Magnus Gribbestad

# Prognostics and Health Management for Air Compressors Based on Deep Learning Techniques

Master's thesis in Simulation and Visualization
Supervisor: Ibrahim A. Hameed, André L. Ellefsen, Vladimir Krivopolianskii
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

# Summary

Today, typical maintenance strategies on air compressor systems depend on doing scheduled maintenance actions based on experience, and repairs after failure. The overall goal of this thesis is to explore prognostics and health management (PHM) for air compressors based on deep learning techniques. It is researched in order to see the potential of replacing traditional maintenance strategies with modern predictive maintenance that can capture the actual condition of an air compressor. Three important features in such a system are investigated. First, anomaly detection is investigated to give a descriptive measure of how much the system is deviating from normal operating condition. Second, diagnostics is explored towards identifying faults and their severity. Finally, the topic of prognostics is investigated to predict time until an air compressor fails to operate. Prognostics is also explored towards benefiting from transfer learning and providing uncertainty bounds related to the predictions.

The case study on anomaly detection achieved promising results. Variational autoencoder (VAE) and long short-term memory (LSTM) with encoder-decoder architecture were able to give insight into how much the compressor deviated from expected behaviour. Both models were able to accurately separate between normal and faulty conditions. A method was proposed to increase the transparency of the anomaly detection approach. The results showed that each fault type followed a unique pattern of sensor contribution. The method was able to capture this information and give an indication of why the air compressor behaves unexpectedly. Diagnostics showed that feed-forward neural network (FNN), LSTM, and convolutional neural network (CNN) were accurately able to identify both faults and their severity. Predicting severity has the benefit of giving an earlier indication of potential faults. It can, on the other hand, be challenging to obtain severity labels. This makes the traditional fault identification approach more applicable. Results from prognostics proved that LSTM was the most accurate in predicting when a compressor will fail. Although most of the predictions were accurate, some predictions got too large errors. The concept of transfer learning in prognostics proved useful and were able to improve the predictions. It also has the potential to reduce the number of needed run-to-failure examples. A single-valued prediction can give an illusion of certainty. A data-driven approach was proposed for including uncertainty bounds to the predictions. It contributed to more realistic predictions.

The methods related to anomaly detection, diagnostics and prognostics that was investigated in this thesis are useful features in a PHM system. Together these features can improve the current maintenance strategy on air compressors by allowing online monitoring of the condition of a system. The suggested approaches have the potential to predict when a compressor will fail and why.

# Acknowledgements

I am profoundly grateful to my supervisors; Ibrahim A. Hameed, André L. Ellefsen and Vladimir Krivopolianskii. They have provided indispensable guidance and valuable support throughout the thesis. I would also like to thank Vilmar Æsøy for introducing me to Sperre Industri AS and thereby make this thesis possible.

Besides my supervisors, I would like to thank Sperre Industri AS for their collaboration. They have shown great interest in the project and been available for discussions and domain-knowledge. I want to direct a special thanks to Freddy Stene for the assistance when collecting data from the air compressor.

Finally, thanks to my family and friends for all the support, encouragement and motivation. This work would have been impossible without them.

# Preface

This master thesis is submitted as the final work of the Master of Science degree at the Simulation and Visualization program at the Norwegian University of Science and Technology (NTNU), Department of ICT and Natural Sciences. The research and report are done during the final semester, spring 2019. It has been performed with Sperre Industri AS as a collaborating partner, providing access to data from their products. They are a respected supplier of air compressors.

This thesis aims to explore prognostics and health management for air compressors using deep learning techniques. It is investigated if it has the potential to improve the current maintenance strategies used for such systems. The main parts of the thesis are to use deep learning to detect anomalous behaviour, identify faults, and predict failures. I was inspired to pursue this topic since I am interested in deep learning and data analysis, especially towards predictive maintenance. I also have background knowledge of predictive maintenance from my bachelor thesis on a related topic.

# Table of contents

# Figures

# Tables

# Abbreviations

| | |
|---|---|
| AAE | adversarial autoencoder |
| ACO | ant colony optimization |
| AE | autoencoder |
| AI | artificial intelligence |
| ANN | artificial neural networks |
| CBM | condition based maintenance |
| CM | corrective maintenance |
| CNN | convolutional neural network |
| DAE | denoising autoenconder |
| DBN | deep belief network |
| DL | deep learning |
| ED | encoder-decoder |
| FNN | feed-forward neural network |
| GA | genetic algorithm |
| GAN | generative adversarial network |
| GRU | gated recurrent unit |
| HP | high pressure |
| LP | low pressure |
| LSTM | long short-term memory |

MAE . . . . . mean absolute error

ML . . . . . machine learning

MLP . . . . . multi-layer perceptron

MSE . . . . . mean squared error

MTTF . . . . . mean-time-to-failure

NN . . . . . neural network

OSVM . . . . . one-class support vector machine

PCA . . . . . principle component analysis

PdM . . . . . predictive maintenance

PHM . . . . . prognostics and health management

PM . . . . . preventive maintenance

PSO . . . . . particle swarm optimization

RBM . . . . . restricted Boltzmann machine

RMSE . . . . . root mean squared error

RNN . . . . . recurrent neural network

RUL . . . . . remaining useful life

SAE . . . . . sparse autencoder

SOM . . . . . self-organizing map

SVM . . . . . support vector machine

SVR . . . . . support vector regression

VAE . . . . . variational autoencoder

# Chapter 1

# Introduction

In this chapter, the background and motivation for this thesis are introduced. Further, the scope and objectives are stated and described. The chapter continues with information about a confidentiality requirement from the collaborating company. Finally, the structure of the thesis and the remaining chapters are described.

## 1.1 Background & motivation

Autonomous ships have in recent years received much attention in the maritime industry. The concept of these ships is more than just autonomous navigation. It will also be important that systems and equipment on board are operational and reliable [1]. In other words, if unmanned autonomous ships will be a reality, they will be dependent on monitoring the condition of systems and predict when vital equipment will fail [2]. This is necessary to plan maintenance to the best possible time, for instance, when a ship is in port. Such requirements create a demand for suppliers of ship equipment such as generators, propellers, and compressed air systems to improve their maintenance strategy.

Today, many industries, including the maritime sector follows traditional maintenance strategies such as corrective maintenance (CM) and preventive maintenance (PM) [3]. These strategies are based on doing maintenance after a system fails (CM) or doing periodic inspections and repairs to attempt to keep the system in satisfactory operational condition (PM). Waiting until a system breaks down and then do the repairs, is leading to unreliable systems with unexpected standstills. PM is a strategy used to reduce the risk of unexpected failures and increase the life of systems based on experience and expected life statistics [4]. Drawbacks with this approach are that in most cases, the condition of the system is not taken into account. This can lead to using a lot of resources on changing parts that are still in good condition or experience unexpected failures [5].

In maritime systems, failures often happen in a seemingly irregular pattern [6], which makes strategies based only on PM unsuitable. An unexpected breakdown on a ship can be critical, especially with no personnel on board. This means that CM is not applicable for equipment on unmanned autonomous ships. A predictive maintenance (PdM) strategy is more appropriate, as it can be used to monitor the

condition of equipment and predict when it will fail [1]. Prognostics and health management (PHM) has proved itself to be a promising engineering discipline for obtaining a PdM strategy. It has shown successful implementation in industries such as aerospace [7] and automotive [8]. In general, PHM solutions are based on detecting faults, predicting failures, and providing decision support. It has the ultimate goal of zero-downtime performance [9]. Such systems can contribute to economic benefits, since maintenance is done when necessary, instead of before or after. Besides, it can increase reliability and safety by reducing unexpected failures and standstills.

The idea of PdM and PHM is not new, research on these topics has been conducted for many years. The focus has been on model-based and traditional data-driven approaches [10, 11], while in modern research, some of the focus has shifted towards machine learning (ML) and deep learning (DL) techniques [12]. These approaches have shown promising results for detecting abnormal behaviour (anomaly detection) [13], recognizing faults (diagnostics) [14] and predicting failures (prognostics) [15].

Sperre Industri AS is a supplier of air compressor systems and they state that every fifth ship sails with their products [16]. Today, they are following a PM strategy, which forces them to keep a large inventory with necessary parts for up to 30 years old compressors. Their service concept promises customers a replacement part within 48 hours, everywhere in the world. They are currently in the process of developing a PHM system for their products. Having such a system can improve their current maintenance strategy and contribute to better products, service, and customer support. The thesis is carried out in collaboration with Sperre Industri AS and all experiments are conducted on their air compressors.

The motivation for this thesis is based on two main aspects. The thesis will contribute towards improving the currently used maintenance strategies on air compressors in general. This is relevant to the future vision of unmanned autonomous ships since it aims to increase the reliability of vital ship equipment. The second aspect is based on defining research towards DL for PHM. Traditional methods are often application-dependent, due to manual work such as feature engineering [17]. DL techniques are promising to overcome the limitations and lack of flexibility in the traditional methods by being able to work with unlabelled and complex noisy real-world data [18].

## 1.2 Scope

The scope of the project is to explore DL techniques for PHM on air compressor systems. The techniques are used to recognize patterns in data and contribute to determining the condition of the system. Experiments are conducted on an air compressor from Sperre Industri AS, but the thesis aims to provide suggestions on flexible methods to use in a PHM concept. The scope lay within the boundaries of predictive maintenance (PdM), deep learning (DL) and air compressors as indicated in figure 1.1.

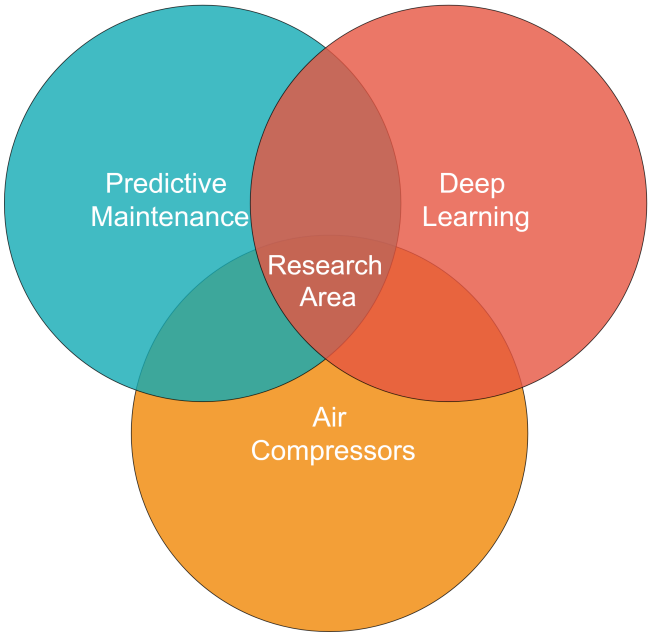



Figure 1.1: Venn diagram of the thesis scope

## 1.3 Objectives

According to Goebel [19], the field of PHM is concerned around doing a systematic assessment of a system's state of health. The overall goal and research questions of this thesis are therefore related to exploring the three questions a PHM system should answer [19]. These questions are:

- Is everything going fine? (Anomaly detection)
- If not, what is wrong? (Diagnostics)
- If something is wrong, when are things going to break? (Prognostics)

The overall goal of this thesis:

> *Explore approaches on how DL can be used in a PHM solution to detect anomalous behaviour, identify faults and predict future failures on air compressors.*

The stated goal includes exploring if a PHM solution based on DL techniques can improve the current maintenance strategy on air compressors. The central part revolves around exploring how DL can be used for anomaly detection, diagnostics and prognostics. In this context, prognostics are predictions of remaining useful life (RUL), while diagnostics are in terms of identifying potential faults. Anomaly detection is to detect if the system is behaving anomalously or outside of normal operating condition. The techniques should be implemented as a proof of concept to show that this is feasible on an air compressor from Sperre Industri AS. In this thesis, the research questions (RQs) stated below are investigated and answered.

**RQ1:** How can DL be used to detect abnormal behavior in air compressor systems?

Detecting abnormal behaviour is strongly related to anomaly detection and determining if a system is behaving as expected. In this thesis, it is investigated towards giving an anomaly score that can describe how much the system is deviating from the normal condition. Usually, anomaly detection algorithms are considered black boxes, only indicating that something is an anomaly, or not. The thesis explores if the model can be made more transparent and assist in determining why the system deviates from normal condition.

**RQ2:** How can DL be used to identify faults in air compressor systems?

Identifying faults is considered as a part of diagnostics in a PHM system. In this thesis, different DL techniques are explored towards identifying faults on air compressors, and predicting their severity.

**RQ3:** How can DL be used to predict the remaining time until failure, and how to emphasize the typical problem of few run-to-failure examples?

Predicting the remaining useful life (RUL) is considered an important part of PHM. Several DL techniques and labelling approaches are compared. The typical problem of few run-to-failure examples is also explored by using transfer learning. A single-valued RUL prediction can give an illusion of certainty. A method for obtaining uncertainty bounds are explored for giving more realistic predictions.

**RQ4:** What are the advantages and disadvantages of using DL in a PHM system, and how does it improve the current maintenance strategy on air compressors?

While the three first research questions are related to both research and practical experiments, the fourth is related to the impacts of using DL for PHM. That means discussions about the advantages and disadvantages, and how such a system can improve the current maintenance strategy on air compressors. Several air compressor suppliers are investigated to get an overview of typical maintenance strategies of such products.

## 1.4 Confidentiality requirements

This project is executed in collaboration with NTNU and Sperre Industri AS. For the collaboration to take place, a non-disclosure agreement is signed to protect valuable information and domain-knowledge within the company. Therefore the data used in this thesis cannot be fully disclosed. Section 4.2 will explain some details about the data and how it is collected, but due to the non-disclosure agreement, sensitive information is withheld from this thesis. In addition, the code developed during the thesis cannot be disclosed.

The main impact of the confidentiality requirements is that the fault types and the time units of the collected data could not be revealed. This is considered sensitive information from the company and

of their best interest to keep undisclosed. The fault types have been referred to with only letters to be able to separate them. It might have made it harder to relate to the thesis, but the analytic foundation and results are not impacted. Ideally, disclosing the time units could make the thesis more descriptive, but since the faults in the data are forced in an unnatural speed, it is irrelevant to include the information. The time units can be considered as seconds, minutes, hours, or cycles.

## 1.5 Thesis structure

The layout of the thesis is as follows:

**Chapter 2 - Theory:** Describes relevant theory for this thesis. This includes theory about air compressors, maintenance concepts, PHM and DL. The chapter explains the DL approaches that will be used, in detail.

**Chapter 3 - Related work:** Explores research relevant to this thesis. The chapter gives a summary of PHM with traditional methods, but also modern approaches for anomaly detection, diagnostics and prognostics. An introduction to how maintenance is done on air compressors today is also presented.

**Chapter 4 - Methodology:** Presents the methodology used in this thesis. This includes data collection, implementation details, and an overview of the three cases studied in this thesis. These cases are anomaly detection, diagnostics and prognostics.

**Chapter 5 - Case A: Anomaly detection:** Describes and presents results from the anomaly detection experiments. It contains two sub-parts, one for online anomaly detection and one for unsupervised fault detection in historical data.

**Chapter 6 - Case B: Diagnostics:** Presents the results related to identifying faults in air compressors and predicting their severity.

**Chapter 7 - Case C: Prognostics:** Presents the results from the prognostics experiments. This includes comparing different DL techniques for RUL predictions, exploring two labelling approaches, trying to improve predictions with transfer learning, and providing more realistic predictions with uncertainty bounds.

**Chapter 8 - Discussion:** Discusses the results, data foundation and PHM for air compressors in general.

**Chapter 9 - Conclusion:** Contains the conclusion by answering research questions, stating the contributions from this thesis, and presenting ideas for future work.

# Chapter 2

# Theory

## 2.1 Air compressors

Air compressors are a type of equipment that aims to increase the pressure of air by reducing the volume. Air with increased pressure or compressed air can be obtained in different ways. Reciprocating, rotary screw and rotary centrifugal are the three basic types of air compressors [20]. These types can have different specifications, such as which compression stages, cooling method, drive method, or lubrication. Air compressors are often connected to a tank which stores the pressurized air.

In this thesis, a two-stage reciprocating air compressor is used as a test subject and source of data. A reciprocating air compressor generates compressed air by using a piston as a displacement element inside a cylinder [21]. Typically, an electric motor is used as the source to make the piston move. A compressor like this can have several compression stages, which means more cylinders and pistons. This can be necessary if the compressor is to be working in higher pressure ranges.

Compressors like these can typically be a part of compressed air systems on ships. The main task of such a system is to deliver compressed air that is used to start main- and auxiliary engines. Since this is vital equipment, class regulations require redundancy by having two separate compressed air systems. Compressed air systems on board ships are often used for running machines, valves, doors and other miscellaneous equipment. It is often separated into systems for starting air and working air.

## 2.2 Maintenance

This thesis is primarily about PdM and PHM, but understanding the broad terms of maintenance is important. This section aims to give an overview of maintenance concepts and expressions. According to the maintenance terminology standards [22], maintenance is a combination of all necessary actions to keep a system or item in a state where it can perform its required function during its life cycle. Typically, maintenance is the steps and processes done in order to keep a system in a specified condition. The European Standard has defined several important terms within maintenance [22]. Figure 2.1 provide an overview of the main groups of maintenance categories. Maintenance is a broad topic

which usually is divided into three main categories; corrective, preventive, and predictive [5]. PdM is normally mentioned in combination with condition based maintenance (CBM) and is defined as a sub-category to PM. The next sections clarify these terms.

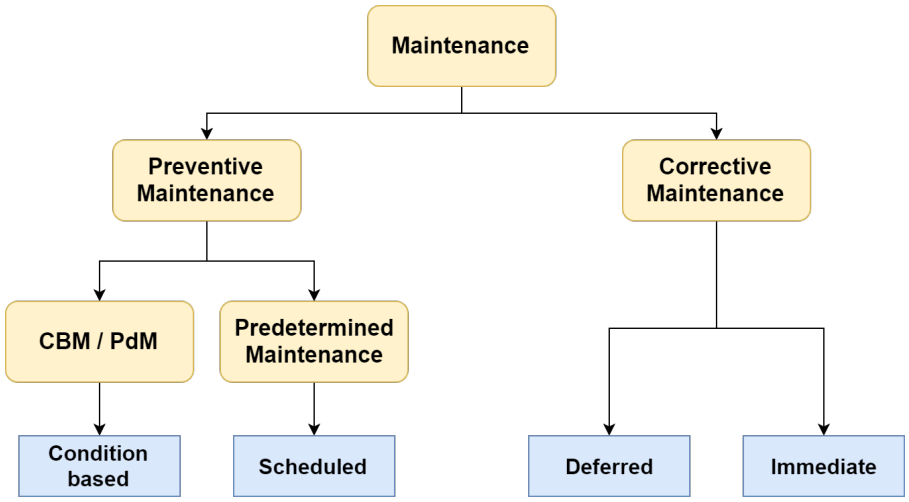



Figure 2.1: Overview of maintenance concepts [22]

### 2.2.1 Corrective maintenance

Corrective maintenance (CM) is the unplanned style of doing maintenance with the philosophy to fix something when it breaks [4]. Run-to-failure (RTF), breakdown maintenance, hysterical maintenance, or reactive maintenance are other terms also used for this category [4, 5]. CM refers to waiting until a machine or system breaks, to fix it. In other words, it is a strategy where no money is spent on maintenance until a system fails to operate. In general, it is the most expensive maintenance approach, and the main costs associated with this type of maintenance are related to inventory of spare parts, overtime labor, downtime and loss in production efficiency [4]. According to Mobley [5], a repair performed due to CM, costs on average about three times more than the same repair made within a PM concept.

### 2.2.2 Preventive maintenance

Preventive maintenance (PM) is actions that are done to keep a system in the preferred condition, by doing tasks based on elapsed time or hours in operation [4]. Doing basic preventive tasks such as lubrication, adjustments, and visual checks are normal efforts to improve reliability in systems. Instead of repairing a system after it fails, actions are executed on scheduled time to retain a system in working order [5]. Preventive maintenance is defined as follows in BSI EN-13306:2010 [22]: *"maintenance carried out at predetermined intervals or according to prescribed criteria and intended to reduce the probability of failure or the degradation of the functioning of an item"*. This is related to the philosophy of fixing something before it fails. Scheduled, predetermined, and cycle based maintenance are other terms used for this type of maintenance [5].

The idea is that these actions lower the probability of failures and extends the lifetime of the equipment. The maintenance activities are typically planned based on individual lifetime distribution of components or requirement from the manufacturer to full-fill warranty [5]. Mean-time-to-failure (MTTF) statistics are often used to determine these lifetime distributions and help to plan machine repairs or rebuilds. Figure 2.2 shows an example of a MTTF curve. The figure illustrates that a system usually has a higher probability of failure early due to start-up problems. Next, it goes into the normal life period, where the probability of failure is low. When the system is starting to tear, the probability of failure increases again.

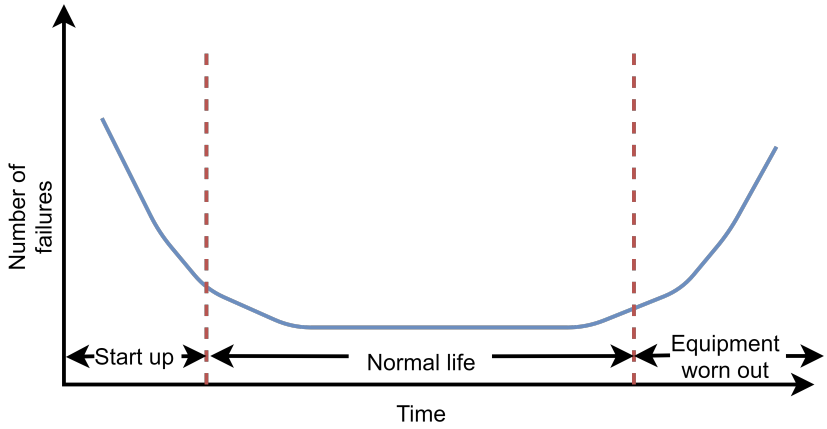



Figure 2.2: Example of MTTF-curve [5]

One of the problems with basing maintenance on MTTF statistics is that equal components can be used quite differently depending on what it is used for. Combinations of other equipment, settings and conditions can make a system last either longer or shorter than what is expected [5]. Therefore, either unnecessary repairs or catastrophic failure can be a result. The first case means that repairs are done to a system which actually shows no sign of degradation, which leads to wasted labor, parts, and downtime. The other case is even more costly and triggers unplanned standstill and repairs, which usually is much more expensive than planned repairs [5].

### 2.2.3 Predictive maintenance

Predictive maintenance (PdM) tries to avoid the previously explained case by predicting when maintenance should be executed [5]. The European standards [22] defines PdM as follows: *"condition based maintenance carried out following a forecast derived from repeated analysis or known characteristics and evaluation of the significant parameters of the degradation of the item"*. In general, this means to monitor the condition and efficiency of a system with sensors (temperature, vibration, images, etc.) to try to optimize the maintenance of a system [5]. PdM is not to make these measurements, but to use the condition that can be inferred from them, to optimize performance. This includes maximizing the time between repairs while minimizing unscheduled standstill and the cost of them.

PdM is therefore referred to as a condition-based, preventive maintenance method. CBM is often used in the same terms as PdM and means to do maintenance when the need arises [22]. While traditional PM uses average lifetime statistics for scheduling maintenance, PdM uses the actual condition of the system. PHM is a term closely related to PdM which often is used either as a term for an engineering discipline or as an implementation of the PdM strategy. PHM is covered in the next section.

