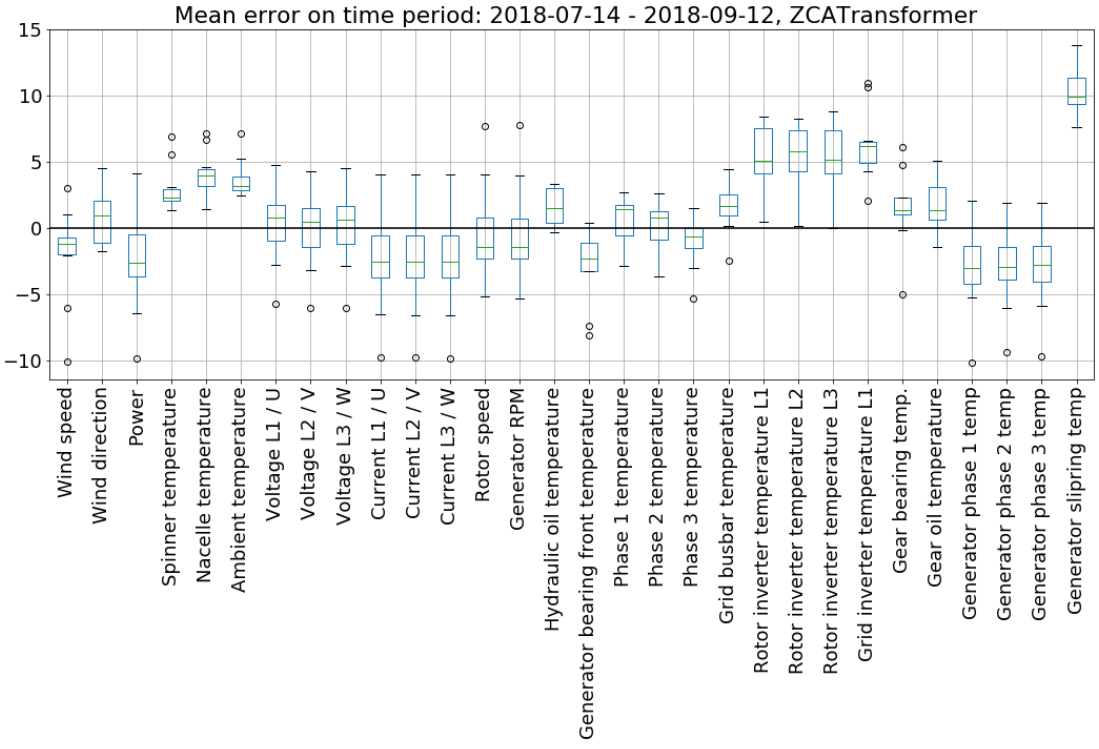


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure A.2: Data from **Wind farm A: Turbine 1**, transformed using ZCA whitening. The turbine was experiencing **cooling system issues** according to service logs. According to the existing anomaly detection method the signals *Rotor inverter temperature L1, L2, L3* and *Grid inverter temperature L1* were high. In (a), the test period for this test is marked.