### 2.2.4 Prognostics and health management

While PdM and CBM refers to the maintenance strategy, PHM is as mentioned a term used both for the engineering discipline and implementation of systems which follow the PdM / CBM strategy. In general, it aims to increase system reliability, availability, safety, and reduce maintenance costs [23] with the ultimate goal of zero-downtime performance [12]. Research in the field of PHM has tried to capture the health state of systems to provide decision support [23]. An important part of this is to predict the RUL of a component or system. This is a prediction of the future performance of a system based on potential degradation.

PHM consists of a set of steps adopted from CBM [12]. These steps revolve around data acquisition, data processing, diagnostics, prognostics, and decision support. Data acquisition refers to measuring and storing sensor data related to the condition of a system. Another type of data that can be collected is event-data, which can be information about when, where, and which failure occurred [24]. The data processing step involves cleaning and analysis of data. Cleaning is processes such as reducing noise and data compression, while analysis can be to extract potential condition indicators with, for instance, wavelet transform or frequency analysis [12].

As stated in section 1.3, Goebel [19] considers a successful PHM system to contain anomaly detection, diagnostics, and prognostics. The diagnostics and prognostics steps are important since an effective PHM system needs these in order to provide decision support. Diagnostics are about identifying faults and determining how serious a fault is [25]. Anomaly detection can also be an important part of diagnostics, as it aims to recognize when the system is outside of normal operation [26]. Anomaly detection can help to detect faults where there is not enough historical data to recognize them.

Prognostics refers to predicting the progression of a fault, which can help to prevent a failure from occurring. This means to predict the remaining time before a component is unable to operate as expected [12]. In research related to PHM, the estimated time until a failure is often mentioned as the remaining useful life (RUL) [27]. Ideally, RUL should include confidence intervals to increase reliability, which will make it easier to determine when to do maintenance [28].

Decision support and human-machine interface is the final part of a PHM solution and the part that makes anomaly detection, diagnostics and prognostics available for service personnel. Ideally, a system could also automatically take decisions regarding maintenance [29]. The RUL obtained from prognostics help to determine when to do maintenance, while diagnostics provide information on which parts of a system that needs maintenance. Anomaly detection can detect that there is something wrong with the system, which can be useful when diagnostics and prognostics fail. Approaches in PHM are typically divided into three different groups [12]. These are:

- **Data-driven approach:** Based on using pattern recognition and machine learning on historical data to estimate condition and predict RUL [30]. Modern approaches include the use of ML methods such as varieties of neural networks. Data-driven approaches usually need several examples of the system running until failure in order to learn the patterns. Such data can be hard to acquire, especially in a new system.
- **Model-based approach:** These approaches are based on creating accurate physical models of a system and use this to estimate RUL [31]. This can be mathematical models that represent the system or the actual degradation of the system.
- **Hybrid approach:** As the name suggests, this is a hybrid approach which tries to benefit from the strengths of both data-driven and model-based approaches.

Chapter 3 presents research done on PHM with traditional methods and more thoroughly on work related to this thesis, which is PHM with DL. First, DL and the algorithms used in this thesis are described.

## 2.3 Deep learning

In order to understand DL, it is first necessary to explain ML, which is about turning data into information [32]. It is considered a set of adaptive models that can enable computers to learn to find patterns in data based on examples [33]. Their learning capabilities makes them able to improve performance over time. ML algorithms can be considered a function, that automatically can learn the relationship between descriptive features and some target value. This allows the algorithms to make predictions or decisions by transforming a set of inputs $X$ into output(s) $Y$ [34].

DL is a category within the wider term of ML. The difference between traditional ML methods and DL is that DL emphasizes on learning successive layers of increasingly meaningful representation [35]. It is a set of models that have deeper representations, typically consisting of several layers of non-linear processing that can recognize more complex patterns in data [12]. The idea of DL is not new, but due to increased processing power in CPU, and especially GPU, DL has gained a lot of momentum [36].

DL has been successfully applied to many problems, such as face recognition [37], language translation [38], playing games [39] and stock price forecasting [40]. It has also been used within the field of PHM to recognize complex patterns such as degradation of health [27, 41, 42]. Most of the models in DL are based on artificial neural networks (ANN). It is an umbrella term for several types of algorithms that are vaguely inspired by the processing and communication of information in the biological neural networks. A standard ANN method called feed-forward neural network (FNN) is explained in detail in section 2.4.1. This contains several important terms and concepts that are common for ANN methods. ML algorithms are normally categorized into three categories based on how they learn [43]:

- **Supervised learning:** In these cases, a dataset with training examples containing both features and the targets/labels are available. Based on the training data, the model can learn to generalize and make correct predictions.

- **Unsupervised learning:** In the case of unsupervised learning, training data is available, but without the associated targets. This means that several of these algorithms aim to either find similarities in data (clustering) or infer features. Unsupervised learning is important in the topic of PHM since a typical problem in the industry is the lack of labelled data.
- **Reinforcement learning:** This is cases where the algorithm uses a system of reward and punishment to learn. After making a prediction or decision, the algorithm is given a score which is used to adjust and learn. This field will not be explored in this thesis.

The term semi-supervised learning is often mentioned, as well. This refers to using both labelled and unlabelled data to perform a supervised- or unsupervised-learning problem [44]. The reason for this combination is that in real-world problems, labelled data can be difficult to obtain. Therefore, combining large amounts of unlabelled data with labelled data can be an advantage.

ML and DL can solve several types of problems. Supervised learning is typically divided into two categories [43]:

- **Regression:** Try to model the relationship between inputs and output, where the output is a number. In prognostics, predicting RUL is an example of a regression problem.
- **Classification:** This is problems where the model attempts to recognize certain categories based on inputs. In diagnostics, this can be to identify faults.

In addition, anomaly detection is in some cases considered as supervised learning, but can also be semi-supervised or unsupervised. Anomaly detection is often used in many different terms, but in general, it can be considered to detect when something is out of the ordinary or an irregularity from the norm. In this study, it is to detect abnormal behaviour of a system or a component.

The DL models explored in this thesis is explained in more detail in section 2.4. The detailed description contains concepts such as neurons, layers, activation functions, learning rate, and much more. These concepts are often referred to as hyper-parameters and are parameters that need to be set before training or optimized as a part of the training process.

## 2.4 Deep learning algorithms

This section explains the theory and brief mathematics behind the DL algorithms that are used in this thesis. The following algorithms are explained and were explored towards PHM in this thesis:

- **Feed-Forward Neural Network (FNN)**: Supervised learning algorithm that is used for regression and classification. Much of the theory on FNN is general for all ANN methods and layers of FNN is often combined with convolutional neural network (CNN) or long short-term memory (LSTM).
- **Long Short-Term Memory (LSTM)**: Supervised learning algorithm that is specialized in working in sequential data. It is a type of recurrent neural network (RNN).

- **Convolutional Neural Network (CNN)**: Supervised learning algorithm known for its performance on 2D- and 3D-data, but it can also be applied to 1D-data.
- **Autoencoder (AE)**: Unsupervised learning algorithm used for feature extraction and anomaly detection. Often combined with a supervised decision layer. Sparse autencoder (SAE) and variational autoencoder (VAE) is also described.
- **Deep Belief Network (DBN)**: Unsupervised learning algorithm used for feature extraction. Often combined with a supervised decision layer.

### 2.4.1 Feed-forward neural network

Feed-forward neural network (FNN) are a type of cyclic ANN. It is considered to be the first and simplest type of ANN. A FNN consist of multiple, simple, processing units called neurons, organized into layers [33]. A neuron can have multiple inputs, but only one output, which again can be distributed to other neurons. Neurons are connected together with weighted connections that are used to transfer signals. Neurons in the input layer get activated from input data, while neurons in other layers are activated through weighted connections [33]. The output of a neuron is either an input to another neuron or an output of the model. Figure 2.3 shows an example of the architecture of an artificial neuron. The output of a neuron is determined from the sum of the weighted inputs passed through an activation function.

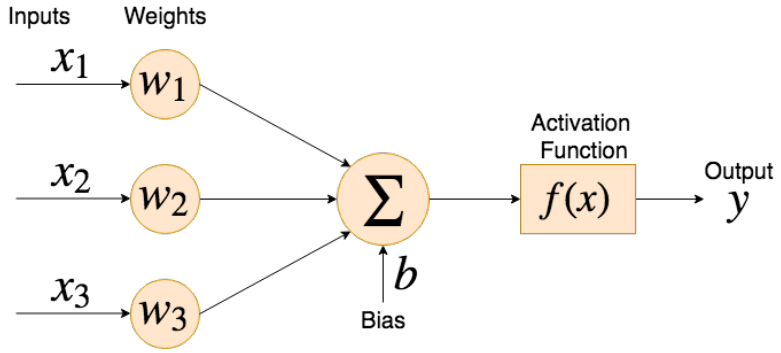



Figure 2.3: Overview of architecture and components of an artificial neuron

A neuron, $n_i$ has the inputs $x_1$, $x_2$, ... , $x_n$ and the output $y$. First, the sum of the weighted inputs, $o_i$ are calculated with equation 2.1. In this equation $i$ is the number of the neuron and $j$ is the index of the total number of $N$ inputs and weights, $b$ is the bias and $w$ is the weights [33].

$$o_i = \sum_{j=1}^{N} w_{i,j} x_j - b_i \tag{2.1}$$

The output, $y_i$ is found by passing the sum of the weighted inputs, $o_i$ through an activation function,

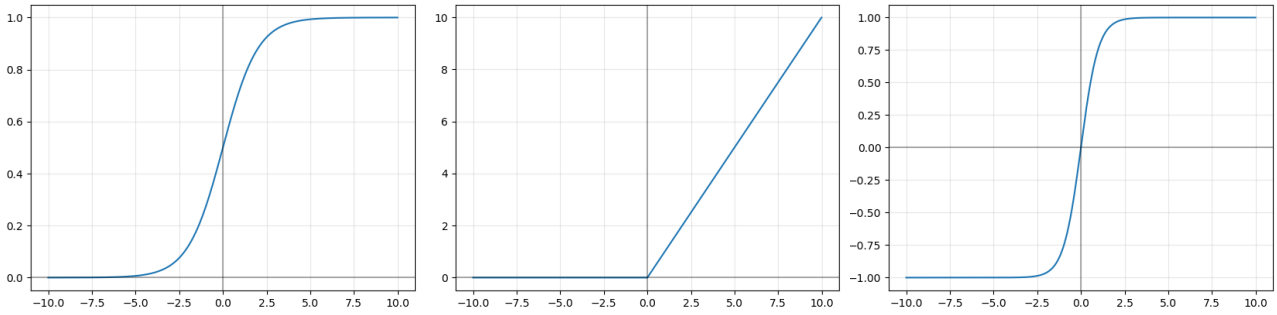
$f$. The output from a neuron is therefore given by equation 2.2 [33].

$$y_i = f(o_i) = f(\sum_{j=1}^{N} w_{i,j}x_j - b_i) \tag{2.2}$$

Assuming the inputs, $X_j$, and the weights, $W_j$, are structured as 1D-vectors, the formula can be expressed with with vector multiplication as shown in equation 2.3.

$$y_i = f(o_i) = f(w^T x - b) \tag{2.3}$$

The activation function, $f$, can be one of many variants that can define the output of a neuron, given the inputs. Figure 2.4a, 2.4b and 2.4c shows three popular activation functions called sigmoid, Rectified Linear Unit (ReLU) and hyperbolic tangent (tanh), respectively [45].



(a) Sigmoid (b) ReLU (c) Tanh

Figure 2.4: Three common activation functions

The sigmoid function (equation 2.4) gives a bounded output between 0 and 1, which can be interpreted as a probability in classifications [45]. The ReLU function (equation 2.5) is the most popular activation function for deep learning models. It returns either 0 or a positive number. It has benefits due to its simple calculations and is known to speed up convergence and accelerate training [45]. Other relevant activation functions are tanh and identity. The identity activation function is often referred to as the linear activation function and is simply a sum of the weighted inputs with no confinement on the range. The tanh activation function provides a confinement between -1 and 1 with a similar shape as the sigmoid function. The output is determined with equation 2.6.

$$f_{sigmoid}(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

$$f_{relu}(z) = max(0,\ z) \tag{2.5}$$

$$f_{tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.6}$$

Neurons are as mentioned organized into layers and figure 2.5 shows an example of a simple FNN. A network typically has an input layer, output layer and one or more hidden layers. The output from each neuron can be calculated layer-wise, which results in a final output of the network [33]. The

process of calculating the activation from the neurons is referred to as the forward pass.

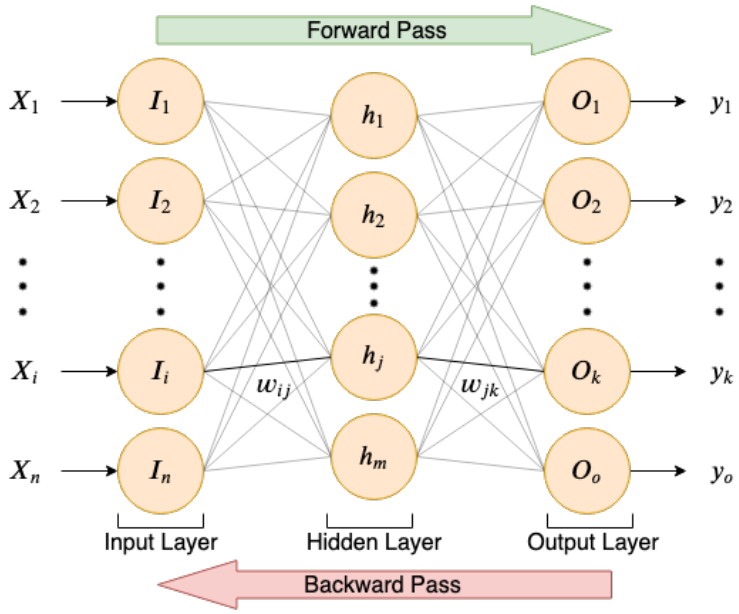



Figure 2.5: FNN architecture with an indication of forward and backward pass [33]

The final output of the network is determined by the inputs, weights, bias, and activation functions. The weights are typically initialized randomly, which means that initial predictions or outputs are also random [33]. Therefore, the next step is to update the weights through training. The training process is often referred to as the backward pass and is the process of trying to improve the output of the network by updating the weights [33]. This is done by minimizing the error between the output and the desired output. This is captured in a loss function, which in general are a measure of how incorrect the output of a network is. It is important to choose a loss function which correlates with success, as the network will take any shortcut it can. A typical loss function for a regression problem is the mean squared error (MSE) [34]. MSE is shown in equation 2.7, where $E(w)$ is the loss, $N$ is the number of outputs, $t_i$ is the desired output and $y_i$ is the actual output.

$$E(w) = \frac{1}{N}\sum_{i=1}^{N} ||t_i - y_i||^2 \tag{2.7}$$

The goal of the backward pass is to minimize the given loss function by adjusting the weights. For each time the weights are trained, they are adjusted with a small $\Delta w$. The weights are updated using gradient decent $\nabla E(w)$ [33]. This means that the weights are updated in a structured way, in order to reach the minimum error. As the gradient of $\Delta E(w)$ approaches 0, the error rate also approaches zero or are converging. In the backward pass, a technique called backpropagation is normally used. Backpropagation is a technique for propagating the error backward from the output and through the network, towards the input layer [46]. This allows the gradient of the error to be calculated in each layer and thus adjust the weight and bias subsequently. Several different implementations of gradient descent can be used to optimize training. The list below mentions some common approaches:

- Stochastic gradient descent (SGD) calculates the error and updates the parameters for each training example. In this thesis, a variant called mini-batch SGD is used. This approach takes a batch of samples before updating the parameters [47].
- Adagrad is a variant of SGD which has individual learning rates for each parameter. This can increase the learning rate for sparse features while decreasing it for the opposite. A more detailed description can be found in [48].
- RMSProp is also a variant of SGD with individual learning rate. The method normalizes the gradient by using a moving average of squared gradients. The purpose of this is to control the step sizes. More details can be found in [49, 50].
- Adam Optimizer is considered a newer variation of the RMSProp. It differs by using not only the average of the moment of the gradients, but also for the second moments. The complete description can be found in [51].

The gradient descent can guarantee a global minimum for a convex function. Most real-world problems are not convex and might have several local optimums the gradient descent can converge to. The learning rate $\gamma$ is introduced to reduce the chance of this. It determines how fast learning is applied, or in other words, how far each step down the gradient is [52]. This is done by multiplying the learning rate with the gradient of the loss function, as shown in equation 2.8. A large learning rate means the gradient takes large steps, while a small learning rate means small steps.

$$\Delta w^{t+1} = \mu \nabla E(w^t) \tag{2.8}$$

where $\Delta w^{t+1}$ refers to the change in weight at step $t + 1$.

Another concept called momentum $\alpha$ is also used when updating the weights. The concept of momentum is that a weight adjustment at step $t$ is dependent on the adjustment at step $t - 1$ [46]. The momentum is a value between 0 and 1, where 0 means that the weight change is only dependent on the gradient, while 1 means the update is only dependent on the previous weight adjustment. Equation 2.9 describes the relationship between the learning rate $\gamma$, momentum $\alpha$, gradient and weight adjustment.

$$\Delta w^{t+1} = (1 - \alpha)\mu \nabla E(w^t) + \alpha \Delta w^{t-1} \tag{2.9}$$

A problem with ML in general, is over-fitting. This can be translated to memorizing. An over-fitted model has an overly complex model for representing the pattern. In most ANNs, a normal regularization method to avoid over-fitting is called dropout [53]. The idea of dropout is to force the network to learn different representations of the data by "turning off" a share of the neurons in each training stage. The remaining neurons in each training stage will be updated. Normally, the dropout is given as a number between 0 and 1, where 0 means no dropout and 1 means none of the neurons are trained. A typical way of detecting over-fitting is if the model performance on the test data starts to increase. Therefore, another common regularization approach is called early-stopping. It aims to stop training before the model starts to over-fit. Both of these regularization techniques are used in this thesis.

### 2.4.2 Recurrent neural network

Recurrent neural networks (RNNs) are a group of NNs designed to recognize patterns in sequential data, such as text or time series. It has proved to be successful in for instance natural language processing (NLP) [54] and time-series forecasting [40]. RNN differ from FNN by introducing memory, which can connect past information to current. This is considered useful since some patterns and information are in the sequence itself [54]. This means that RNN can access data from both the present and recent past, while FNN only cares about the present.

Traditional RNN share weights across sequences or time-steps. This led the technique to suffer from a problem called vanishing or exploding gradient [55]. This can occur when the gradient is calculated through backpropagation. The weights $w$ will be multiplied by itself several times, which leads to a resulting weight $w^t$ which goes towards zero or infinity. In a network, this means that the error will go towards zero in the first layers of a network, which means that it will take much longer time to train those layers, than the later ones. In other words, traditional RNN struggle with long-term dependencies. Several variants of RNN has been proposed to deal with the vanishing gradient problem [55]. Two of the most popular variants are called LSTM and gated recurrent unit (GRU).

#### Long short-term memory

LSTM is a variant of RNN designed to learn long-term dependencies [55]. The LSTM introduces the idea of a memory cell, which contains gates that tries to regulate the information through the cell. The result is a network that achieves contextual weights that can deal with long-term dependencies in a flexible manner. Several variants of the LSTM has been introduced, such as the Vanilla LSTM [56] and GRU-LSTM [57]. The Vanilla LSTM has proved itself popular for PHM, therefore, it is the preferred variant of LSTM in this project. Many variants of the Vanilla LSTM exist [58]. In this thesis, the Vanilla LSTM without peephole connections were used. The Vanilla LSTM (referred to as just LSTM from now on), has four interacting NN layers. The architecture of a LSTM and its memory cell is illustrated in figure 2.6.

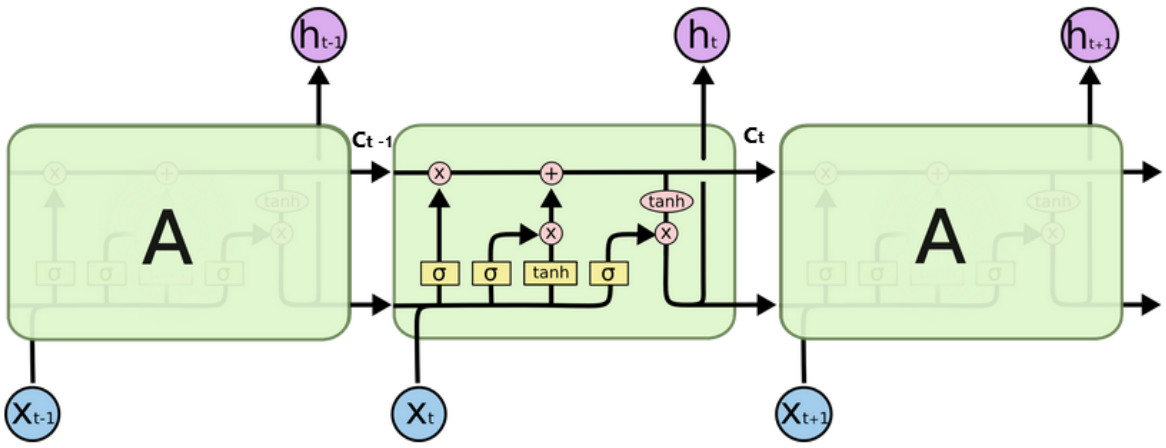


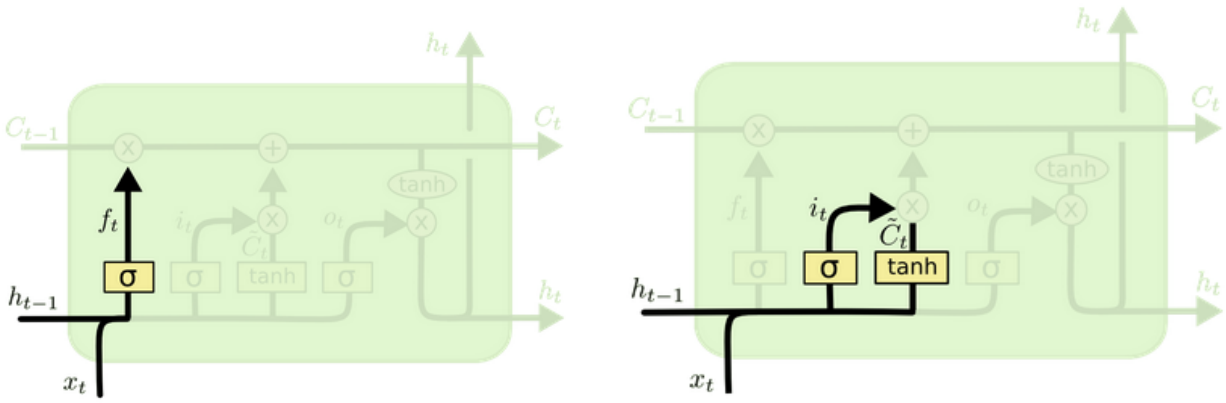

Figure 2.6: Basic components of the LSTM architecture and its memory cell [59]

The symbols in the figure represents:

- Yellow squares: a neural network layer
- Red circle: a point-wise operation
- Arrow: vector transfer
- Merging arrows: concatenation
- Splitting arrows: copy of vector

The upcoming description of the steps and layers of a LSTM is inspired by an extensive blog explaining LSTM in detail [59]. The cell state $C_t$ is an important part of the LSTM. It is represented by the top vertical line in figure 2.6. It is regulated from three of the NN layers, which often is referred to as gates. In the upcoming explanations and formulas, the $w, b, x$ and $h$ refers to the weights, biases, inputs, and outputs, respectively. The sub-scripted letter on $w$ and $b$ refers to which of the layers it is located to. For instance, $w_f$ means the weights in the forget gate. The sub-scripted notations are also marked in the related figures.

The line between $C_{t-1}$ and $C_t$ in figure 2.6 illustrates how the cell state can be effected through the LSTM. The $\oplus$ and $\otimes$ can remove or add information in the current cell state. A gate consists of a point-wise multiplication ($\otimes$) and a sigmoid layer. The sigmoid layer returns values between 0 and 1, which decides how much information to pass through.




(a) Forget gate layer of LSTM

(b) Candidate generation step of LSTM

Figure 2.7: Forget gate layer and candidate generation in LSTM [59]

The **Forget Gate Layer** is highlighted in figure 2.7a. It decides what information to pass on from the previous cell state. It uses $h_{t-1}$ (the output from the previous cell) and $x_t$ (the input to the current cell) to determine how much of each number in the cell state to keep. The resulting vector $f_t$ from the layer consist of numbers between 0 and 1, which is to be multiplied with the previous cell state. The formula for $f_t$ is described in equation 2.10.

$$f_t = \sigma(w_f * [h_{t-1}, x_t] + b_f) \tag{2.10}$$

Figure 2.7b highlights the next steps, which consists of two NN layers. These layers decide what new information to include in the cell state. The tanh-layer generates a candidate cell state $\tilde{C}_t$, while the **Input Gate Layer** (the sigmoid layer) determines how much of the candidate solution to add to the cell state. The output $i_t$ from the input gate layer and the cell state $\tilde{C}_t$ is calculated from equation 2.11 and 2.12.

$$i_t = \sigma(w_i * [h_{t-1}, x_t] + b_i) \tag{2.11}$$

$$\tilde{C}_t = \tanh(w_c * [h_{t-1}, x_t] + b_c) \tag{2.12}$$

The new cell state $C_t$ is calculated with the point-wise operations from the forget gate and input gate, which lead to equation 2.13.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.13}$$

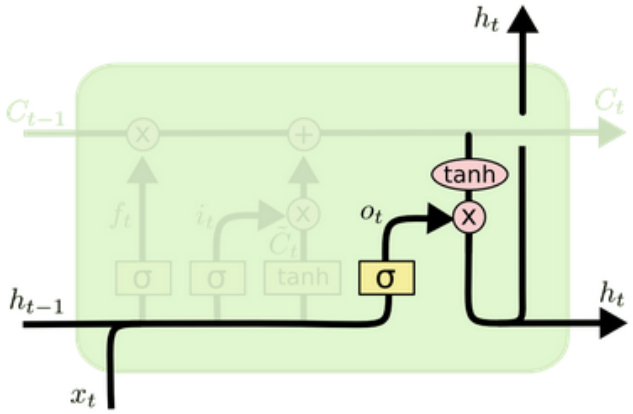



Figure 2.8: Determining the output of a LSTM cell [59]

The final step in a LSTM is to calculate the output, going to the next LSTM layer and/or as an output of the network. The process is highlighted in figure 2.8. In order to determine the output, the values in the current cell state is forced between -1 and 1 by using a tanh-function. Next, is to decide how much of the cell state to output by using the Sigmoid layer. The resulting vector $o_t$ is found by equation 2.14.

$$o_t = \sigma(w_o * [h_{t-1}, x_t] + b_o) \tag{2.14}$$

The final output $h_t$ from the LSTM cell is calculated with equation 2.15.

$$h_t = o_t * \tanh(C_t) \tag{2.15}$$

### 2.4.3 Convolutional neural network

CNNs are a type of DL techniques known for their performance on images. They have been used to classify images, cluster images, identify faces, and much more [17]. Although they are often mentioned for images, they can also be used for 1D-data such as time-series or 3D-data such as videos. A CNN is

an ANN model that uses convolution operations in at least one layer. CNN has become popular due to its ability to automatically extract important features from input data. One of the motivations of using CNN is that it reduces computation requirements due to weight sharing [60].

A typical CNN consist of four types of layers: **convolutional**, **pooling**, **flattening** and **fully connected** [61]. Pooling is often referred to as subsampling. The basic intuition is that the convolution layers work as feature detectors. The pooling tries to preserve the features and often reduce the number of parameters. The flattening maps the potential 2D features into a 1D representation that can be passed to a normal FNN. Often a network can consist of several alternating layers of convolution and pooling. Figure 2.9 shows an example of a CNN architecture containing two convolutional and pooling layers, flattening, and finally a fully-connected network. Each of these layer types is described in the next sub-sections.

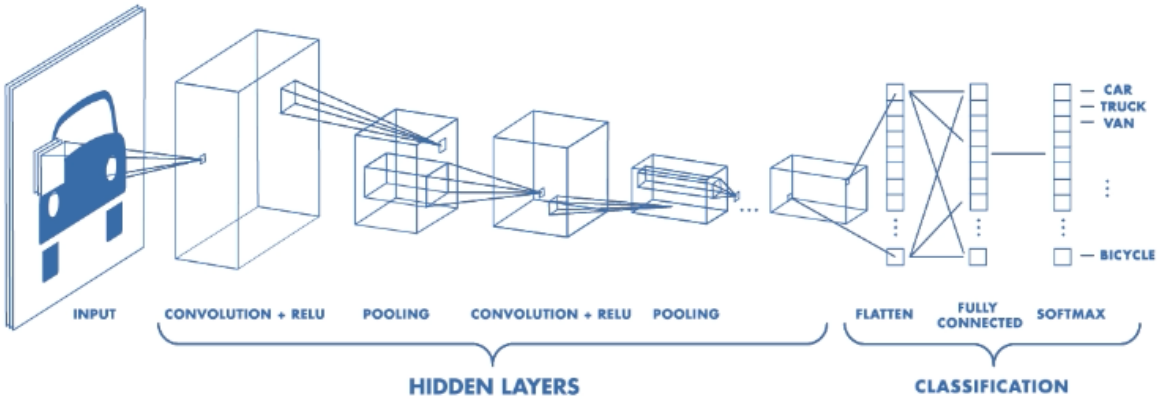



Figure 2.9: Example of CNN architecture with two convolutional and pooling layers [62]

### Convolutional

In mathematics, a convolution operation is an integral that measures how much two functions overlap, as one is passed over the other. It is a way of mixing two functions by multiplying them. In a convolution layer, the input data is convoluted with something referred to as a feature detector or filter, which results in a feature map [61]. During the training phase, the feature detectors learn which features to look for. When the feature detector is convoluted with the input data, it is multiplied with different sections of the images. Figure 2.10 illustrates the convolutional operation between the feature detector $K$ and input data $I$. The mathematical formula for a 2D convolution operation is given in equation 2.16.

$$s(i,j) = (I * K)(i,j) = \sum_i \sum_j I(m,n)K(i-m, j-n) \tag{2.16}$$

where $i$ and $j$ are indexes, $m$ and $n$ are the number of array elements in each dimension, $s$ is the output, $I$ is the input and $K$ is the feature detector. The convolution layer in a CNN is using convolution in a related matter as the equation. The result after a convolution layer is as mentioned referred to as a feature map. The feature map is typically of smaller size than the input data. The height and width of the feature map can be determined with equation 2.17 and 2.18, where $W$ is the input width, $H$ is

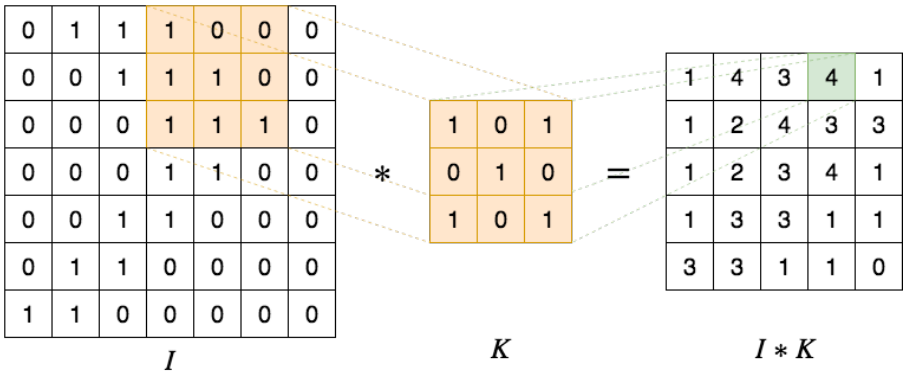



Figure 2.10: Example of the convolution layer operation in a CNN [63]

the input height, $F_w$ is the filter width, $F_h$ is the filter height, $P$ is padding and $S$ is the stride [64].

$$s_h = \frac{H - F_h + 2P}{S} + 1 \tag{2.17}$$

$$s_w = \frac{W - F_w + 2P}{S} + 1 \tag{2.18}$$

The stride is how many pixels in each direction the feature detector is moved for each step. Padding is a strategy that can add borders of, for instance, zeros around the input volume, to avoid or reduce dimensionality reduction. The convolution layer is typically used in combination with ReLU activation functions in order to increase non-linearity in the network. This means the network can detect more complex and non-linear features in the input data. A convolution layer is normally combined with a pooling layer.

### Pooling

The pooling layer is used to reduce the spatial dimensions in a CNN, which leads to a reduced size of the data and fewer parameters [61]. This reduces the chance of over-fitting. The idea of the pooling layer is that it provides spatial invariance, which means that features can be detected even though they are noisy, rotated, squeezed, or somewhat different than normal. Typically, these layers use either max pooling or average pooling. The pooling operation goes through sections of a defined size (pool size) in the input data to calculate the max or average value among the values in the section [65]. Stride can be used to define how far the section is moved for each step. Figure 2.11 gives an example of max pooling with a pool size of (2x2) and a stride of [2,2] is. The stride is given as a vector to indicate both horizontal and vertical stride. The output from pooling is either going into a new convolution layer or flattened before passed to a fully-connected layer.

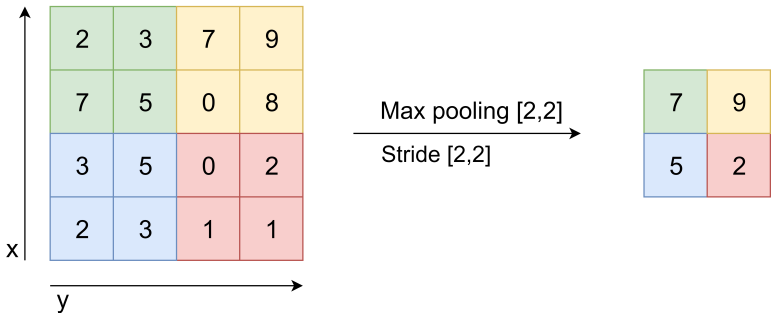



Figure 2.11: Example of max pooling with a 2x2 filter and a stride of [2,2], adopted from [64]

#### Flattening

Flattening is the simple process of taking, for instance, a 2D data matrix and flattening it into a vector [65]. This stage is done in order to get data in an accepted format for a fully-connected layer. If the input to this layer is a 5x5 matrix, the output will be a flat vector of 25 samples.

#### Fully connected

A CNN is typically combined with fully-connected layer(s) to make a classification or regression decision [65]. In general, the convolutional and pooling layers can be combined with any type of neural network (NN) layer as long as the data format is correct. One of the most used approaches is to use flattening and then pass the vector into one or several FNN-layers.

### 2.4.4 Autoencoder

An autoencoder (AE) is an unsupervised approach based on ANN [61]. In general, an AE is a FNN with an input layer, output layer, and one or more hidden layers. It is trained to attempt to copy its inputs to its outputs, through the hidden layer(s). Hidden layers are considered as a bottleneck that forces the network to do dimensionality reduction of the inputs and thus find which characteristics that are important in the data [66]. Compared to traditional methods such as PCA, AE can learn non-linear transformation.

AEs have been applied to solve several types of problems, often around feature extraction, noise reduction, dimensionality reduction, and anomaly detection. Speech enhancement (removing noise) [67], natural language processing [68] and images [69] are examples of problems where AE has been applied. It has also been applied in PHM for feature extraction [70] or in combination with a supervised layer to do fault diagnostics [66]. AEs are often considered to consist of two stages; encoder and decoder. Figure 2.12 shows an example of AE architecture with several hidden layers. The figure shows that the hidden layers are smaller than the input and output layers, which are a typical requirement of an AE. These types of networks are also called under-complete AE. The first half of the network is referred to as the encoder. It aims to compress the inputs $x$ into a representation $h$ of reduced dimensions [61]. The center part of the architecture is often referred to as the code or the bottleneck

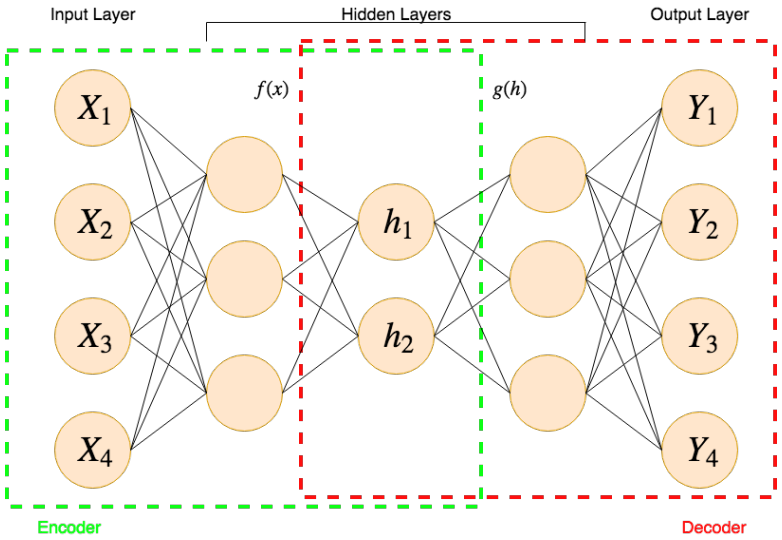



Figure 2.12: Example of AE architecture with hidden layers

and is the representation of the compressed inputs. The second half of the network is typically referred to as the decoder and aims to reconstruct the inputs based on the compressed representation. The AE is designed to not perfectly copy the inputs to the output. Therefore the model is forced to prioritize important characteristics in the data. This means the encoder creates the compressed representation $h$ as shown in equation 2.19, while the decoder tries to reconstruct it as shown in equation 2.20 [66].

$$h = f(x) \tag{2.19}$$

$$y = g(h) \tag{2.20}$$

AEs are considered a special case of FNN and can be trained with similar techniques, such as back-propagation with gradient decent. The learning process is based on minimizing a loss function $L$ such as the one shown in equations 2.21 [12]. The $L$ can for instance be the squared error which penalizes the loss when the output of the network $g(f(x))$ is not equal $x$.

$$L(x, g(f(x))) \tag{2.21}$$

A normal AE have weights $w$ and biases $b$ related to the encoder $e$ and decoder $d$ layer. As with FNN, the training process aims to find the values of the weights and biases. Equation 2.22 and 2.23 shows how the activation from the encoder and decoder layer is found [12]. The activation functions $\sigma_e$ and $\sigma_d$ refers to the encoder and decoder activation function, respectively.

$$f_j(x) = \sigma_e \left( b_j + \sum_i w_{j,i} * x_i \right) \tag{2.22}$$

$$g_i(h) = \sigma_d \left( b_i + \sum_j w_{i,j} * h_j \right) \tag{2.23}$$

Several modern variants of the AE exist. The next sections will briefly explain the ones relevant to this thesis.

### Sparse autoencoder

SAE is a variant of the standard AE which uses an alternative approach which does not require a reduced number of neurons in the hidden layer(s) to provide the bottleneck. Instead, it uses a loss function that penalizes activation's in the hidden layer(s) [71]. The idea is that the network learns encoding and decoding, which relies on a small set of the total neurons, which limits the capacity of memorizing. The penalization in the hidden layer $\Omega(h)$ is added to the loss function. It is typically calculated in one out of two ways [71]. The first is called L1-regularization and penalizes the absolute value of the activation's $a$ in layer $h$, and scales it with a factor $\lambda$. The loss function with the added L1-regularization is given in equation 2.24.

$$L(x, g(f(x))) + \lambda * \sum_i |a_i| \tag{2.24}$$

The second penalization uses something called KL-divergence, which is a way of measuring the difference between two probability distributions. The penalization formula is shown in equation 2.25.

$$\sum_j KL(\rho||\hat{\rho}_j) \tag{2.25}$$

Figure 2.13 shows an example of a potential SAE architecture where it is indicated that only some of the nodes are fired.

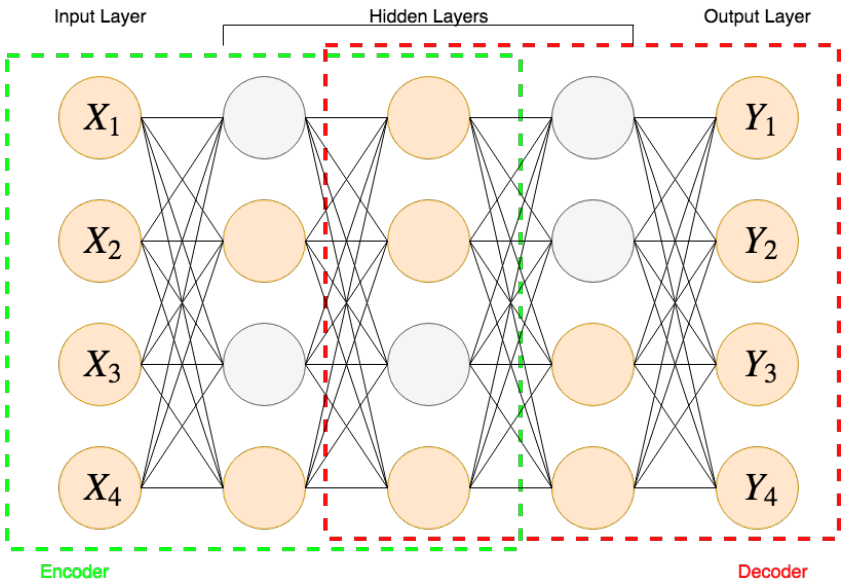



Figure 2.13: SAE-architecture and illustration of none-firing neurons

### Variational autoencoder

The basis of the VAE is that the encoder outputs a probability distribution for each extracted characteristic of the input data, instead of giving each of the characteristics a value [71]. The distribution is assumed to be normally distributed, which means a probability distribution can be represented only by the mean and standard deviation. After the encoding, the input data is represented only as a set of means and standard deviations. A random sample from each of these probability distributions is passed to the decoder model. The decoder model will then try to reconstruct the original input based on these samples. The concept is illustrated in figure 2.14 by having the probability distributions represented by the mean $\mu$ and the standard deviation $\sigma$.

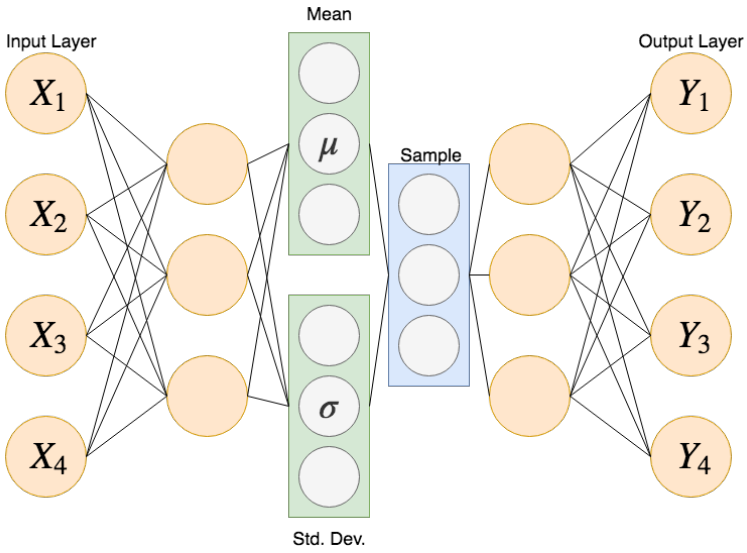



Figure 2.14: VAE-architecture

## 2.4.5 Deep belief network

Deep belief network (DBN) is an unsupervised DL algorithm proposed by Hinton et al. [72] as an alternative to back-propagation. It is considered a generative graphical model composed of multiple layers of hidden units. The layers are connected, but not the units within each of the layers. The architecture of a DBN looks similar to a multi-layer perceptron (MLP), but the building blocks and training process are different. A DBN is a set of stacked restricted Boltzmann machine (RBM). It can be combined with a final layer to do classification or regression, or similar to AE be used for feature extraction. A DBN can be trained to reconstruct its inputs in a probabilistic matter. Its applications can, for instance, be as features detector, later combined with a regression or classification layer [73, 74]. Before further explaining DBN, a quick introduction to RBM is given.

### Restricted Boltzmann Machine

RBMs are often referred to as stochastic neural networks. It is probabilistic graphical models that consist of an input layer referred to as visible nodes, a hidden layer referred to as hidden nodes and a bias unit [72]. The weights related to the nodes are stored in a weight matrix. This makes an undirected graph where the visible nodes represent observable data, while the hidden nodes try to capture dependencies between the observed nodes. The architecture of a general RBM is visualized in figure 2.15. As the figure illustrates, there are connections between all nodes in the two layers, but no connections between nodes in the same layer.

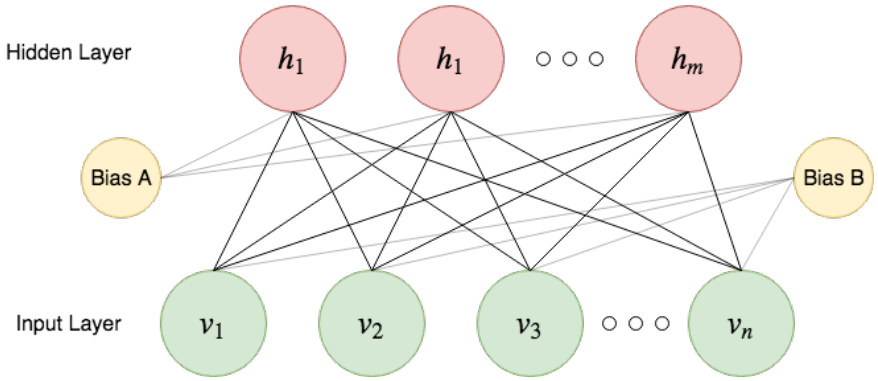



Figure 2.15: RBM with m visible and n hidden nodes (undirected graph)

### Deep belief network

A DBN is as mentioned stacked RBMs. In the network, each hidden layer can be seen as a visible layer for the next hidden layer. In other words, the hidden layer of the first RBM can be seen as the input (visible layer) to the second RBM [72]. It is considered a complex network where typically two types of training are mentioned. The first is the greedy layer-wise training algorithm, which trains the RBMs one by one (it is undirected in this training process). The idea is that the network can learn advanced input-output mappings directly from the data. The learning algorithm pre-trains one layer at the time, while the other layers are kept constant. Each RBM is trained to reconstruct its input through the hidden layer. A layer that is being trained uses the outputs from the previous layer and tries to reconstruct its values. It performs unsupervised learning at every layer to keep important information from the input. The specifics and mathematics of the training process are not important for this thesis, but a thorough description can be found in [75]. The other type of training is referred to as the wake-sleep algorithm, but will not be used in this thesis.

It is important to separate between DBN and deep Boltzmann machines. The main difference between these is that the later has no restrictions on the directionality, while in a DBN all except the top two layers are directed downwards. Figure 2.16 shows an example of the architecture of a DBN consisting of three RBMs.