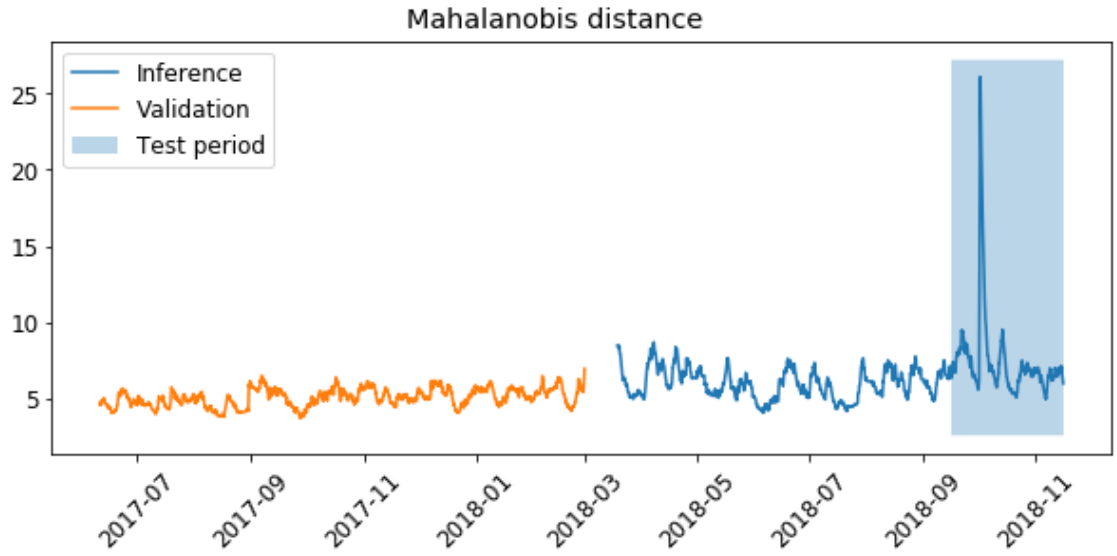


(a) Mahalanobis distance on validation and inference residual. Test period marked.

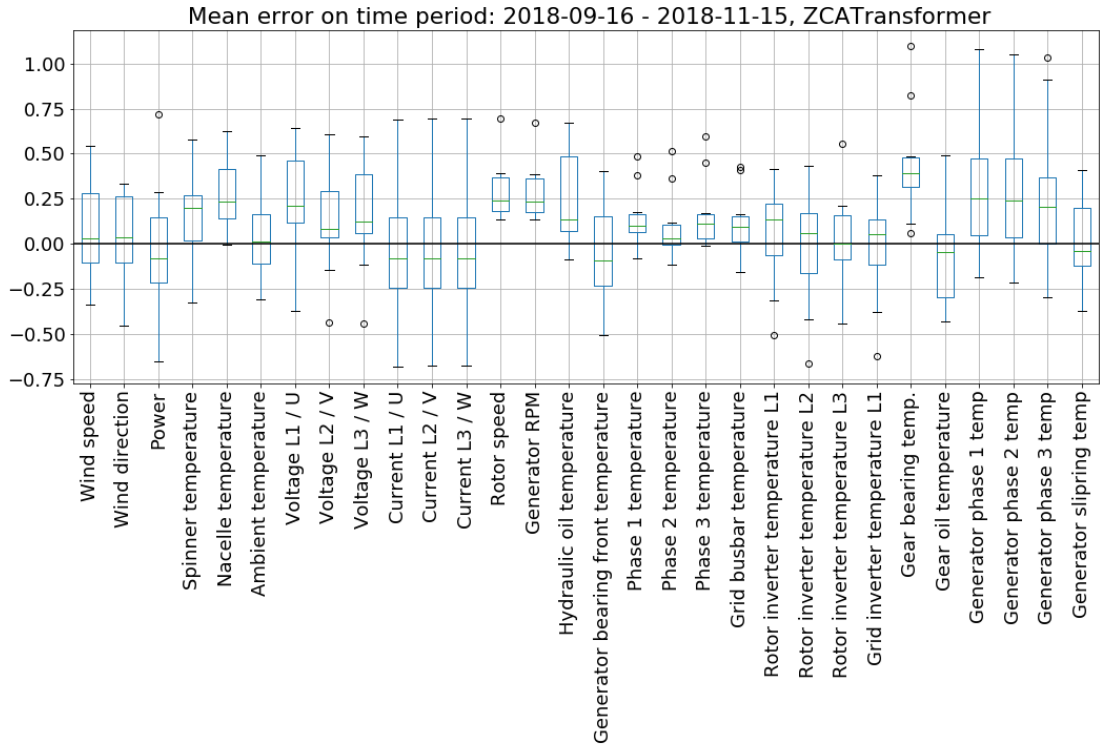


(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure A.3: Data from **Wind farm A: Turbine 5**, transformed using ZCA whitening. The turbine was experiencing a **failure in the generator slipring hose** according to service logs. According to the existing anomaly detection method the signal *Generator slipring temperature* was high. In (a), the test period for this test is marked.



(a) Mahalanobis distance on validation and inference residual. Test period marked.



(b) Boxplot on the mean values from the test period. Results from 10 autoencoders.

Figure A.4: Data from **Wind farm A: Turbine 4**, transformed using ZCA whitening. **Gearbox failure** according to service logs. In (a), the test period for this test is marked.

B

Gaussian residuals

This Appendix examines if the residual from applying the autoencoder on data from a healthy wind turbine can be approximated by a Gaussian distribution. This is done by plotting the distribution of each residual signal and comparing it with a Gaussian distribution with parameters estimated from the data. The data used for this was the training data from Wind farm A; Wind turbine 1 and the result is shown in Figure B.1 and B.2. As seen in the figures, many of the signals are indeed following a Gaussian distribution, while the exceptions, like the *Rotor speed* and *Generator RPM* in Figure B.1, or the *Rotor inverter temperature L1-L2* in Figure B.2, are still within the boundaries of a Gaussian distribution. No signal is heavy tailed.