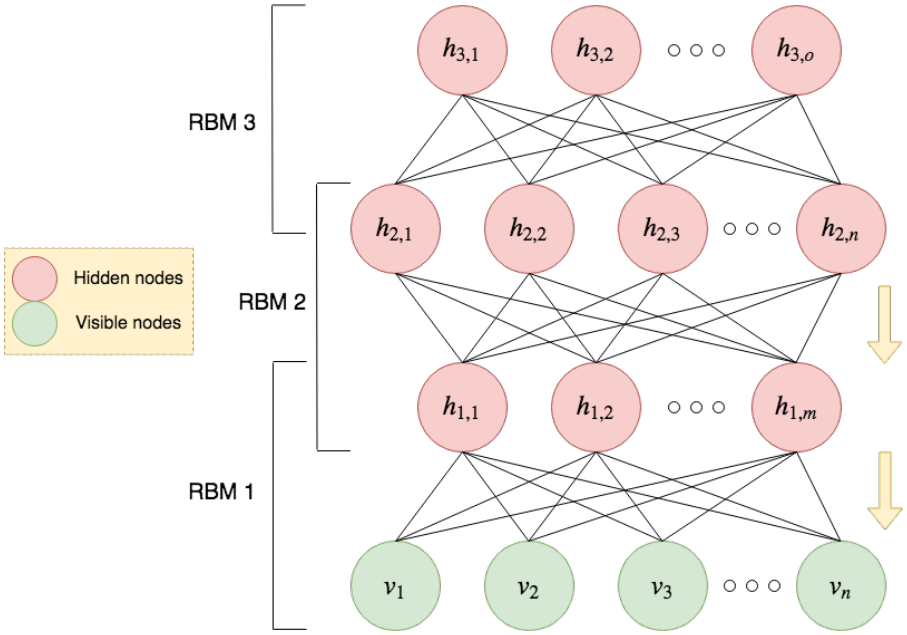



Figure 2.16: DBN architecture consisting of three RBMs.

## 2.5 Particle swarm optimization

Particle swarm optimization (PSO) is a robust stochastic optimization technique based on swarm behaviour [76]. It was originally introduced to optimize continuous, non-linear functions, but other versions of the algorithm can also solve binary and permutation problems. In the topic of ML, PSO has among other things been used as a training algorithm for the weights in a NN [77] and for optimization of ML hyper-parameters [78, 79]. In this thesis, PSO will be used for tuning hyper-parameters.

The basic idea of the PSO is based on a group of individuals (particles) collaborating to improve both the collective and individual performance. The PSO is based on a swarm of particles that aims to find $x$ that minimizes an objective function $f(x)$ [80]. First, the swarm of particles with size $N$ is initialized where each particle is a assigned a random position, $X$, in the search space with the velocity $V$. Internal parameter settings such as the maximum number of iterations and weights are also assigned. At every iteration, the position of each particle is evaluated based on the objective function. Each particle keeps track of their personal best solution throughout a search. The algorithm keeps track of the global best solution among all particles as well. The next step of the PSO is to update the velocity and position of each particle. The velocity is determined based on three factors: inertia, personal influence, and influence by the society. Inertia is considered a way to keep the momentum and is found with equation 2.26, where $w$ is the inertia weight and $v_i(t)$ is the previous velocity.

$$Inertia = w * v_i(t) \tag{2.26}$$

The personal influence lets a particle $i$ to move towards its personal best solution so far. It is determined by equation 2.27, where $c_1$ is an acceleration coefficient, $r_1$ is a random number between 0 and

1, $p_i$ is the personal best solution and $x(t)$ is the current position.

$$\textit{Personal Influence} = c_1 * r_1(p_i - x_i(t)) \tag{2.27}$$

The influence by the society lets a particle move in the direction of the global best solution. It is found from equation 2.28, where $c_2$ is an acceleration coefficient, $r_2$ is a random number between 0 and 1, $p_g$ is the global best solution and $x(t)$ is the current position.

$$\textit{Global Influence} = c_2 * r_2(p_g - x_i(t)) \tag{2.28}$$

These three factors updates the velocity of a particle with equation 2.29, where $w$, $c_1$ and $c_2$ are parameters used to adjust the behaviour of the algorithm. A large $w$ (inertia weight) gives the algorithm better exploring abilities, while smaller values mean better exploiting capabilities. Exploitation can also be achieved by having larger $c_1$ than $c_2$. The opposite gives better exploration ability.

$$v_i(t+1) = w * v_i(t) + c_1 * r_1(p_i - x_i(t)) + c_2 * r_2(p_g - x_i(t)) \tag{2.29}$$

Finally, the position $x$ of each particle is updated with equation 2.30.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.30}$$

After updating the position of all particles, the search loop either continues or stops if the search criterion's is met. Such search criteria can be maximum number of iterations or no improvements for $k$ iterations [80]. The flowchart of the algorithm is showed in figure 2.17.

Established advantages of the PSO is its tendency to converge quicker than for instance genetic algorithm (GA). It also has few parameters to tune and is computationally inexpensive [80]. On the other hand, it can quickly get stuck in a local optimum. It was chosen for hyper-parameter optimization in this thesis due to its tendency for faster convergence. The next chapter explores related work for this thesis.

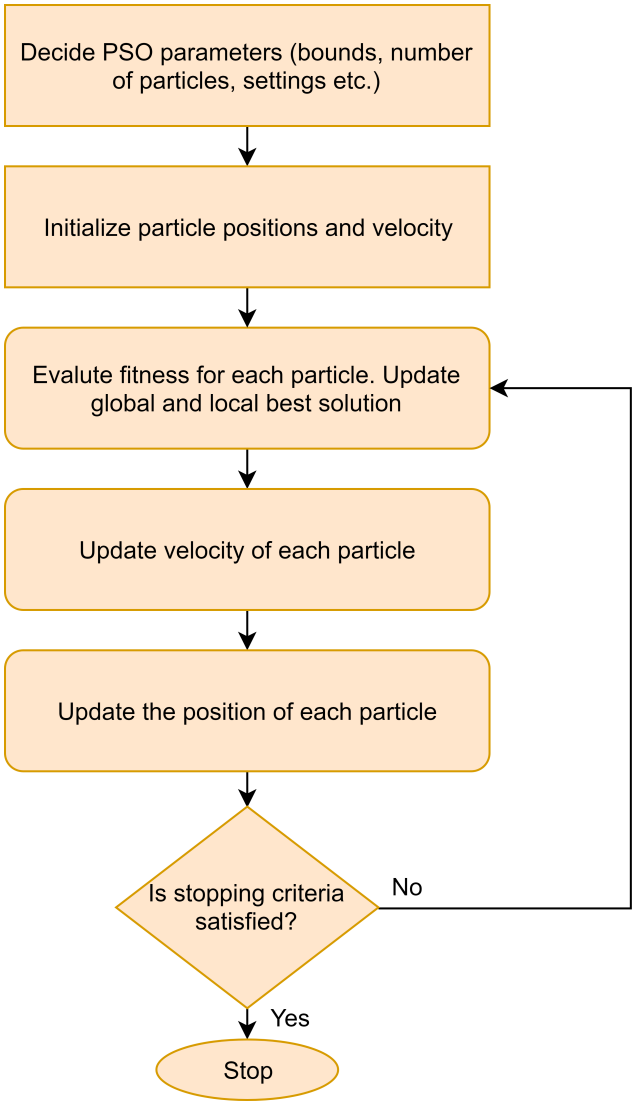



Figure 2.17: Flowchart of the PSO algorithm

# Chapter 3

# Related work

In this chapter, research related to PHM are presented. The chapter starts with an introduction to PHM with traditional methods and continues with research related to modern approaches for anomaly detection, diagnostics and prognostics. The chapter ends with giving insight into how maintenance on air compressors is done today.

PHM, CBM and PdM are related terms and popular research topics. Research has been done on several problems, such as milling machines [81], bearings [82], batteries [83] and gas turbines [84]. IEEE International Conference on Prognostics and Health Management is a relevant conference for the popular topic of PHM. Yearly they arrange an open, accessible competition within the field of prognostics and health management [85]. These competitions lead to a lot of development in the field. This thesis focuses on modern approaches with DL, but much research has also been done with traditional methods such as model-based approaches and shallow ML techniques.

## 3.1 PHM with traditional methods

In 2010, Peng et al. [86] did a review on research related to prognostics and CBM. The review describes the use of both model-based and data-driven approaches. Model-based approaches have been used for both prognostics and diagnostics. Billington et al. [87] proposed an approach based on mechanical modelling of defect propagation to estimate RUL on bearings. Oppenheimer et al. [88] proposed another model-based approach that focused on prognostics and diagnostics of cracked rotor shafts using observers and life models. Byington and Stoelting [89] used a model-based approach to predict RUL and distinguish between failures on flight control actuators. Peng et al. [86] state that some of the limitations of model-based methods are the difficulty of making them and their lack of flexibility. It can be resourceful to develop accurate mathematical models.

Another type of approaches used is knowledge-based systems [86]. These can be considered a type of model-based system, but relates to rules and not mathematical modelling. Butler [90] proposed a framework for PdM based on expert systems. The method detected incipient failures. Another approach with expert systems did prognostics and diagnostics on energy conversion processes [91].

Fuzzy systems is another type of knowledge-based approaches used for PHM [92, 93]. The knowledge-based methods have shown to be exhaustive to maintain, and one of the main problems is the difficulty with converting domain knowledge into rules [86].

Peng et al. [86] also discusses several data-driven approaches to PHM before 2010. The discussed approaches are divided into statistical and artificial intelligence (AI) methods. The statistical methods include multivariate statistical methods, state space models, hidden Markov models, Bayesian Networks, and much more. In 1996, Ray and Tangirala [94] used a statistical approach with non-linear stochastic models to evaluate fatigue crack dynamics in mechanical structures. Another example is predicting the RUL and machine running condition using logistics regression and an auto-regression moving average time series model [95]. The methods Peng et al. [86] refer to as AI methods include variants of ANN's such as dynamic wavelet neural network, self-organizing map (SOM) and MLP. As early as 1997, Zhang and Ganesan [96] estimated fault development and RUL of a bearings by using a NN approach based on SOM. Yam et al. [97] used a traditional RNN approach for predicting fault trends on a gearbox. In 2004, Byington et al. [98] did a comprehensive study on both prognostics and diagnostics for aircraft actuator components based on traditional ANN approaches.

So far, a variety of problems and traditional approaches to PHM has been presented. A problem with the more traditional methods is that they are often highly application-dependent. Model-based approaches are often exhaustive to develop. Traditional ML methods can struggle to find complex patterns in large datasets, and often require manual labor such as feature engineering [17, 12]. This thesis focuses on PHM based on DL techniques. In recent years, research on PHM with these techniques has received much attention. The next section gives an overview of related work with such approaches.

## 3.2 PHM with deep learning

In recent years, much of the work on PHM is related to ML and DL. Ellefsen et al. [12] have done a comprehensive study on the use of DL for PHM in the maritime industry. They discuss how DL gives flexible solution that can work with noisy real-world data, create less application-dependent solutions and how they can work with fewer run-to-failure examples. They also review the use of four well-established DL methods for PHM [12]. These are LSTM, CNN, AE, and DBN. Other DL methods used for PHM are deep generative models [99], deep Boltzmann machine [100] and other variations of deep NN, such as the classic FNN [100]. In this thesis, research related to anomaly detection, diagnostics and prognostics are explored. The next sections goes through these topics one by one. Some research is relevant for several categories, but are in such cases added into one of them.

### 3.2.1 Anomaly detection

Anomaly detection is related to diagnostics but in this thesis, it is treated as two different approaches. Often anomaly detection differs from traditional fault identification based on if labels are available or not. Several different DL techniques have been explored towards anomaly detection. Park et al. [101] used a LSTM-based VAE for detecting anomalies in a robot-assisted feeding system. The approach received higher accuracy than other methods such as one-class support vector machine

(OSVM) and AE. In 2015, Malhotra et al. [102] proposed a method for detecting anomalies in time-series based on stacked LSTM networks. The network is trained on normal data (without anomalies), and predictions error are evaluated based on Gaussian distribution to find the probability of abnormal behaviour. Malhotra et al. [103] used a reconstruction-based anomaly detection approach with LSTM to detect anomalies in multi-sensor data. The model was trained only with normal data, and the results indicated that it was successfully able to detect anomalies in several datasets.

Several AE-based algorithms have been applied successfully for anomaly detection. Yan and Yu [104] proposed a method based on stacked denoising autoenconder (DAE) combined with a supervised classifier to detect anomalies in gas turbine combustor. Another approach used VAE to do anomaly detection based on reconstructed probabilities [105]. They used these probabilities to classify if samples were anomalies or not. Their results showed that the VAE approach performed better than AE-based and principle component analysis (PCA)-based methods. Anomaly detection with AEs or other encoder-decoder (ED)-architectures is typically done by training on normal data, which leads to good reconstructions of normal data, but large reconstruction errors on abnormal data. In 2019, Ellefsen et al. [106] proposed an unsupervised reconstruction-based algorithm for detecting faults in maritime components. Their approach is based on ML algorithms with ED-architecture. The results proved that the maximum acceleration in the reconstruction error can be used to detect faults without having labels. Their method will be further explored in this thesis (section 4.4.1).

DBN has also been used for anomaly detection. Wulsin et al. [107] proposed the use of DBN to measure anomaly score on clinical electroencephalography (EEG) images from humans. Their approach showed that DBN can be used effectively to measure anomalies. Zenati et al. [108] proved that generative adversarial network (GAN) also can be used effectively to detect anomalies in high-dimensional data. They tested the model on a dataset with handwritten digits and one with network intrusions. Li et al. [13] also used GAN for anomaly detection, but on multivariate time series data. In order to capture potential anomalies across time-steps, the proposed method used LSTM as a GAN. Lim et al. [109] highlighted the problem of having few examples of anomalies in a dataset and proposed a method based the GAN called adversarial autoencoder (AAE) for data augmentation. The idea is that the model will create a low-dimensional latent distribution, from where samples can be drawn. Instead of drawing samples based on the probability, which will create a lot of normal data, the samples are systematically drawn from the tail of the probability distribution; hence, generating anomalies.

So far, much work related to detecting anomalies in systems and data is presented. The literature indicates that reconstruction-based approaches are promising in detecting abnormal behaviour. This thesis uses LSTM, CNN, DBN and variants of AEs to detect anomalous behaviour in air compressors. Next, research related to diagnostics is investigated.

### 3.2.2 Diagnostics

While anomaly detection mainly concerns about detecting faults without labels, diagnostics in this thesis is related to fault identification. A lot of different approaches to detect and identify faults in a system have been proposed. Balouji et al. [110] found promising results in classifying voltage dips using a method based on LSTM and FNN. The approach improved the traditional way of detecting

voltage dips, which is based on human expert knowledge and manual labour. In 2018, Xiao et al. [111] researched fault diagnostics on asynchronous motors using LSTM. The results showed that LSTM achieved higher fault diagnosis accuracy than more traditional methods such as support vector machine (SVM), MLP and traditional RNN.

The use of CNN has proven successful for diagnostics and fault detection research. In 2018, Liu et al. [112] proved that it is an effective method for fault detection of hot components on gas turbines. They took advantage of CNN's ability to extract important features and then the predictive power of fully-connected layers. Liu et al. [113] used a multivariate CNN to classify faults on an industrial plant. Their idea revolves around transforming time series signals into an input tensor suitable for CNN. They used a sliding time window and concatenated the individual signals into a 3-dimensional volume. The results indicate that the method outperforms other methods on the same dataset.

Other algorithms such as AEs and DBN has also been used in diagnostics research, but mainly for feature extraction. In 2015, Lu et al. [41] used stacked AE with two hidden layers to do feature extraction for fault diagnostics on rolling bearings. Ma et al. [70] also used AE for finding useful features. Their approach starts with a RBM that combines images and other structure data into one representation. This representation is passed to a stacked AE layer which extracts features before a supervised linear classifier is used for fault diagnostics on power transformers and circuit breakers. Another diagnostics approach based on AE was proposed by Jia et al. [66]. Their goal was to reduce the human labor and feature engineering process involved in diagnostics system to make a more flexible solution. The network consisted of several hidden AE layers with a classification layer at the end. The method got promising results on classifying health condition on machinery.

Zhao et al. [114] proposed a method for doing fault diagnostics on centrifugal pumps. In order to reduce manual labor in the feature extraction phase of vibration signals, they used a stacked DAE to adaptively extract features from the data. The technique resulted in improved fault pattern classification. A similar approach was used by Verma et al. [115], but by using SAE instead. They compared classification accuracy based on different methods and found the approach with SAE for feature extraction to perform the best. In 2018, Liu et al. [116] proposed a method combining stacked AEs to extract features from frequency-domain signals and deep FNN for fine-tuning the weights and making diagnostics on gearboxes. The method showed improved performance on diagnostics compared to traditional methods.

An approach with DBN was proposed by Shao et al. [117] for diagnostics on rolling bearings from vibration signals. They used PSO to optimize the structure and parameters of the network. Ma et al. [118] also presented a DBN approach for diagnostics on bearing degradation. In their research, they used a Weibull distribution to find a health state and several degradation states, while a DBN was used to classify the state of the bearing. Tamilselvan and Wang [74] used DBN for diagnostics purposes as well. They used it for classification and health diagnostics on multi-sensor data. The results showed that the DBN approach generally got better diagnosis performance than SVM, SOM and FNN. Tran et al. [119] proposed a combination of DBN and signal processing techniques to do fault diagnostics on valves on air compressors.

Related work has shown that LSTM, CNN and FNN has been used successfully in different diagnostics problems. These three methods are also tested in this thesis. Other methods such as DBN and AEs

has mainly been used for feature engineering in high dimensional data, which is not relevant for this particular problem. Next, research related to prognostics is explored.

### 3.2.3 Prognostics

Sequential data such as sensor measurements are a typical format of data in prognostics problems [120]. As explained in section 2.4.2, RNN's are designed to work with these kinds of data formats and is therefore considered as suitable for prognostics. Among RNNs; LSTM and GRU are the most used. In general, the vanilla LSTM is indicated to give the best results.

One of the most popular datasets used for research related to RUL predictions is called C-MAPSS [121]. It is a collection of four datasets obtained from simulated degradation on turbofan engines. They consist of nominal and fault of turbofan engines and their degradation over several flights. In 2008, a competition with the goal of predicting the most accurate RUL on a related turbofan engine dataset was arranged by the IEEE Prognostics and Health Management conference. These datasets are often used in prognostics research.

Heimes et al. [122] proposed a method for predicting RUL using traditional RNN trained with back-propagation and extended Kalman Filter training. Their results were accurately able to predict the RUL and therefore received second place in the 2008 PHM competition. Instead of using a pure linear RUL label, they used a piece-wise linear RUL label, which has become accepted as the best labelling approach so far. In this approach, the label is constant until a certain level of degradation is reached, from there it is linearly decreasing. The degradation point is selected to be the same for all sequences. An example of this is shown in section 4.4.3. In 2019, Ellefsen et al. [123] proposed an alternative labelling approach which resulted in one of the best performances on the C-MAPSS dataset so far. The approach is an adaptive version of the piece-wise linear RUL labels. In this approach the starting point of the linear RUL decrements are selected individually for each sequence, based on faults in the system.

Others have also used the C-MAPSS dataset for their research. Wu et al. [42] used LSTM to estimate RUL. In addition, they compared the performance with traditional RNN and GRU. They found that the LSTM performed much better. In 2016, Yuan et al. [124] also used LSTM on turbofan engines, but for both diagnostics and prognostics. They aimed to predict a piece-wise linear RUL label and probability of fault occurrences. The dataset did not contain fault labels, so they used a SVM approach to detect anomalies and use them for labelling faults. Similar to other research, they compared their RUL predictions with other variants of RNN, but found the standard LSTM to perform better. Ellefsen et al. [125] proposed a deep semi-supervised architecture for predicting RUL on turbofan engines (C-MAPSS). The approach used a layer of RBM for weight initialization and feature extraction, together with LSTM and finally a FNN layer for the final prediction. The proposed architecture achieved good results compared to pure supervised approaches. In 2017, Zheng et al. [126] combined sequences of LSTM-layers and normal FNN-layers to estimate RUL on both turbofan engines and milling machines. They state that their approach performs better than traditional methods. Their approach used a piece-wise linear RUL label. According to them, this labelling approach is not general enough and should be explored further.

Malhotra et al. [81] highlighted the problems with assumptions on degradation following a linear or exponential curve. They proposed a method that combined LSTM and encoder-decoder architecture for obtaining an unsupervised health index. The health index is then used to train a regression model that predicts RUL. The approach proves to be promising and achieves better results than several others that make normal degradation assumptions on the same datasets. Hinchi and Tkiouat [127] proposed an approach that combined LSTM and CNN. A convolution layer was used to extract features directly on vibration data from rolling bearings. The features were passed to a LSTM-layer that predicted the RUL on the bearings. The results are promising, but the authors state that further work needs to be done to include uncertainty in the predictions. In 2018, Zhang et al. [128] proposed a method based on LSTM to predict a capacity-oriented RUL on lithium-ion batteries. In order to introduce uncertainties to the predictions, they used a Monte Carlo simulation method.

Having few samples of failure progression is a typical problem in prognostics research that Zhang et al. [129] highlights. They used a Bi-directional LSTM for RUL prediction, but experimented with transfer learning by pre-training the network with a different, but related dataset. Finally, the model is fine-tuned with the exact dataset. The results show that the transfer learning approach in general improved the prediction accuracy on datasets with few samples. The use of transfer learning in prognostics is investigated further in this thesis.

Yoon et al. [99] proposed an approach based on combining VAE and RNN to predict RUL on turbo engines. Their approach used the encoder part of a VAE to reduce the dimensions of the data. Tang et al. [130] used a combination of SAE (for feature extraction) and LSTM to predict bearing degradation performance. The results show better performance than with more traditional methods such as PCA-LSTM, SVM, and FNN. Senanayaka et al. [131] used a similar approach. They used a combination of AE for unsupervised feature extraction and LSTM for prognostics on bearings.

CNN has also been used for predicting RUL. Babu at al. [132] used a deep CNN for estimating RUL on turbofan engines. The network consisted of two stages of convolution and pooling for automatic feature learning, before a fully-connected FNN was used to do the final RUL prediction. The input data consisted of sensor values structured into a 2D-format where each column represented a time-step, while each row was a specific type of sensor measurement. In 2018, Li et al. [27] used a similar approach based on a sliding window to structure the data in a 2D-format. They optimized their solution in terms of the number of convolutional layers and the size of the time window to achieve accurate results.

Deutsch and He [133] proposed a method based on DBN in combination with a FNN for predicting RUL on rotating components. The method tries to utilize the strengths of DBN for feature extraction and FNN for its predicting power. The approach was compared with a model where feature extraction was done with a particle filter-based approach instead of DBN. They achieved quite similar results. Simpler DL techniques such as a FNN with several hidden layers have also been used towards PHM. Tian [134] used age and sensor measurement from present and previous inspections as input to a FNN with two hidden layers. The method was applied to predict RUL in the form of percentage of health state on bearings.

Among the attempted approaches on prognostics, LSTM and CNN seems the most promising. Both of these methods, and FNN is used for prognostics in this thesis. Both the piece-wise linear RUL

labelling approach, and the newly proposed adaptive approach are used for labelling. Next section presents how maintenance on air compressors is done today.

## 3.3 Maintenance on air compressors

PHM has been applied to many industries and problems. This thesis focuses on exploring maintenance towards air compressors, thus, this section investigates state-of-the-art maintenance strategies and solutions for such systems. Not much work is published on PHM for air compressors and therefore several suppliers of compressors are also investigated. Most of the published work is concerned around diagnostics.