Note that the variance differs between the residual signals shown in Figure B.1 and B.2. This is due to the difference in variance in the original signals. Due to the difference in variance, and the non-zero mean values, the residual should be standardized with respect to the residual from healthy data before the residual signals are compared to each other.

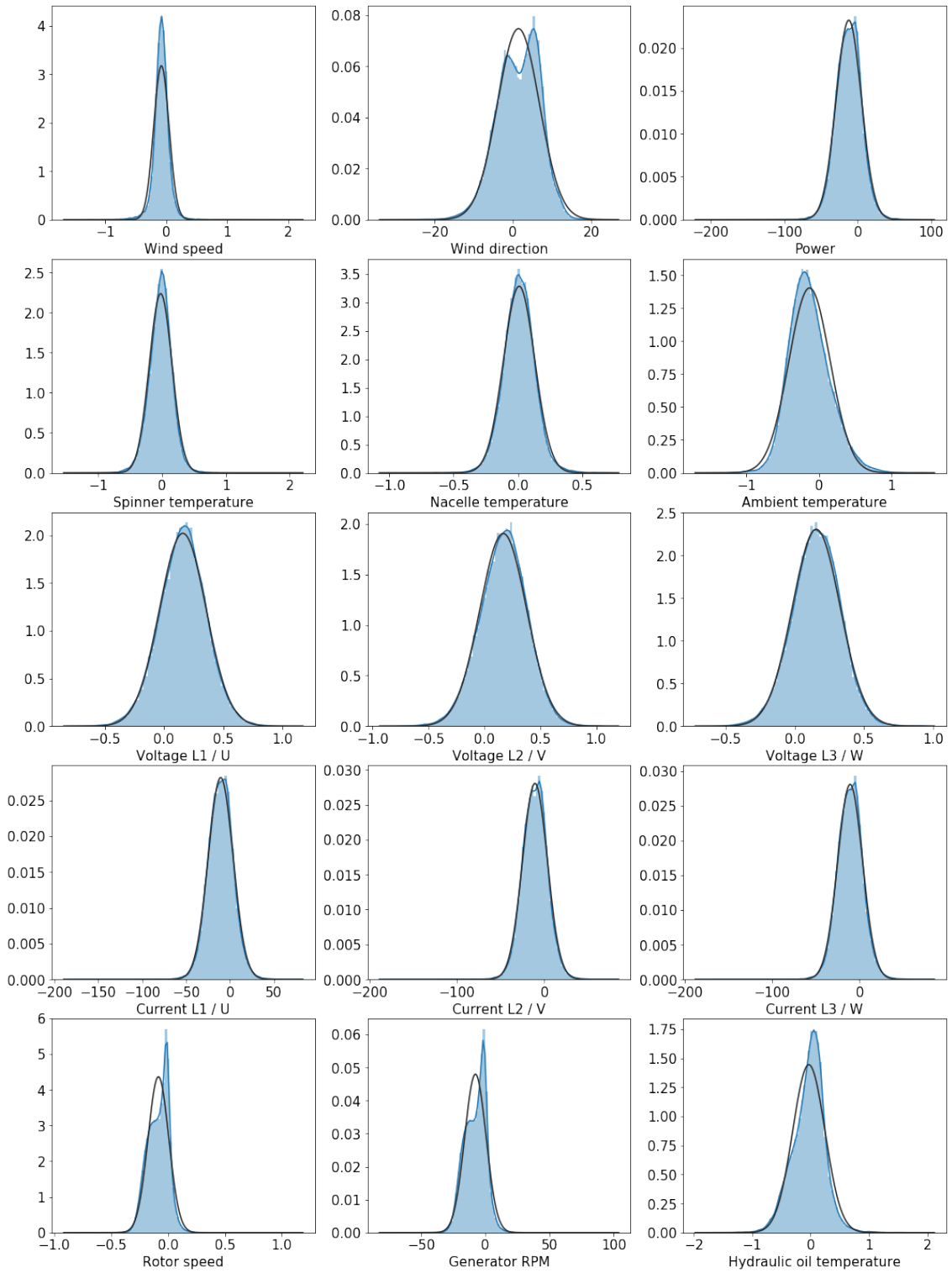


Figure B.1: Distribution of the first 15 residual signals of the training data from Wind farm A; Wind turbine A. The black line represents a Gaussian distribution with parameters estimated from the data.