In 2005, Carnero [135] investigated instrumentation and diagnostics towards a PdM program for screw compressors. The results showed that traditional methods such as lubricant and vibration analysis could facilitate the planer of a PdM program and thus improve a decision support system towards such compressors. Tran et al. [119] proposed a solution based on DBN for fault diagnostics on valves on two-stage reciprocating air compressors. The proposed method collected signals such as vibration, pressure, and current of an induction motor. These signals were processed with different filtering and computational methods before passed to the DBN, which identifies faults in the valves. The method used stochastic gradient descent for fine-tuning of the weights. In 2018, Maurya et al. [136] proposed a method for fusing features with AE to classify between 5 different states of an air compressor: normal condition, damaged inlet valve, leakage, faulty bearing, and piston fault. A stacked AE was used to extract features from two different sources: human manufactured features based on signal processing and automatically inferred features from a stacked AE. The final classification decision was made with SVM. The results showed promising accuracy compared with other feature fusion methods. Ma et al. [137] also proposed a method based on AE on compressors. They used a VAE for early warning of faults on a four-cylinder reciprocating compressor. The VAE was used to compress the feature parameters and then find a model of the statistical distribution.

PHM for air compressors has been explored mainly towards identifying faults. Several air compressor suppliers were investigated to get a better understanding of the current maintenance strategies on such products. Sperre Industri AS [138], Portland Compressor [139], Arizona Pneumatic [140], A&W Compressor & Mechanical Services [141] and Zabatt Power Systems [142] are referring to PM procedures on their compressors. They give instruction on which time intervals different maintenance actions should be done. These actions include checking the oil level, visual inspection, change oil, and eventually change wear parts or service from the supplier. Most of them seem to have a policy where they provide service when something breaks down. This means that most compressor suppliers use a simple PM and CM strategy. Mazanec [143] discussed the importance of maintenance on air compressors, and which preventive actions should be taken quarterly and annually on different types of compressors and related equipment. Quincy Compressor [144] discusses common mistakes related to air compressors which can be costly and shorten their lifetime. Not considering air energy costs, limited inspections, ignoring air leaks and pressure loss, and fail to remove pipe contamination are mentioned as common mistakes.

IAC [145] is tracking and analyzing vibrations in order to try to reduce costly and unexpected repairs.

Atlas Copco [146] refers to detecting faults early before damage is significant by using diagnostics examination with advanced electronics. Industrial Compressor Solutions [147] also offers some basic monitoring and decision support systems for their compressors. Bacidore [148] discusses the importance of PdM on compressed air systems in plants and refers to manual measurements of both signals and thermal images to take early decisions on about-to-fail equipment.

# Chapter 4

# Methodology

The main objective of this thesis is to explore how DL can be used to improve maintenance on air compressors. So far, the objectives, theory, and related work have been presented. The related works gave insight into state-of-the-art research within PHM and current maintenance strategies on air compressors. The theory described maintenance concepts and relevant DL algorithms. In this chapter, the methodology of the project, the data, and implementation details are described. The chapter ends with an introduction to the experiments in this thesis. The experiments are divided into three different cases that can be useful features in a PHM system. These are anomaly detection, diagnostics, and prognostics. A short description is provided below, while a more thorough description is given in section 4.4. The complete setup and results from the cases are presented in chapter 5, 6 and 7, respectively.

**Anomaly Detection:** Explore different techniques to detect when the air compressor is deviating from normal behaviour and indicate why. The case also contains fault detection in historical data, which can be used for labelling.

**Diagnostics:** Explore different techniques to recognize system faults and predict their severity. Within the field of diagnostics, this is often referred to as fault identification.

**Prognostics:** Explore different techniques for predicting RUL. The main goal of these experiments is to make a system more reliable by being able to predict failures and standstills. The case compares DL techniques and labelling approaches. Transfer learning and uncertainty related to prognostics are also investigated.

## 4.1 Air compressor setup

The experiments in this thesis were performed on data from an air compressor produced by Sperre Industri AS. The test compressor is a two-stage reciprocating compressor. The basic functionality of such an air compressor was described in section 2.1. It is an air-cooled compressor that runs with constant speed. It is connected to an air receiver.

The air compressor is used for experimental purposes. It is equipped with sensors that measure temperatures, pressures, and other useful parameters from both the compressor and the surroundings. The air compressor has been modified to make it easier to force different types of faults. These faults are generated in a much shorter time interval than in realistic cases with deployed compressors. In addition, the compressors are never run until a complete failure (where something breaks), but is stopped within some safety thresholds, referred to as the end-of-life criteria. This is due to the economic and environmental consequences of destroying the compressors. The experiments with this data are therefore considered as a proof of concept. In the future, more realistic data should be collected, but that is outside of the scope of this thesis. Due to the confidentiality requirements (section 1.4) the exact measured values, types of faults, and time horizon cannot be disclosed. The data and faults are discussed more thoroughly in the next section.

## 4.2 Data

The project has used two different data sources. The primary data source was the data collected from an experimental setup (section 4.1) at Sperre Industri AS. A dataset called the PHM08 Challenge dataset was used to research transfer learning in prognostics.

### 4.2.1 Air compressor data

In this project, 25 datasets collected from the air compressors setup described in section 4.1 are available. Five types of sequences are logged: fault type A, B, C, D, and normal data without any faults. The datasets with fault A, B, and C starts with different operational loads and settings with the air compressor being in a normal condition. At a random point in time in each set, the system starts to degrade. The system degrades until the system is out of operating condition and fails. For these three types of faults, the time of which the fault is approximately 33%, 67%, and 100% severity is logged. The system does not fail when a fault has 100% severity, but the effects from the fault will, with time degrade the system until failure. Fault D is not related to any complete system failure, but a fault related to communication problems. It was only used to test the proposed anomaly detection approach. The datasets without faults are sequences where the compressor is running in normal operating conditions. Each dataset consist of 14 sensor measurements and the length of the sequences differs. More details about the sensors and faults cannot be disclosed due to the confidentiality requirements (section 1.4). An overview of the types of sequences is presented in the list below.

- **Normal:** Data with no faults that are collected during varying, but normal running conditions.
- **Fault A:** Data where the compressor starts in normal condition, but a fault is eventually forced. The approximate time of 33%, 67%, and 100% fault progression is logged. After the 100% fault is introduced, the compressor is running until reaching the criteria for end-of-life.
- **Fault B:** Similar procedure as for *fault A*, but with another type of fault.
- **Fault C:** Similar procedure as for *fault A* and *fault B*, but with another fault condition.

- **Fault D:** Does not start in a normal condition, the fault is present in the entire sequence and is as mentioned related to problems with communication. The fault does not force the compressor towards failure.

Table 4.1 describes the available data; how many sets of each type, the length of the sequences and for which cases it is used. Due to the confidentiality requirement the length of the sequences are only given in number of samples. Each case explains the data usage in more detail.

Table 4.1: Description of available data and its usage

| Type | # Datasets | Length of sequence | Case A | Case B | Case C |
|---|---|---|---|---|---|
| Normal | 8 | 77 - 1250 | Yes | Yes | Yes |
| Fault A | 7 | 909 - 3643 | Yes | Yes | Yes |
| Fault B | 7 | 666 - 2554 | Yes | Yes | Yes |
| Fault C | 1 | 1064 - 1064 | Yes | No | No |
| Fault D | 2 | 1228 - 1660 | Yes | No | No |

For fault A, B and C the end of sequence determines the end-of-life of the compressor, which is the time that prognostics experiments tries to predict the time until. Even though fault A, B and C are started in so-called normal conditions, the initial part of the sequence often shows signs of the previous fault. This is due to collecting several sequences with a short time interval between. Hence, the compressor has not been able to regain its normal condition before running for a little while. Fault type C and D are only collected in order to evaluate the anomaly detection.

### 4.2.2 PHM08 challenge data

The 2008 PHM conference competition used a dataset that has several run-to-failure examples for turbofan engines [149]. The goal of the competition was to explore techniques for prognostics and get the most accurate RUL predictions. The dataset is based on an aero-propulsion system simulator called C-MAPSS. The simulator is used to simulate degradation in turbofan engines and collect many sequences where the condition goes from normal condition until failure. Each sequence has different running conditions, initial wear, and noise levels. The data contains 21 sensor measurements and 3 signals referred to as operational settings. The PHM08 challenge dataset is a part of a larger and more complex dataset which is just referred to as the C-MAPSS dataset in the literature. These datasets are the most used within the field of PHM and especially when it comes to RUL predictions. In this thesis, the dataset was used for experimentation of transfer learning in the field of prognostics. The details are explained more thoroughly in chapter 7.

## 4.3 Implementation details

### 4.3.1 Hardware

All experiments are developed on a laptop without a dedicated GPU. Next, the models were trained

and optimized using a better computer. The computer is a Dell Alien-ware desktop computer with 2x NVIDIA RTX 2080 GPU's, 64 GB memory and 16 two-threaded Intel Core i9-9960X CPU cores at 3.1 GHz. The GPU's have 8 GB memory each. The computer is water cooled, and therefore both CPU and GPU are overclocked.

### 4.3.2 Programming language & libraries

All development was done with the programming language Python 3. Python is known to be the most popular programming language towards data science [150]. Many libraries for data analysis and DL are available. Even though Python is considered slow itself, it has great interoperability with low-level C code, which gives great speed-up in well-developed libraries. The experiments and implementations of DL algorithms are mainly implemented with available libraries. This is beneficial since they are optimized for speed. The main libraries that were used:

- **Pandas:** It is a library for working with tabular data. It has several useful functions for data exploration, analysis, and time-series operations [151].
- **Scikit-Learn:** It is often referred to as "sklearn". It is a library with tools for data mining and data analysis [152]. Even though it has many ML model implementation, this package was mainly used for its strengths in data pre-processing and model selection.
- **Keras:** A high-level API library for implementing ML algorithms. Keras runs on top of tensorflow or theano. In this project, keras was used on top of tensorflow.
- **Tensorflow:** Building blocks and API for several different implementations of algorithms and enables combinations of different types of layers. An advantage of using these packages is that they support execution on both CPU and GPU [153].
- **Pytorch:** A open source, deep learning library [154]. It was used as a complement to keras and tensorflow.
- **Pyswarms:** A library for implementation of PSO [155] in Python. It was used for hyper-parameter optimization of DL algorithms.

The next sub-sections describe common implementation details about data formatting, normalization, data splitting, DL models, and hyper-parameter tuning.

### 4.3.3 Data formatting

The available data is stored in *csv*-files. Loading the data directly from these files gives a format where each row contains 14 columns with different sensor measurements, each row represents a new time-step. This format is a two-dimensional matrix with the following shape ($\#Samples, \#Features$). This format is possible to use directly in all models used in this thesis, except LSTM and CNN. These models require extra formatting of the data. LSTM models require not only the current sample, but

all the samples in the specified historical time window. This needs to be in a specific format having a list of lists, with samples. In this format, the first list contains all signals for time-step $t$, the next list holds all signals for $t-1$ and so on. The format can be described as a three-dimensional matrix with the following shape: (*#Samples*, *#Time-steps*, *#Features*). This is specific for the libraries used in this thesis. For CNN the time window approach will be used. This was briefly mentioned in section 3.2.3. It means having each sample structured into a 2D data format where each column is a signal and each row is a time-step. The format has the following shape: (*#Samples*, *#Time-steps*, *#Features*, 1).

### 4.3.4 Normalization

It is easier for ML models to work with data in similar scales. This makes normalization important. Two different normalization methods have been used. The individual experiments describe which methods were used. One alternative is the min-max normalization given in equation 4.1.

$$z = \frac{X - min(x)}{max(x) - min(x)} \tag{4.1}$$

The min-max normalization is not always a suitable choice. Therefore another alternative is to use zero mean unit variance normalization, often referred to as z-score normalization. This normalization is achieved with equation 4.2, where $\mu$ is the feature mean value and $\sigma$ is the corresponding standard deviation of the feature values.

$$z = \frac{x - \mu}{\sigma} \tag{4.2}$$

### 4.3.5 Train, validation and test split

Splitting the data into unique datasets that are specified for training, testing, and validation are important to avoid information leakage. The principle of always testing on unseen data is respected throughout the thesis. The idea is to split the available data into three sets. The training set is used to train the DL model, the validation set is used for tuning hyper-parameters, and the test set is used for the final evaluation. In some cases, a validation set is not necessary; then data is only split into training and testing.

### 4.3.6 K-fold cross-validation

K-fold cross-validation is frequently used to get a fair and less biased evaluation of a model when little data is available. The available data are first split into $k$ splits. Next is an iterative process where one of these splits are used as a test set for evaluating the model. The remaining $k-1$ splits are used for training. As figure 4.1 illustrates, each of the splits is used once for testing. The final performance of a model evaluated with k-fold cross validation is the average of the performance obtained for each of the $k$ folds.

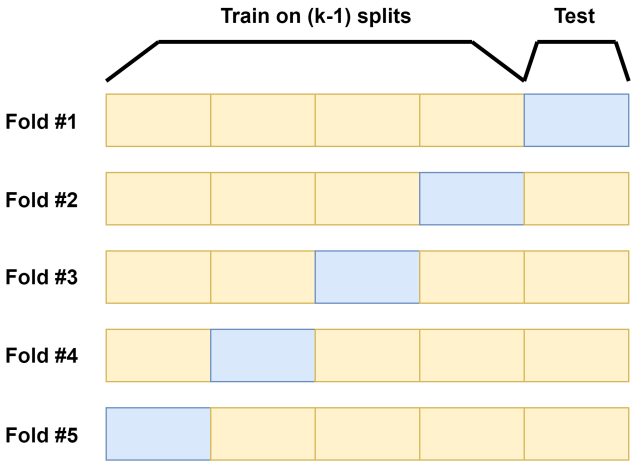



Figure 4.1: K-fold cross validation process with 5 folds

K-fold cross-validation was used for evaluating both diagnostics and prognostics experiments in this thesis. In these experiments, 7 sequences with fault type A, and 7 sequences with fault type B were used. K-fold cross-validation was used with 7 folds, where for each iteration, one sequence with each fault was used for testing.

### 4.3.7 Deep learning implementation

This section briefly mentions which libraries and some details about the implementation of the different DL models.

- **FNN:** Implemented using *Dense* layers in *keras*.
- **AE:** Differs from FNN by its unique architecture, but uses *Dense* layers from the *keras* library.
- **SAE:** Implementation differs from AE by its use of l1-regularization and architecture, but it also uses *Dense* layers in *keras*.
- **VAE:** The VAE is more complex than the traditional AE, and cannot be built with the standard layers in *keras*. A *tensorflow*-based VAE implementation is adopted from [156].
- **DBN:** Implemented with a *pytorch*-based library available on GitHub [157].
- **LSTM:** Implemented with standard layers available in *keras*. The layer called *LSTM* implements a LSTM layer of the vanilla LSTM without peephole connections. The LSTM implementations are in some cases combined with *Dense*-layers for decision making.
- **CNN:** Implemented with standard layers in *keras*. *Conv2D*-layers are convolutional layers, *MaxPooling2D* is a max pooling layer and *Flatten* is used to flatten the input to go into a *Dense*-layer.

### 4.3.8 Hyper-parameter optimization

The hyper-parameters are as explained in section 2.3 a set of parameters that can be tuned in each DL model to try to optimize the performance. Grid search is a common approach for tuning hyper-parameters. This process is often tedious as it requires a DL model to be trained for each of the desired combinations of parameters. A popular approach for tuning the parameters is to use optimization methods such as GA or PSO to avoid searching through the full search-space.

PSO (section 2.5) was used to tune the hyper-parameters for the DL models for both diagnostics and prognostics experiments. The PSO was implemented using the library called *Pyswarms* (section 4.3.2). The library makes it easy to decide on PSO specific parameters, decide how many parameters to tune and the boundaries of each of them. The objective function was based on training the specified DL model with the parameters proposed by the PSO, then returning the achieved score on the validation set.

## 4.4 Cases

This section gives an introduction to what each of the cases investigated and how it was executed. The cases are anomaly detection, diagnostics and prognostics.

### 4.4.1 Case A: Anomaly detection

Anomaly detection is as explained in section 2.3 to detect anomalies in data. In air compressors, it is often collective and contextual anomalies, instead of point anomalies. The main goal of this case is to explore how DL can be used in order to detect anomalous behaviour and give an indication of the current health of the system. Anomaly detection is typically a non-descriptive detection only telling if data is anomalous or not. In this thesis, it is explored towards giving a descriptive range of the anomalies, telling something about how much the system is deviating from normal operation condition. A method for making the results more transparent and descriptive is also proposed. The case also explores offline fault detection. Figure 4.2 shows an overview of the topics in this case.

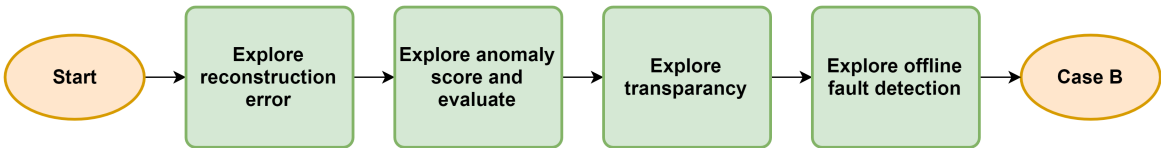



Figure 4.2: Methodology for case study on anomaly detection (case A)

In the literature review (section 3.2) several methods for detecting anomalies were presented, but none of them has been tested on air compressors. The idea of this case was to investigate different types of DL models based on ED-architecture and see their ability to detect anomalous behaviour in air compressors. The following models were investigated:

- Autoencoder (AE), sparse autoencoder (SAE) and variational autoencoder (VAE). See section 2.4.4 for theory.
- Deep belief network (DBN). See section 2.4.5 for theory.
- Long short-term memory (LSTM) with ED-architecture. It is based on the same principle as with an AE, but with LSTM-layers. It tries to reconstruct several time-steps. See section 2.4.2 for LSTM theory.
- Convolutional neural network (CNN) with ED-architecture. Similarly as described for LSTM with ED-architecture, but with CNN-layers. See section 2.4.3 for CNN theory.

Models that use ED-architecture for anomaly detection follow a similar principle. Data in its original dimension is passed through a bottleneck, forcing the data into a lower dimensional representation. The original input is attempted reconstruction based on the compressed representation. The bottleneck forces the model to choose important features that best represents the data. When such models are used for anomaly detection, they are only trained with normal data, resulting in the model providing good reconstructions on normal data, but worse reconstructions on anomalous data. The results refer to a reconstruction error, which basically can be any measure of the error. In this thesis, the absolute value of the difference between the input, $x$, and the reconstructed input, $\hat{x}$, was used (equation 4.3).

$$Reconstruction\ error = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x_i}| \tag{4.3}$$

Each of the stated models was trained using the available data with normal operating condition. This case investigates two features based on the reconstruction error that can be obtained from the trained models. The first is to obtain online insight into the condition of the system and how much it deviates from normal operating condition. The second is offline detection of faults. In this case, online means that it can be used with real-time data and be used live in a real system. Offline means that it is not able to give real-time results. In this case, because it is dependent on "future" data.

**Online: Anomaly score**

Four datasets with fault A and three datasets with fault B are used to investigate and define a method for converting the non-descriptive reconstruction error into an informative scale between 0 and 100. The closer to 0, the more normal. It was investigated if a threshold can be obtained that can decide if data is normal, anomalous or in between. The expected output is an anomaly score that can be used to determine the condition of the system. The proposed method for converting the reconstruction error into a descriptive range is simple and consist of three steps:

1. Scale the reconstruction error into a range suitable for a sigmoid transformation. In the sigmoid function, a -8 gives approximately 0, and 8 gives approximately 1. This first step is a simple scaling from one range to another. The step requires to decide the minimum and maximum values of both the old and new range.

2. This step is a sigmoid transformation with equation 2.4. In this step, the exponent of the $e$ is multiplied with a number between 0 and 1. This value needs to be adjusted to achieve the desired behaviour. It decides the steepness of the transformation.

3. The final step is to transform the values into a scale between 0 and 100. There are no parameters to tune in this stage. The sigmoid transformation gives values in the range between 0 and 1.

The performance of the anomaly detection models is evaluated both visually and based on classification performance. The visual evaluation is important to analyze the behaviour of the models and the approach itself. The desired behaviour is for the anomaly score to keep stable in a normal zone when the air compressor is in normal condition. More importantly, the anomaly score should reach 100 before an air compressor fails. It is therefore expected that the anomaly score for the available sequences starts with a low score, but gradually increases towards 100.

A partly, random selected and labelled dataset are used for testing the classification performance. The evaluation dataset contains 150 normal samples and 200 anomalous samples. The model gets an accepted prediction if it can correctly identify the input. The final performance measures the accuracy on the evaluation set. The evaluation data is selected from unseen data, not used for training or configuring the conversion and thresholds. The 150 normal samples are selected from three sets with fault A (50 samples), four sets of fault B (50 samples) and one set with fault C (50 samples). The anomalies are selected from three sets with fault A (50 samples), four sets of fault B (50 samples), one set of fault C (50 samples) and two sets of fault D (50 samples). The normal samples are randomly selected from parts of the fault sequences before any fault is introduced, while the anomaly samples are taken from parts where the fault has reached 100% severity. A random seed is introduced in order to make the evaluation dataset equal for all the models. Table 4.2 gives an overview of how the available data is used for this case study. Chapter 5 presents the details of the case study and its results.

Table 4.2: Data usage for anomaly detection experiments

| Case A - Anomaly Score: Data usage | | | | |
|---|---|---|---|---|
| Type | # Datasets | Training | Configuration | Testing |
| Normal | 8 | 8 | 0 | 0 |
| Fault A | 7 | 0 | 4 | 3 |
| Fault B | 7 | 0 | 3 | 4 |
| Fault C | 1 | 0 | 0 | 1 |
| Fault D | 2 | 0 | 0 | 2 |

**Offline: Fault detection**

In many cases, logged data is without labels, which can make it difficult to work with. This experiment investigated if the fault time-step can be detected unsupervised based on the approach proposed by Ellefsen et al. [106]. In this case, the fault time-step is where fault A, B, and C reaches 100% severity. The principle is to detect where the acceleration of the reconstruction error is highest and assume it as the point where the fault occurred. The idea is that such an approach can be used for labelling data towards diagnostics or prognostics experiments. It can also be of aid when exploring reasons for

failure in historical data. This is often referred to as root-cause analysis and is based on looking into why something fails.

The approach are briefly explained here, while the full description of the unsupervised reconstruction-based fault detection algorithm is available in [106]. The basis of the method is the obtained reconstruction error. They used mean squared error (MSE) for the reconstruction error, while this thesis uses mean absolute error (MAE). The method uses three sliding time windows that have an adaptive size based on the length of the sequence. The window size is decided with $w = Sequence\ length/35$, which is 2.86% of a sequence. The value of 35 was found by [106] based on trial and error. They state it should be tuned for other applications. The algorithm slides the windows through each time-step in a sequence, with a length of $w$ between each window. Figure 4.3 taken from [106], illustrates the three time-windows and how they are moved through a sequence. Since the reconstruction error often contains noise, the average of the reconstruction error is calculated in each window. Next, the velocity $v$ between window 1-2 and 2-3 are calculated. The velocity is determined by taking the average reconstruction error from window 1 minus the average reconstruction error from window 2. The same procedure is done for window 2 and 3. The next step is to find the acceleration by taking the difference between the two obtained velocities. Finally, the fault time-step can be disclosed by finding where the acceleration is the highest. The time-step might indicate a fault or simply the point where something severe is happening with the degradation.

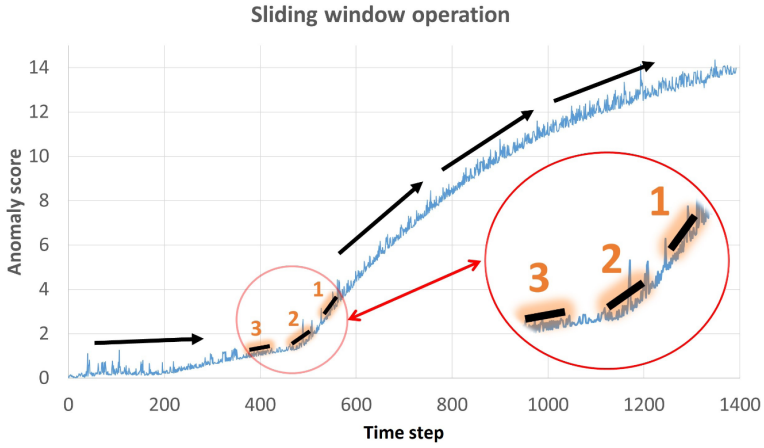



Figure 4.3: The sliding window operation in the reconstruction-based fault detection algorithm [106]

The performance of the algorithm was measured with an accuracy-based scoring function [106]. It is defined in equation 4.4, where $\hat{f}_t$ is the predicted time-step, $f_t$ is the label, and $T_{fdd}$ is the length of the sequence.

$$Accuracy\ (\%) = (1 - \frac{\left|\hat{f}_t - f_t\right|}{T_{fdd}}) * 100 \tag{4.4}$$

In this thesis, the reconstruction error from the six DL models were used. Ellefsen et al. [106] compared AE, VAE and LSTM with a time window of 1. This thesis explores DBN, SAE and CNN in addition. The LSTM is also tested, but with a larger time window. This makes it possible to see which models perform the best for this purpose. The models are retrained by using the normal sequences in addition

to the first 25% of faulty sequences. In order to configure the $w$ parameter, trial and error are done on 7 sequences (4 sequences with fault A and 3 sequences with fault B). When the best $w$ is found, the approach is tested on 8 unseen sequences (3 fault A, 4 fault B and 1 fault C). The scoring method in equation 4.4 was used to evaluate the results.

The results from this case study are presented in chapter 5. The next case study is about diagnostics, and aims to identify faults and their severity.

### 4.4.2 Case B: Diagnostics

Diagnostics is as explained in section 2.2.4, about identifying faults and their severity. While the previous case study looked at detecting anomalous behaviour, this case study focuses on predicting what the exact fault of the air compressor is. This can allow identification of which maintenance actions should be taken. Diagnostics were explored as both a classification (fault identification) and regression problem (predict severity), which is indicated in figure 4.4.

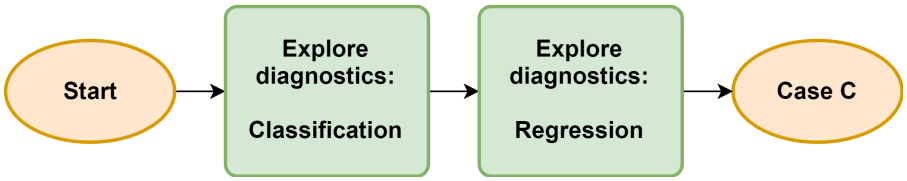



Figure 4.4: Methodology for case study on diagnostics (case B)

This thesis aims to explore diagnostics on air compressors with the following DL models:

- **FNN:** Explore if a standard deep FNN (section 2.4.1) can accurately predict faults in air compressors.
- **LSTM:** Including time into the prediction by using LSTM (section 2.4.2).
- **CNN:** Exploring CNN (section 2.4.3) with the time-window approach.

The diagnostics experiments were done with the data explained in section 4.2.1. Normal data and data that contains fault type A and B were used. Fault C and D was not included since there were too few examples. A set of hyper-parameters of every model was tuned using PSO. The tuning process was done by using 5/7 datasets with fault for training, 1/7 for validation and finally 1/7 for testing. Table 4.3 shows the distribution of datasets used for tuning. After finding the optimized parameters the model performance was evaluated using k-fold cross-validation as described in section 4.3.6 and indicated in table 4.4. Since sequences with fault A and B includes both normal data and faulty data it is sufficient for testing the models.

Table 4.3: Data usage for tuning the models in the diagnostics experiments

| Case B - Diagnostics: Data usage for tuning | | | | |
|---|---|---|---|---|
| Type | # Datasets | Training | Validation | Testing |
| Normal | 8 | 8 | 0 | 0 |
| Fault A | 7 | 5 | 1 | 1 |
| Fault B | 7 | 5 | 1 | 1 |

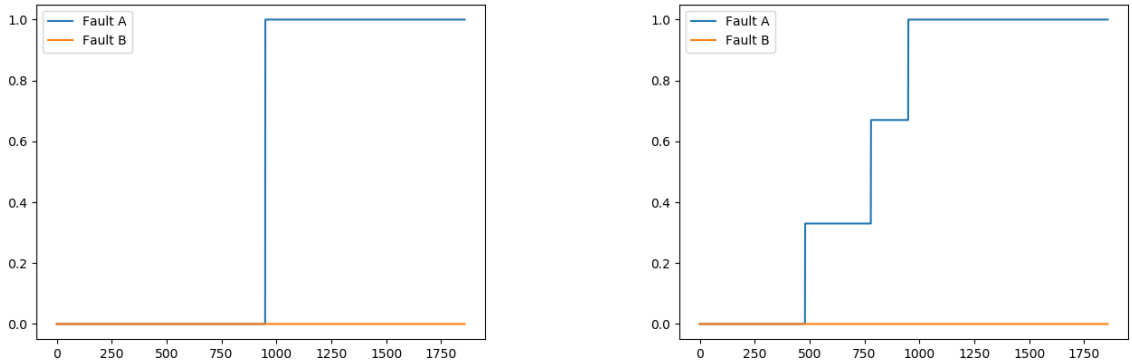
Table 4.4: Data usage for evaluating the models in the diagnostics experiments

| Case B - Diagnostics: Data usage for evaluation | | | |
|---|---|---|---|
| Type | # Datasets | Training | Testing |
| Normal | 8 | 8 | 0 |
| Fault A | 7 | 6* | 1* |
| Fault B | 7 | 6* | 1* |

* Iterative process with k-fold cross validation

### Labelling

The two diagnostics approaches require individual labelling strategies. The first alternative is to label the data only with 0 and 1, where 0 means normal condition, while a 1 means fault (100% severity). There need to be two columns for the label, one for each fault type. The second alternative is to label the data with the severity labels, meaning the problem is seen as a regression problem for predicting how severe the fault is. These labels are in four levels (0, 0.33, 0.67 and 1.0), representing 0%, 33%, 67% and 100% severity, respectively. This labelling also needs two columns, one for each fault. The severity labelling has the potential to give an earlier indication of a progressing fault. Figure 4.5a shows an example of the first labelling approach, where fault B is not present, while fault A shifts from 0 to 1. Figure 4.5b shows an example where fault B is constantly zero, but the fault A label increases from 0 to 0.33, then 0.67 and finally 1.0.




(a) Binary labels for fault identification

(b) Multi-level labels for severity prediction

Figure 4.5: Two different labelling approaches for diagnostics

#### Scoring

As mentioned, the performance of the models were evaluated through k-fold cross-validation. During training, fault identification was optimized using binary cross-entropy as the loss-function. The loss-function for severity prediction was MSE. These losses are not particularly descriptive. Therefore, other scoring methods were used for evaluating the performance. Accuracy was used to evaluate the fault identification, while MAE was used for severity prediction. Chapter 6 presents the case study and its results. The next sub-section explains the prognostics experiments.

### 4.4.3 Case C: Prognostics

Prognostics is as explained in section 2.2.4 about predicting the RUL of a system. In this thesis, it is explored towards predicting the RUL of air compressors with DL techniques. The goal of these experiments is to find a technique able to accurately predict the RUL and therefore be useful to prevent unexpected standstills, and better plan when to do maintenance. Predicting RUL is not something new, but according to the researched literature, it has not been done on air compressors.

This research on prognostics includes several sub-parts. An overview of these topics are listed below and illustrated in figure 4.6.

- Explore three different DL techniques for predicting RUL and test the newly proposed labelling approach referred to as the adaptive piece-wise linear RUL.
- Explore if transfer learning is promising in prognostics. Investigate if a different prognostics dataset can improve the predictions.
- The thesis explores and proposes a new data-driven approach for obtaining uncertainty in RUL predictions.

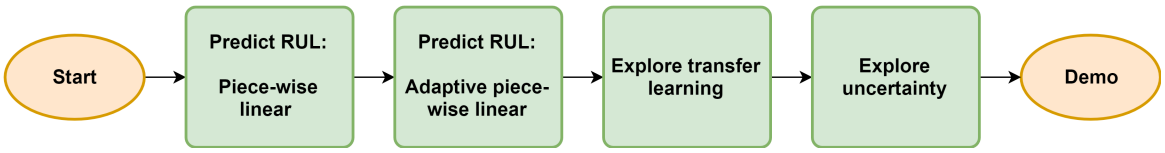



Figure 4.6: Methodology for case study on prognostics (case C)

It is important that predictions are early enough that maintenance activities can be started before it is too late. Predictions should not over-estimate the RUL. Under-estimating is not as bad, but it means that maintenance actions might be taken before it is necessary. The predictions are explored with three different types of DL models:

- **FNN:** Deep FNN is tested as the only model not taking sequences into account.
- **LSTM:** The related work indicated the LSTM as one of the best choices for predicting RUL.
- **CNN:** The CNN is used with the time-window approach described in the related works (section 3.2.3). It has also shown promising towards prognostics.

### Data

In this case, the datasets with fault A and B were used. The normal datasets were not included in this case study. Beyond that, the data usage is equal to what is shown in table 4.3 for hyper-parameter optimization and table 4.4 for evaluation of the best model parameters. These experiments were also evaluated with k-fold cross-validation.

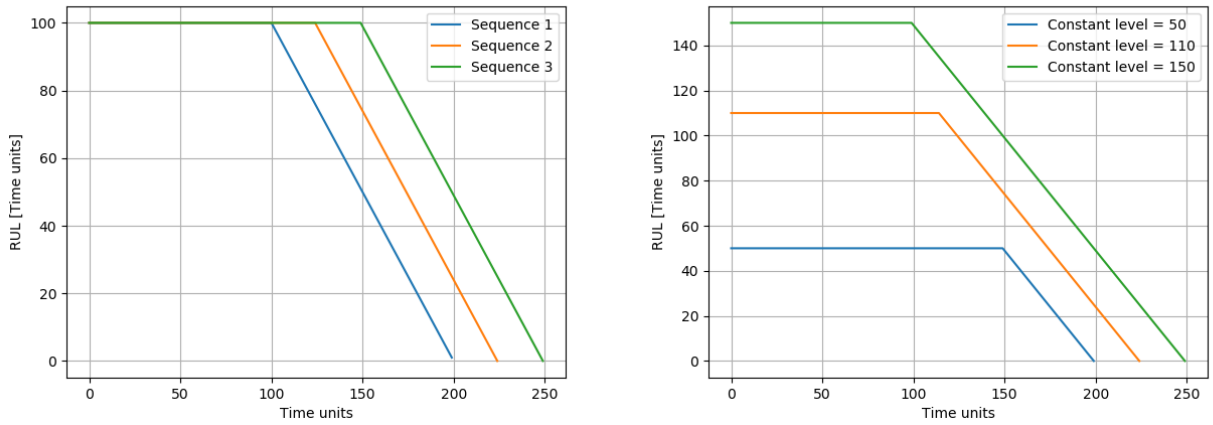
A common problem when working with ML is having few examples for training. Data augmentation is a term used for increasing the data foundation by augmenting existing examples. For images, this can typically be to rotate, skew, flip, and add noise. In this thesis, the data is multivariate time series data, consisting of few run-to-failure examples. Data augmentation was used to create more run-to-failure examples from the initial sequences. If the data is sampled 10 times each second, one run-to-failure example can be split into 10 new run-to-failure examples. This can be done by using the 10 first samples as the first sample in 10 new sequences. Next, every $10^{\text{th}}$ sample is used in each of the new sequences. The first run-to-failure example will then contain sample number 1, 11, 21, 31, and so on. This approach gives more samples for every value of the RUL label, but there is an important restriction to note. The new sequences that originate from the same sequence will, in theory, have values drawn from the same statistical distribution. Therefore, all sequences originating from the same original sequence are used for the same purpose (e.g. training) to avoid information leakage. In the experiments, data augmentation is used to generate 5 new sequences, from each original sequence.

### Labelling

RUL is the number of time units (seconds, hours, cycles, etc.) until a system fails or breaks. As explained in section 4.2.1, the last sample in a sequence is considered as end-of-life. The goal is to predict the time until end-of-life. The available literature has indicated that the piece-wise linear RUL labelling approach is accepted as the best. It emphasizes the fact that systems do not show signs of degradation until a certain level is reached or a fault has occurred. The RUL is decreasing linearly from that point on. Therefore the RUL is kept constant until the last $X$ samples. The constant level must be chosen based on how long in advance predictions should or could be taken.

In addition, the adaptive piece-wise RUL labelling approach proposed by Ellefsen et al. [123] is explored. The approach was able to outperform the traditional labelling approach on the C-MAPPS dataset. Therefore, both of these labelling approaches were explored in this thesis. The adaptive approach were used in combination with the fault labels obtained from the offline reconstruction-based fault detection method. The fault labels was used as the point where the linear label starts. It was only tested on the best performing DL technique on the traditional labels.

Figure 4.7a shows an example of the normal piece-wise linear RUL label, where the constant level was chosen to be 100. It shows three sequences of different lengths (200, 225 and 250). Figure 4.7b shows an example of the adaptive piece-wise linear RUL labels for the same sequences, but with different detection point of degradation. The difference is that this approach takes the time of fault into account, meaning that the constant level will be set individually for each sequence.




(a) Piece-wise linear RUL labelling

(b) Adaptive piece-wise linear RUL labelling

Figure 4.7: Difference between labelling approaches for three different sequences of length 200, 225 and 250

### Scoring

For this thesis, MAE (equation 4.5) and root mean squared error (RMSE) (equation 4.6) were selected to evaluate the performance. MAE gives a descriptive output which says how much the predictions differs from the target in general. In this thesis, it was decided to use a loss-function which penalizes over-estimates more than under-estimates. This is due to a more severe consequence when over-estimating the RUL. The asymmetric absolute error [158] is chosen. The function is quite similar to the ordinary MAE-function. The difference is that when the prediction over-estimates, the absolute value of the error is multiplied with $\beta$. If it under-estimates, the absolute value of the error is multiplied with $\alpha$. If $\beta > \alpha$ over-estimates are penalized more than under-estimates. In this thesis, $\alpha = 1.8$ and $\beta = 2.2$.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} ||t_i - y_i|| \tag{4.5}$$

where $MAE$ is the loss, $N$ is the number of outputs, $t_i$ is the desired output and $y_i$ is the actual output.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2} \tag{4.6}$$

where $RMSE$ is the loss, $N$ is the number of outputs, $t_i$ is the desired output and $y_i$ is the actual output.

The choice of scoring- and loss-function is selected manually based on experience. In the future, a more thorough exploration of loss and scoring for prognostics should be done.

### Transfer learning

Transfer learning was investigated towards trying to improve the predictions, and potentially reduce the need of run-to-failure examples. Transfer learning is much used in object detection in images, where re-using parts of an already trained network can improve predictions and reduce the number of needed training examples. The concept of transfer learning in the field of prognostics has so far received little attention. A part of this case was to investigate the effects of using transfer learning for this purpose. The PHM08 dataset described in section 4.2 was used to build 8 different models that can predict RUL on that dataset. The models are involved in a transfer learning process to try to improve predictions on air compressors.

### Uncertainty

The final part of this case study was to investigate how uncertainty can be used in RUL predictions that are based on DL. Often predictions can be inaccurate, and it can be hard to know if they can be trusted. The goal was to find a method that can give an appropriate uncertainty bound around predictions, leading to more reliability. Often RUL predictions are presented as a single value, which a user may interpret as certain. The prediction is not necessarily accurate, and by using corresponding uncertainty, it is possible to indicate potential deviations based on historical predictions. The proposed method is data-driven and is described in section 7.3. Chapter 7 goes through the case study and results in detail.

# Chapter 5

# Case A: Anomaly detection

This case is as explained in section 4.4.1, a two-part study on anomaly detection, where the main goal is detecting abnormal behaviour in air compressors. The chapter starts with exploring the DL models and how to obtain the reconstruction error. Next, the results from online anomaly detection and offline fault detection are presented.

The six DL models explored in this case study need to be designed to fit this particular problem. The models have several adjustable parameters such as the architecture (number of layers and nodes per layer) and other hyper-parameters like optimizer, activation functions, and learning rate. LSTM and CNN also requires to select a time window to generate the input data format. These parameters were adjusted manually for each algorithm. The process of manual tuning was devoted to finding promising settings for the adjustable parameters. The selection of architecture and parameters was based on finding a model that accurately reconstructs normal data, but obtains a gradually increasing reconstruction error as faults are progressing.

A *tanh* activation function was selected for the output, which means the output of the model are between -1 and 1. Therefore the data was normalized using min-max-scaling (see equation 4.1) between -1 and 1. This forces the model to provide inputs in the right scale. The data normalization was based on the training and configuration sets, and the same transformation was used for the test set. The data used in this case is explained in section 4.4.1. Next, the final architectures and parameters found for each algorithm are presented.

## 5.1 Model architectures & parameters

#### Autoencoder

The architecture and parameters for the AE model were found through trial and error. It is already stated that *tanh* is used as the activation function. Experiments with four types of optimizers; *SGD*, *RMSProp*, *Adam* and *AdaGrad*, indicated that the performance were quite similar, but *RMSProp* was slightly better. A learning rate that gave the desired learning curve was chosen to be 0.0001. The

default weight initialization method called *Xavier* weight initialization was used. Since the problem is hard to optimize with hyper-parameter tuning, several architectures were explored, and the final architectures found is described in table 5.1. The main finding in the architectural choice was that the results were improved when the first hidden layer had increased nodes compared to the input.

Table 5.1: Selected parameters for AE

| Autoencoder - Parameters | |
|---|---|
| Optimizer | RMSProp |
| Learning rate | 0.0001 |
| Input Layer | 14 |
| H1-H5 Layer | 17-8-4-8-17 |
| Output Layer | 14 |

**Sparse autoencoder**

The architecture and parameters for the sparse autoencoder (section 2.4.4) model have equally many layers as the AE, but every layer has 14 nodes. This is due to the sparsity regularization mentioned in section 2.4.4, which penalizes model complexity. This will lead to a similar effect as a normal AE, but the SAE learns which and how many nodes to remove based on forcing weights to zero. The same optimizer, learning rate and weight initialization approach as for the AE was used. The parameters are presented in table 5.2.

Table 5.2: Selected parameters for SAE

| Sparse Autoencoder - Parameters | |
|---|---|
| Optimizer | RMSProp |
| Learning rate | 0.0001 |
| Input Layer | 14 |
| H1-H5 Layer | 14 |
| Output Layer | 14 |

**Variational autoencoder**

The architecture and parameters found for the VAE (section 2.4.4) model is presented in table 5.3. The table shows that the latent space (middle layer) contains information from six dimensions. It also has more nodes in the first hidden layer than in the input layer.

Table 5.3: Selected parameters for VAE

| Variational Autoencoder - Parameters | |
|---|---|
| Optimizer | Mini-batch SGD |
| Learning rate | 0.001 |
| Input Layer | 14 |
| H1-H5 Layer | 16,8,6,8,16 |
| Output Layer | 14 |

**Deep belief network**

The best performing architecture for DBN consists of three stacked RBMs. The input layer has 14 inputs, while the next layers have 16, 13, and 11 nodes, as presented in table 5.4. The $k$ value used was 10, and a learning rate of 0.003. The predictions from the DBN was smoothed using a moving average filter with window size 50.

Table 5.4: Selected parameters for DBN

| Deep Belief Network - Parameters | |
|---|---|
| Learning rate | 0.003 |
| k | 10 |
| Batch size | 25 |
| Input Layer | 14 |
| Hidden Layers | 16,13,11 |

**ED-LSTM**

Similarly to the other models, the best performing parameters were selected based on trial and error. The LSTM does not only reconstruct the sensors for one time-step, but for all time-steps in the selected time window. Table 5.5 shows selected parameters for the LSTM. The table shows that the SGD with a learning rate of 0.001 performed the best with three hidden layers of 10, 7, and 10. The time window was selected to be 20.

Table 5.5: Selected parameters for LSTM

| Long-short-term-memory - Parameters | |
|---|---|
| Optimizer | SGD |
| Learning rate | 0.001 |
| Batch size | 100 |
| Time window | 20 |
| Input Layer | 14 |
| Hidden Layers | 10,7,10 |
| Output Layer | 14 |

#### ED-CNN

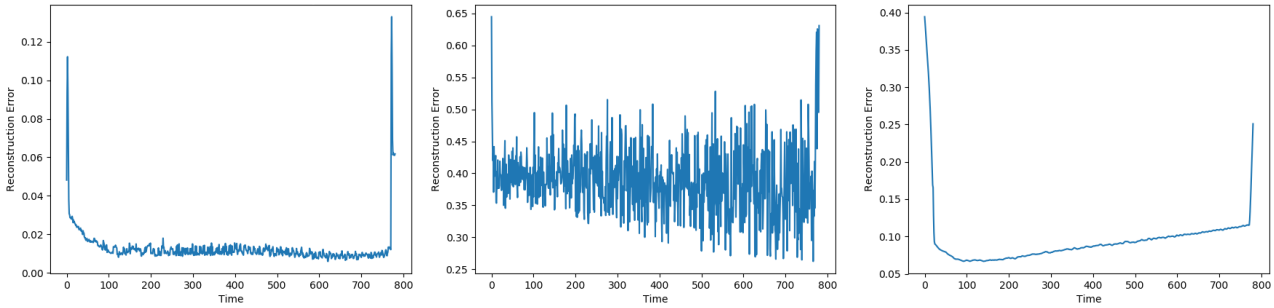
A CNN has many parameters to tune. A time-window of size 20 gave the most promising results. The architecture found consists of combinations of convolutional layers (Conv2D) and max-pooling layers for encoding and convolutional layers and upsampling for decoding. Table 5.6 lists each layer used for the CNN model and its settings. The padding approach called *same* was used for all convolutional layers. This means that the output is padded to reach the same dimension as the input. The middle layers use the ReLU activation function, but the final layer uses tanh to get the data into the correct scale. *Adam* optimizer with the default learning rate of 0.001 performed the best.

Table 5.6: Selected CNN-architecture and parameters

| Convolutional neural network - Parameters | |
|---|---|
| L1: Conv2D | 16 filters and 8x8 kernel. ReLU activation function. |
| L2: Maxpooling | Pool size (2,2) |
| L3: Conv2D | 8 filters and 3x3 kernel. ReLU activation function. |
| L4: Maxpooling | Pool size (2,2) |
| L5: Conv2D | 8 filters and 3x3 kernel. ReLU activation function. |
| L6: Upsampling2D | Size (2,1) |
| L7: Conv2D | 16 filters and 3x3 kernel. ReLU activation function. |
| L8: Upsampling2D | Size (2,2) |
| L9: Conv2D | 1 filter and 8x8 kernel. Tanh activation function. |

## 5.2 Reconstruction error

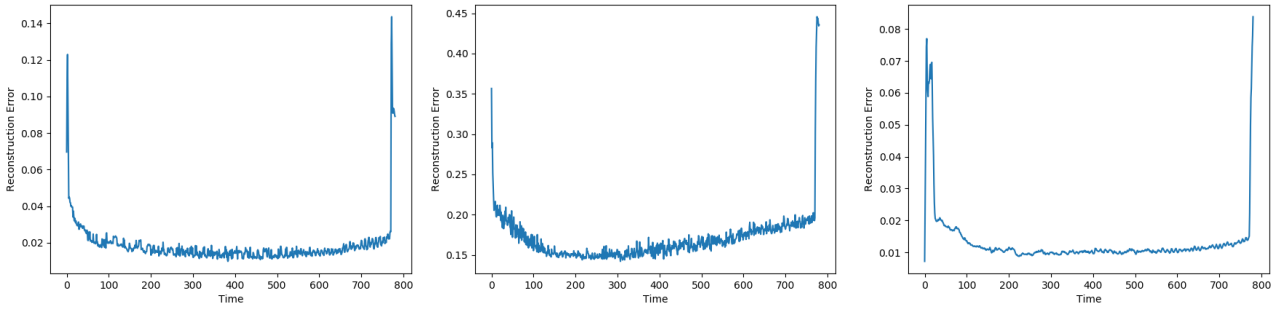
Each model was trained on the available normal sequences. Figure 5.1a, b and c show the reconstruction error obtained on one of the normal sequences with AE, DBN and LSTM, respectively. Each of the reconstructions has a peak in the beginning and end of the sequence. This can be explained by logging starting before the compressor starts, and stopping after the compressor stops. The models achieve quite different results both when it comes to how well the signals are reconstructed and the level of noise. DBN produces a lot of noise and a high reconstruction error compared to the other two models. The AE has a quite low reconstruction error; below 0.02 and some noise. The LSTM model produces even less noise, but with a slightly higher reconstruction error than the AE.