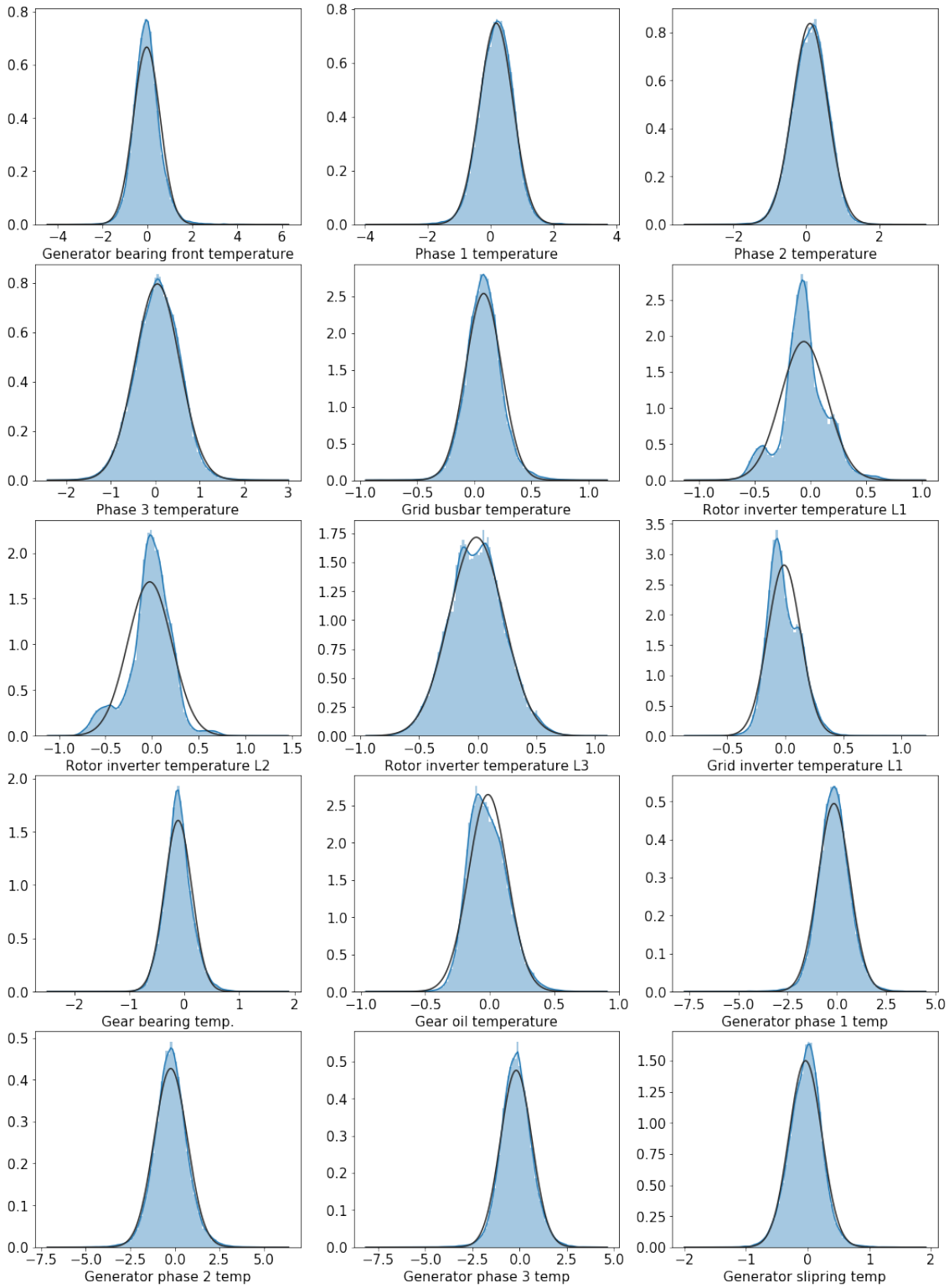


Figure B.2: Distribution of the last 15 residual signals of the training data from Wind farm A; Wind turbine A. The black line represents a Gaussian distribution with parameters estimated from the data.