(a) AE (b) DBN (c) LSTM

Figure 5.1: Reconstruction error on sequence with normal data with AE, DBN and LSTM

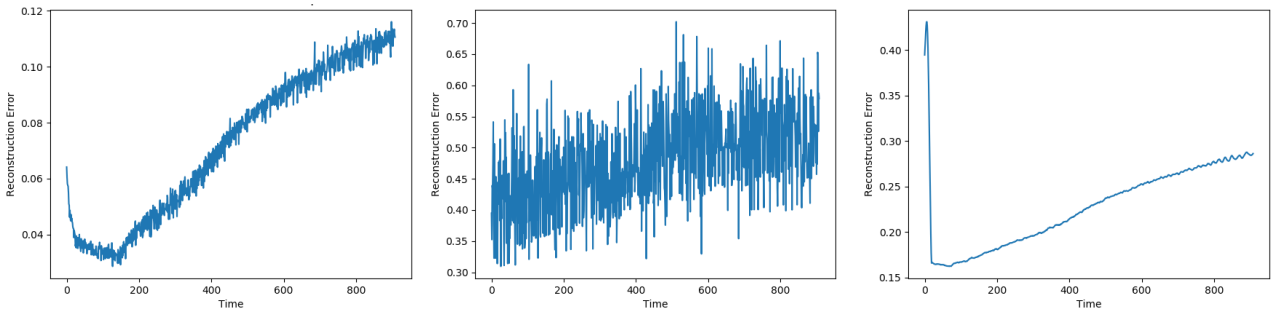
The reconstruction error on normal data from the three other models is shown in figure 5.2a, b and c. The results show similar trends as the previous models. The CNN has less noise than the two others. The SAE and CNN have similar level of reconstruction error as AE, while the VAE has higher.




(a) SAE (b) VAE (c) CNN

Figure 5.2: Reconstruction error on sequence with normal data with SAE, VAE and CNN

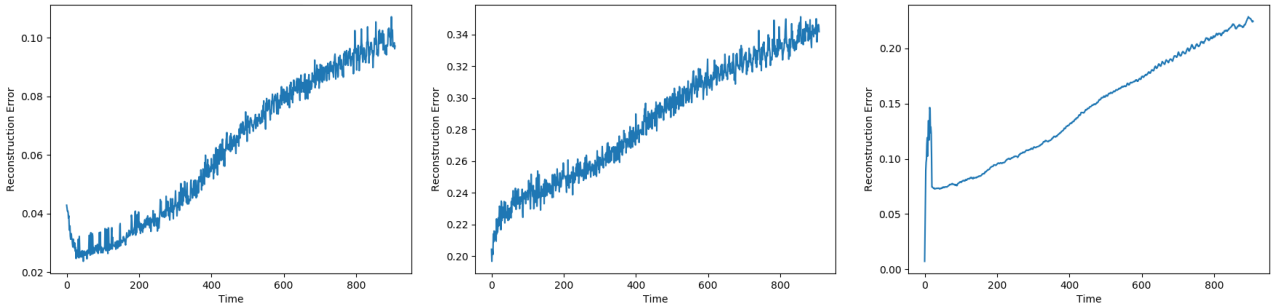
Next the reconstruction errors from the models are evaluated on data with faults. These data sequences start from normal condition, but are gradually introduced for faults until end-of-life. Figure 5.3a, b and c shows the reconstructions from AE, DBN and LSTM on a sequence with fault A. The individual models were affected by similar patterns of noise. The AE and LSTM both start with a low reconstruction error which increases towards the end of the sequence. It is hard to see any clear trend from the DBN due to the high level of noise.




(a) AE (b) DBN (c) LSTM

Figure 5.3: Reconstruction error from AE, DBN and LSTM on sequence with failure due to fault type A

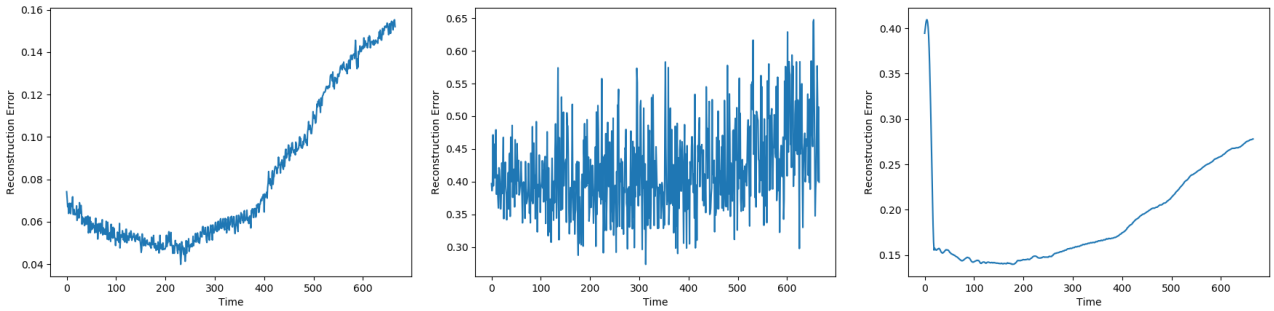
Figure 5.4a, b, and c show the reconstructions from the SAE, VAE and CNN models on the same data sequence. These models were able to start with a low reconstruction error, then gradually increase until failure.




(a) SAE (b) VAE (c) CNN

Figure 5.4: Reconstruction error from SAE, VAE and CNN on sequence with failure due to fault type A

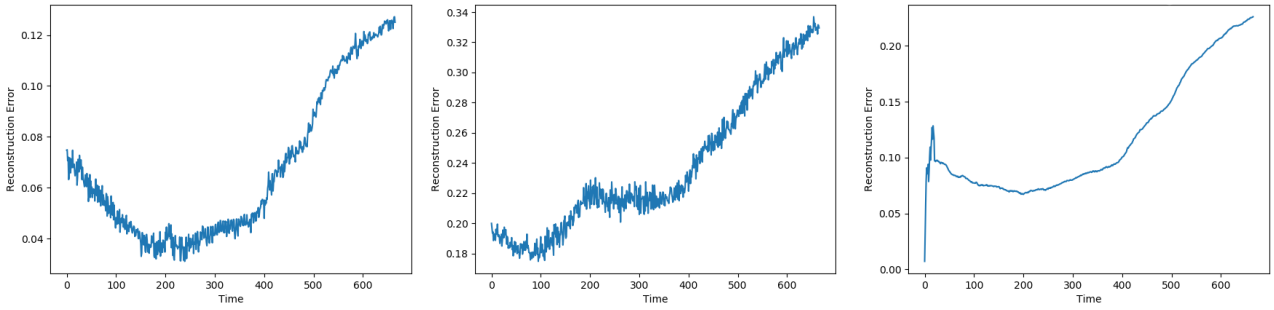
Figure 5.5 indicated that the AE, DBN and LSTM achieve similar results as seen so far. The AE and LSTM went from a low reconstruction error to an increasing one. It was hard to see a clear trend in the results from the DBN due to the noise.




(a) AE (b) DBN (c) LSTM

Figure 5.5: Reconstruction error from AE, DBN and LSTM on sequence with failure due to fault type B

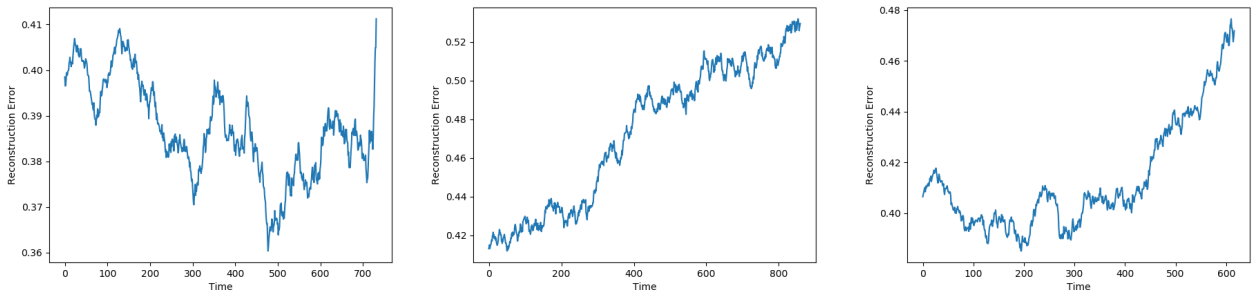
The SAE, VAE and CNN models achieved the desired trend in the reconstruction error. This is illustrated in figure 5.6. The beginning of the sequence seems to lead to an increased reconstruction error, before going down to the normal level. This can be due to the data collection process described in section 4.2.1, where new data sequences were collected before the compressor had regained normal condition.




(a) SAE (b) VAE (c) CNN

Figure 5.6: Reconstruction error from SAE, VAE and CNN on sequence with failure due to fault type B

All the tested models, except DBN, has produced results according to the desired behaviour. A moving average filter is applied to see if the reconstruction error from the DBN model also has an increasing trend. Figure 5.7a, b and c show the reconstruction error on the three sequences seen so far, but with a moving average filter with a time window of 50 units. This reveals that the sequences with faults obtain the desired trend from DBN as well.




(a) Normal sequence (b) Fault A sequence (c) Fault B sequence

Figure 5.7: Reconstruction error with moving average filter obtained from DBN

The results show that all models were able to reconstruct the normal data better than fault sequences, resulting in an increasing reconstruction error close to failure. Each model reproduce the input with different error rate; ranging from the AE with the lowest, and DBN with the highest. In theory, it does not matter if the reconstruction error is high for normal data, as long as it is even higher for failure data. It could be possible to use the reconstruction error directly to indicate if the system is behaving unexpectedly. The scale is not informative, and the only indication of abnormal behaviour is an increasing score. In the next section, it is explored if a more descriptive range of the reconstruction error can be obtained, and if the range can be made more transparent.

## 5.3 Online: Anomaly score

In this section, it is explored how the reconstruction error can be transformed into a more descriptive range between 0 and 100 referred to as an anomaly score. An anomaly score of 100 indicates faulty behaviour, while 0 indicates normal condition. The range is divided into three zones which represent the state of the system. These are normal, warning, and danger zone. The next section describes how the transformation and thresholds are decided.

### 5.3.1 Transformation

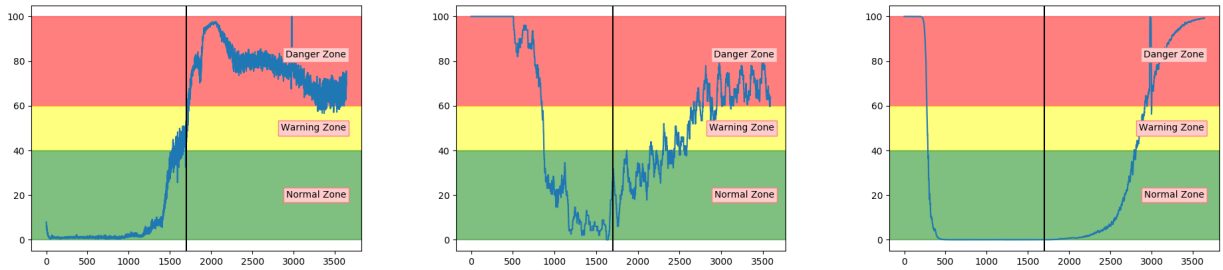
The transformation is tuned individually for each model based on the configuration set. The tune-able parameters are related to the min-max-scaling and sigmoid transformation, as explained in section 4.4.1. There are three phases of the transformation: *(1)* to scale the reconstruction error between two values, *(2)* transform with Sigmoid function and finally *(3)* which is to scale it to between 0 and 100. In step 1, both the old and new minimum and maximum values must be selected. In step 2, the sigmoid exponent must be selected. These values are manually tuned until the desired trend ranging from 0 to 100 is obtained. The outcome of this transformation is referred to as the anomaly score. The parameters that gave the best fit on the configuration set is shown in table 5.7.

Table 5.7: Anomaly score transformation parameters for each model

| | Old min | Old max | New min | New max | Sigmoid exponent |
|---|---|---|---|---|---|
| **AE** | 0.01 | 0.18 | -8 | 9 | 0.8 |
| **SAE** | 0.01 | 0.16 | -8 | 9 | 0.6 |
| **VAE** | 0.205 | 0.32 | -8 | 9 | 0.8 |
| **DBN** | 0.452 | 0.458 | -8 | 9.5 | 0.7 |
| **LSTM** | 0.15 | 0.27 | -9 | 9 | 0.4 |
| **CNN** | 0.05 | 0.20 | -8 | 8 | 0.4 |

The goal of the previous configuration was to get a common scale where the anomaly score indicates the condition of the system. The scale can also be divided into zones. Ideally one normal zone and one danger zone, with a neutral or warning zone in between. Based on the results from the configuration set, the thresholds are decided to be 0.4 and 0.6. This means that when the anomaly score is below 40, it is considered as a normal operating condition, while increasing over 60 something is wrong. These limits are manually selected and need to be adjusted according to the use-case.

Figure 5.8a, b, and c shows the achieved anomaly score from the AE, DBN and LSTM model, on one of the sequences with fault type A in the configuration set. The fault is introduced over three levels of severity. The black vertical line in the figures indicates when the fault has 100% severity. The idea with the anomaly detection is not about predicting that something is about to fail, but give insight into the current condition. Especially fault A is according to the available sensor measurements resulting in long-term changes, not sudden impact. The AE starts with an anomaly score close to zero, before gradually increasing towards a peak, then it decreases slightly. The DBN starts with a high anomaly score. It decreases down to the normal zone before increasing towards the danger zone, reaching it just before it fails. The LSTM model starts similar to the DBN but quickly finds the normal level before gradually increasing after the 100% error is introduced.

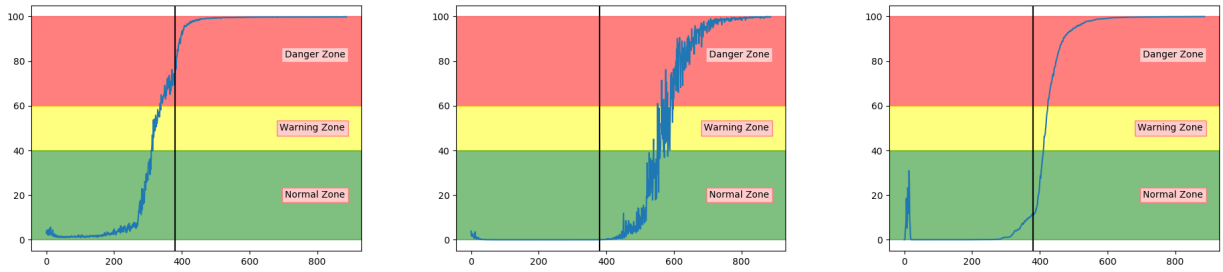



(a) AE (b) DBN (c) LSTM

Figure 5.8: Anomaly score from AE, DBN and LSTM on sequence from configuration set with failure due to fault type A

Figure 5.9a, b, and c show the anomaly score obtained from the SAE, VAE and CNN models on a sequence from the configuration set with fault type B. The results show that the three models start in the normal zone and increases. The main difference between them was that the VAE and CNN

increase towards the maximum score later than the SAE.




(a) SAE (b) VAE (c) CNN

Figure 5.9: Anomaly score from SAE, VAE and CNN on sequence from configuration set with failure due to fault type B

All models except DBN has shown promising results on the sequences from the configuration set. The next step is to evaluate the performance of the anomaly detection on unseen data.

### 5.3.2 Results

The models were evaluated and tested based on unseen data, which, as explained in section 4.4.1 contains four different types of faults. The models were first evaluated based on the accuracy of classifying the 350 test samples into normal or anomalous data. Next, the anomaly score of the complete sequences was visually analyzed and interpreted. Table 5.8 presents the accuracy of each model for all test samples and for each fault type.

Table 5.8: Total accuracy and accuracy per fault type for each DL model

| | **Accuracy** | | | | |
|---|---|---|---|---|---|
| **Model** | **Total** | **Fault A** | **Fault B** | **Fault C** | **Fault D** |
| **AE** | 0.797 | 0.8 | 0.99 | 0.5 | 1.0 |
| **SAE** | 0.783 | 0.81 | 0.93 | 0.5 | 1.0 |
| **VAE** | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| **DBN** | 0.794 | 0.83 | 0.65 | 0.8 | 1.0 |
| **LSTM** | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| **CNN** | 0.963 | 1.0 | 0.87 | 1.0 | 1.0 |

The results showed large differences in the classification performance of the models. Both the VAE and LSTM impressively achieved 100% accuracy. CNN performed quite closely by achieving 96.3% and correctly classifying all samples in the test set with fault type A, C, and D. The three best models were able to perfectly detect all samples within the two new fault types (C and D). The three remaining models (AE, SAE and DBN) achieved between 78 and 80% accuracy. Table 5.9 presents an overview of the number of wrongly classifications and how many of them were present in the warning zone.

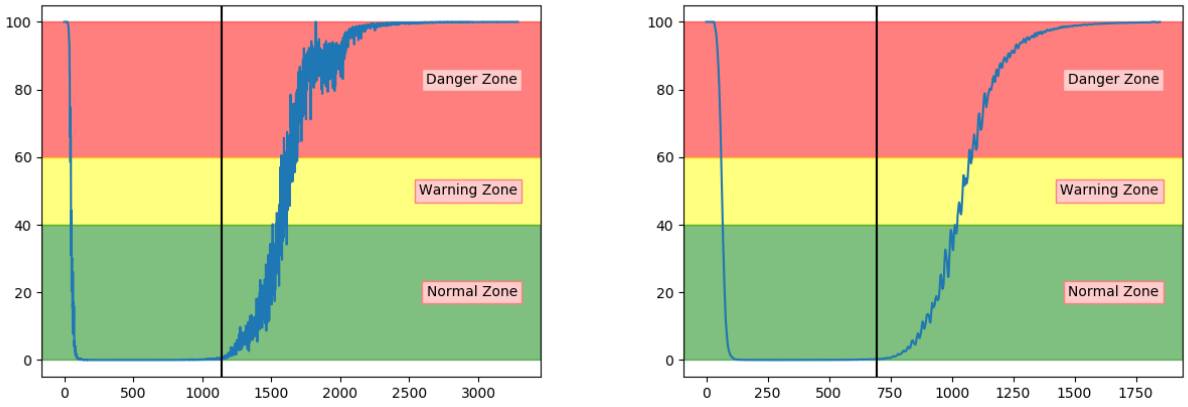
Table 5.9: Miss-classifications in the anomaly detection models

| Model | # Misses | # In void | Share in void |
|---|---|---|---|
| **AE** | 71 | 60 | 0.85 |
| **SAE** | 76 | 58 | 0.76 |
| **DBN** | 72 | 2 | 0.03 |
| **CNN** | 13 | 11 | 0.85 |

The AE, SAE and CNN have most of their miss-classifications in the void, while the DBN has almost every miss-classification in the wrong zone. DBN has performed worse than the other models. There were large differences in the classification performance of the models. While VAE, LSTM and CNN performed with an high accuracy, the other models have several miss-classifications. The anomaly score is not designed or intended to be a pure classifier of normal or failure samples. Therefore the visual analysis is just as important. Next, results from each fault type are analyzed.

### Fault A

Both the VAE and LSTM got 100% accuracy on the test set. Figure 5.10a shows the anomaly score for VAE on one of the unseen sequences with fault type A. Figure 5.17b shows for another sequence, but with LSTM. For both models, the anomaly score started around zero and increased gradually after the 100% fault severity was reached. They were as desired able to detect and indicate an anomaly score of 100 in advance of end-of-life. The results were quite similar for all sequences with fault A for these two models.

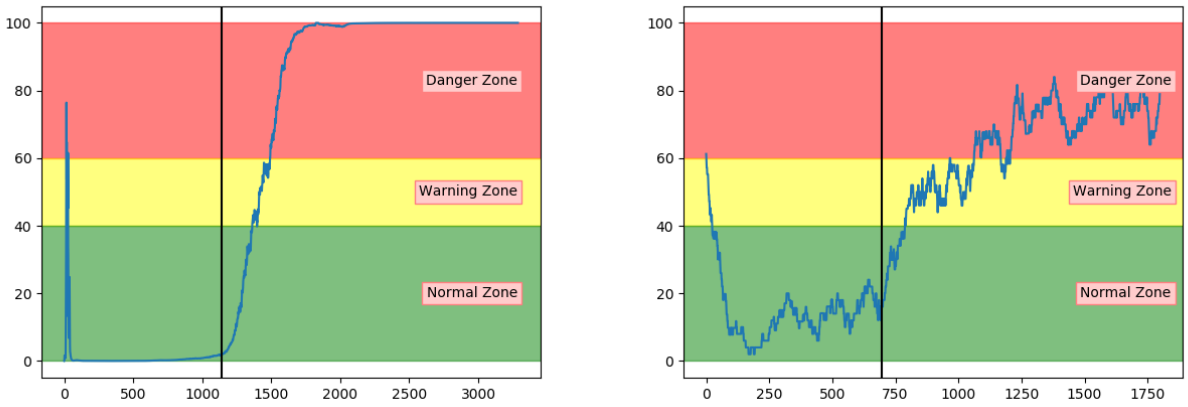



(a) VAE (b) LSTM

Figure 5.10: Anomaly score from VAE and LSTM on unseen sequence with fault type A

Figure 5.11a shows that the CNN performed similarly as VAE and LSTM for fault type A. Figure 5.11b indicate that DBN was less confident in the normal zone, and increased only slightly into the danger zone. It was enough for the model to get several correct classifications, but it provides worse

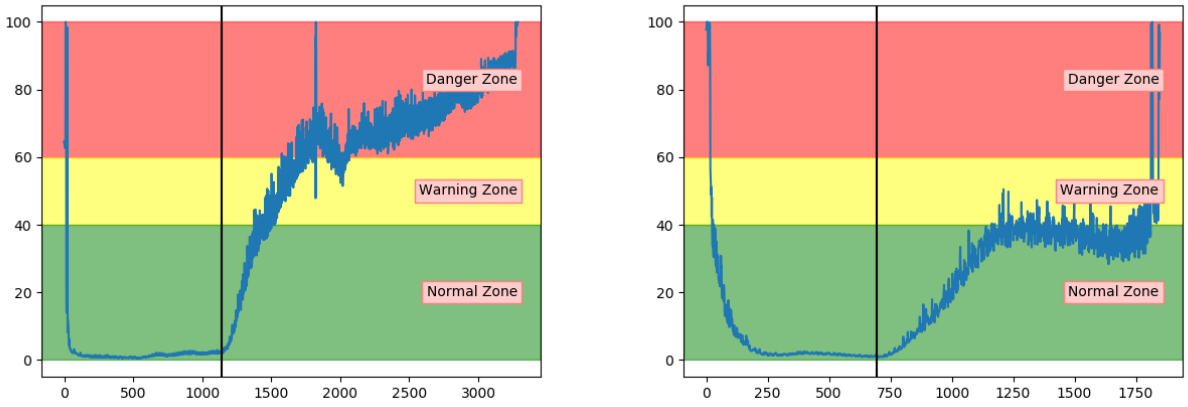
certainty than the other models. It also has more noise.




(a) CNN (b) DBN

Figure 5.11: Anomaly score from CNN and DBN on unseen sequence with fault type A

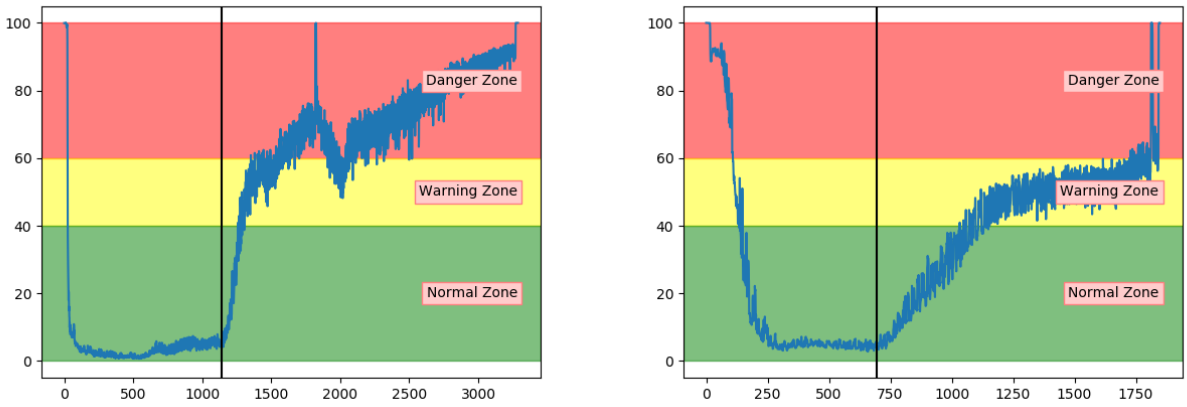
Figure 5.12a and b shows the anomaly score obtained from the AE for two different sets with fault A. While *(a)* shows satisfactory performance, *(b)* shows that the anomaly score was barely able to leave the normal zone. The AE received inconsistent results for sequences with the same fault type.




(a) Split 5 (b) Split 7

Figure 5.12: Anomaly score from AE model on unseen sequences with fault type A

Similarly to the AE, the SAE achieves inconsistent results. This is indicated in figure 5.13. The results on split 7 were better than the AE results, but still not able to properly increase into the danger zone.



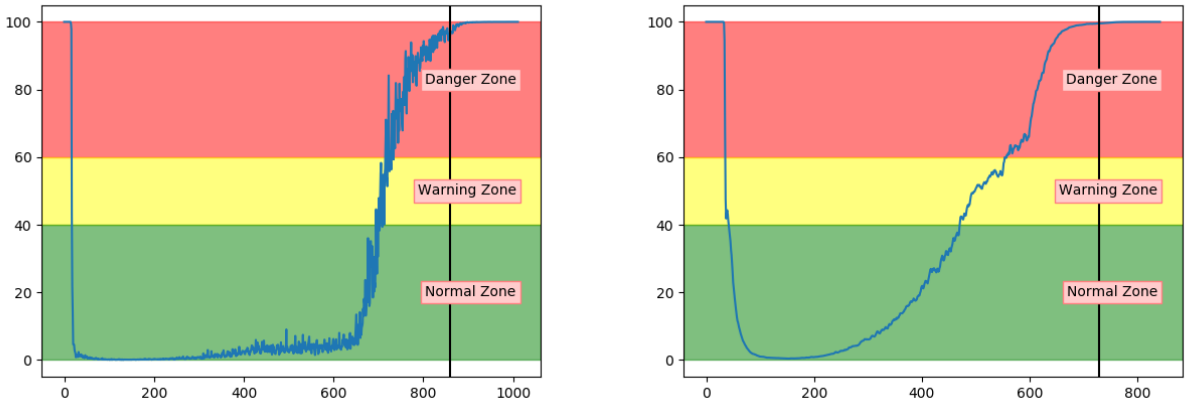

(a) Split 5 (b) Split 7

Figure 5.13: Anomaly score from SAE model on unseen sequence with fault type A

Analysis of the anomaly score on sequences with fault type A has indicated that only VAE, LSTM and CNN achieved the desired results. They were able to indicate anomalous behaviour before reaching end-of-life. The three other models had more fluctuating and inconsistent results.

### Fault B

Sequences with fault B was according to sensor values and domain-experts affecting the air compressor in a higher degree than fault type A. This means that there was a shorter period from the fault reaches 100% severity, until the system fails.

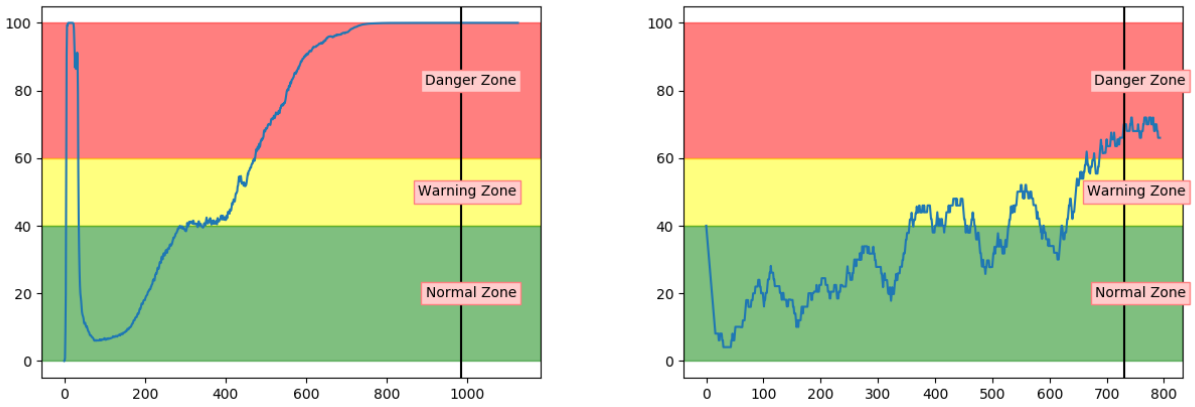



(a) VAE (b) LSTM

Figure 5.14: Anomaly score from VAE and LSTM on unseen sequence with fault type B

Figure 5.14a shows that the anomaly score from VAE reaches 100 before the system fails. Compared to results on fault A the anomaly score increased before the 100% fault severity was reached. This supports the information that fault B has larger effects on the system. Figure 5.14b show similar tendencies from the LSTM model. The main difference was that the anomaly score from the LSTM model started to increase earlier.

The accuracy on the test samples indicated that the CNN model struggled to correctly identify the samples from sequences with fault type B. Figure 5.15a supports these results by showing that the anomaly score was rarely in the normal zone. The anomaly score increases slowly, but starts too early. According to figure 5.15b, the anomaly score from DBN were not able to increase into the danger zone until right before failure. Compared to the other models the anomaly score from DBN lies much higher in the normal zone and much lower in the danger zone.

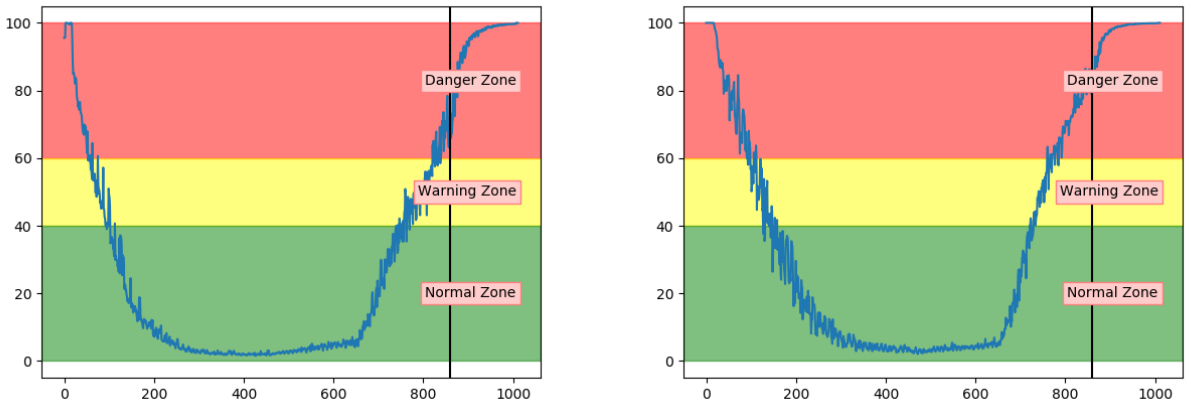



(a) CNN

(b) DBN

Figure 5.15: Anomaly score from CNN and DBN on unseen sequence with fault type B

Figure 5.16a and b show that both the AE and SAE achieved the desired trend of the anomaly score. For both models, it stayed in the normal zone before gradually increasing up to an anomaly score of 100. Compared to the other models, the results from the AE and SAE started with a longer period in the error zone, before reaching the normal zone. This can either be because of how the sequences are logged or a problem with these models.



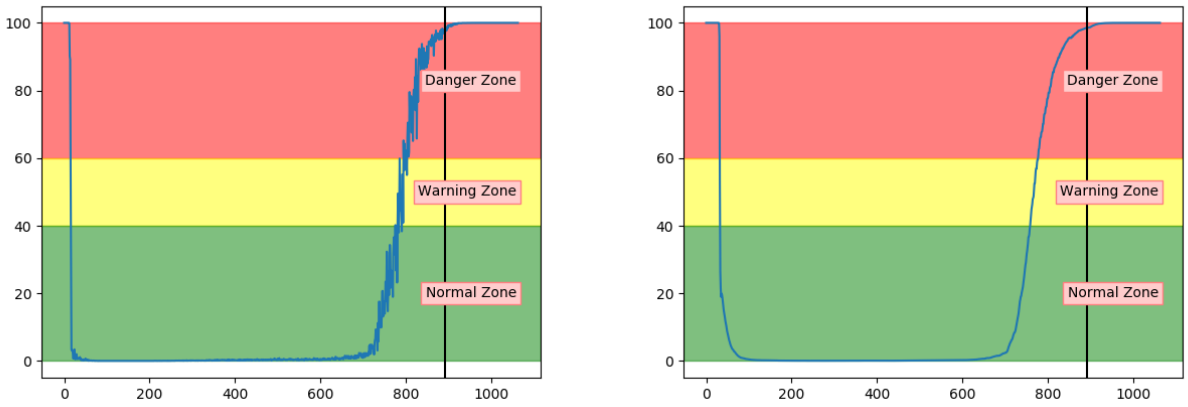

(a) AE (b) SAE

Figure 5.16: Anomaly score from AE and SAE on unseen sequence with fault type B

Results have indicated that the VAE and LSTM was the best performing models on sequences with both fault type A and B. The CNN showed promising performance, but was too early on increasing the anomaly score, especially on fault type B. The DBN was not able to properly increase the anomaly score into the danger zone and had a lot of fluctuation. Both the AE and SAE provided better results for fault B, than A.

### Fault C

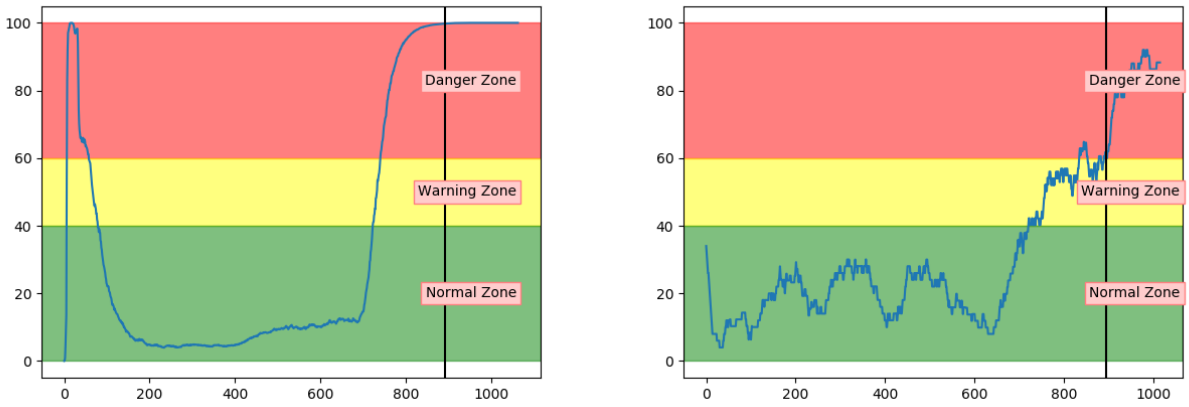
The performance of the models were also analyzed on a new type of fault, not present in either training or configuration. This can show if the algorithm can generalize and detect anomalous behaviour caused by new, unseen types of faults. The two best performing models so far are the VAE and LSTM. Figure 5.17a and b show that both of these models achieve the desired behaviour and can give a clear indication of anomalous behaviour before the system fails.




(a) VAE (b) LSTM

Figure 5.17: Anomaly score from VAE and LSTM on unseen sequence with fault type C

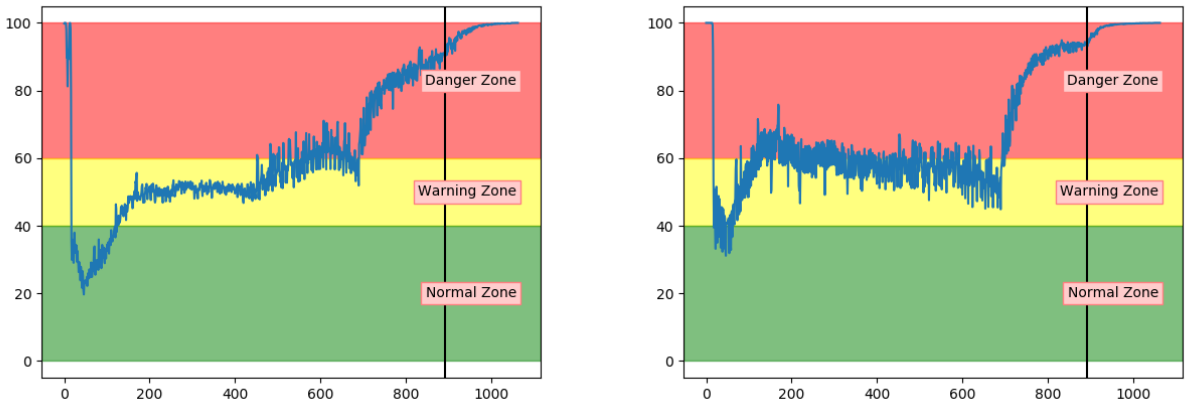
Figure 5.18a shows that the CNN model were able to detect anomalous behaviour on a sequence with fault type C before the system fails. It performs similar to the VAE and LSTM. Compared to previous results, the DBN were able to detect anomalous behaviour on the sequence with fault type C. It has an anomaly score which fluctuates more than for other models, but it stays within the appropriate zones and gives an indication of anomalous behaviour before the system fails.




(a) CNN (b) DBN

Figure 5.18: Anomaly score from CNN and DBN on unseen sequence with fault type C

Figure 5.19 indicates that the AE and SAE were not able to generalize. The anomaly score stayed in the warning zone, instead of the normal zone. They were both successful in detecting an anomaly score at 100 before system failure. They were, on the other hand, unsuccessful in capturing that the system was in the normal zone most of the sequence.

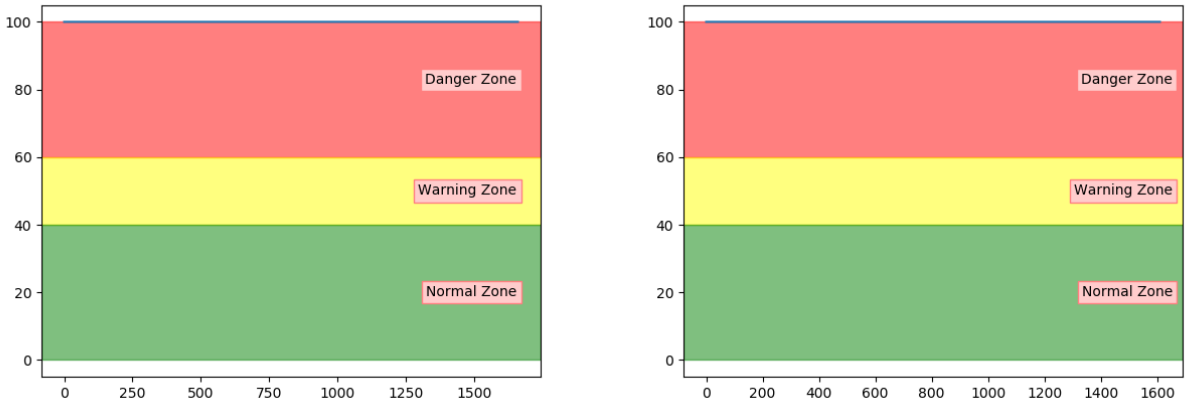



(a) AE (b) SAE

Figure 5.19: Anomaly score from AE and SAE on unseen sequence with fault type C

### Fault D

The models were analyzed based on the performance on sequences with fault type D. Compared to the other fault types, this fault type has the same severity through the entire sequence. The results from the classification of the test samples showed that every model was able to accurately classify between normal and anomalous samples on sequences with fault type D. Visual analysis showed that all models keep an anomaly score of about 100 through the entire sequences. Figure 5.20 proves this by showing the anomaly score on a sequence with fault D with both VAE and DBN. The other models follow the same pattern.




(a) VAE (b) DBN

Figure 5.20: Anomaly score on unseen sequence with fault type D

#### Model evaluation

The results have proved that VAE and LSTM can detect anomalous behaviour in air compressors accurately. Both models showed promising results in classification and visual analysis. CNN did not perform as good as VAE and LSTM. It received a quite high accuracy on the test samples, and visual inspection proved that the CNN is promising. The three remaining models achieved lower accuracy on the test samples and were only able to deliver suitable anomaly scores on some of the fault types, and some of the sequences. The results and model evaluation are discussed further in chapter 8.

Anomaly detection is typically a black box, only indicating if data is anomalous or normal. The proposed method based on anomaly score can indicate how much the behaviour is deviating from normal condition. This increases the transparency compared to typical anomaly detection approaches. In the next section, a method to make the anomaly score more transparent is proposed.

### 5.3.3 Transparency - Error contribution

A low anomaly score shows that the system is acting close to normal behaviour. An increasing anomaly score indicates that something unexpected is happening. When the anomaly score goes towards 100, it can be interpreted as a danger for failure. It indicates that the system is behaving anomalously and might fail. The anomaly score gives no information about what is wrong with the system, only that something is wrong. Historically, anomaly detection is not used for this since its purpose is mainly to discover anomalies, while diagnostics will uncover the actual fault. Diagnostics is dependent on having examples of previous faults. The proposed method for increasing the transparency in the anomaly score can give an alternative which can help to identify faults, without historical examples.

The proposed method for increasing the transparency is to calculate each input's (in this case, each sensor's) contribution to the anomaly score. Since all inputs are normalized between the same range, they can influence the score equally. The MAE is the mean of the absolute error between each pair of input and reconstructed input. Therefore, by not taking the mean of the errors, the error of each input in the sample is known. The contribution can then be found by dividing the error of one sensor, by the total error. Equation 5.1 shows how the contribution of one input, from one sample, can be calculated. In the equation $i$ indicates which sample and $j$ indicates which input (sensor).

$$\text{Contribution Input}_{i,j} = 100 * \frac{\left| Input_{i,j} - \hat{Input}_{i,j} \right|}{Reconstruction\ Error_i} \tag{5.1}$$

The proposed method adds information to the anomaly score that indicates which sensors in the signal, contributes the most to the reconstruction error. The idea is that if a domain expert or service personnel see the top deviating sensors, the potential faults can be recognized. The method works for all the proposed models. Next, some examples of the anomaly score and the belonging contribution are showed. Due to the confidentiality requirements, the sensors are named with numbers (S1, S2 etc.). In reality, the sensor names have a description telling what is being measured (temperature, pressure, etc.) and where it is measured.

First, 5 samples from a sequence with fault type A were explored. The chosen samples range from normal to failure condition. They are marked in figure 5.21. Table 5.10 shows the top 3 sensor contributions for each of the samples. The first sample, which is considered to be in normal operation condition indicates that sensor S3 is the main contribution to the anomaly score. The four remaining samples show an increasing contribution from sensor S6. This makes the anomaly score more transparent and lets a domain expert have an idea of potential faults that are occurring. The exact position of the sensor related to S6 can tell if it is related to an engine, cooling, or other faults. The contribution of each sensor does not make sense for a low anomaly score. It should only be used together with an anomaly score outside of the normal zone, to investigate why there is something wrong with the system.

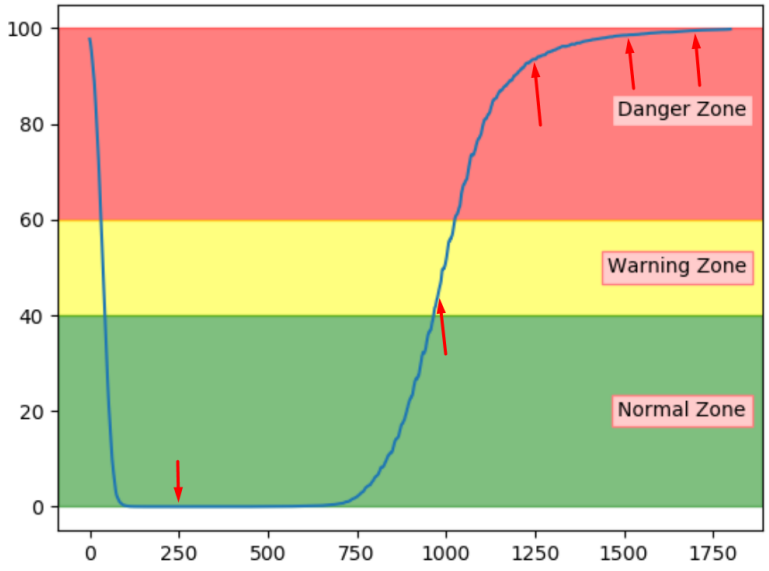



Figure 5.21: Selected test samples from sequence with fault A

Table 5.10: Top sensors contributing to the anomaly score on samples from figure 5.21

| | **Sample #1** | **Sample #2** | **Sample #3** | **Sample #4** | **Sample #5** |
|---|---|---|---|---|---|
| **Top 1 Contributor** | S3: 38.85% | S6: 25.09% | S6: 29.29% | S6: 30.63% | S6: 32.13% |
| **Top 2 Contributor** | S1: 20.38% | S5: 19.30% | S5: 17.78% | S1: 17.83% | S1: 19.41% |
| **Top 3 Contributor** | S8: 10.93% | S1: 14.66% | S1: 16.11% | S5: 17.52% | S5: 17.30% |

Another example of a sequence with fault A was explored to see if the sensor contributions are helpful. This should provide results that indicate a similar trend as the previous example. Figure 5.22 shows the selected samples. According to table 5.11, the contributing sensors are following a similar trend for this sequence, where S6 has an increasing contribution. The contributions from the normal sample can be ignored since the anomaly score is quite low and therefore, no fault to investigate.

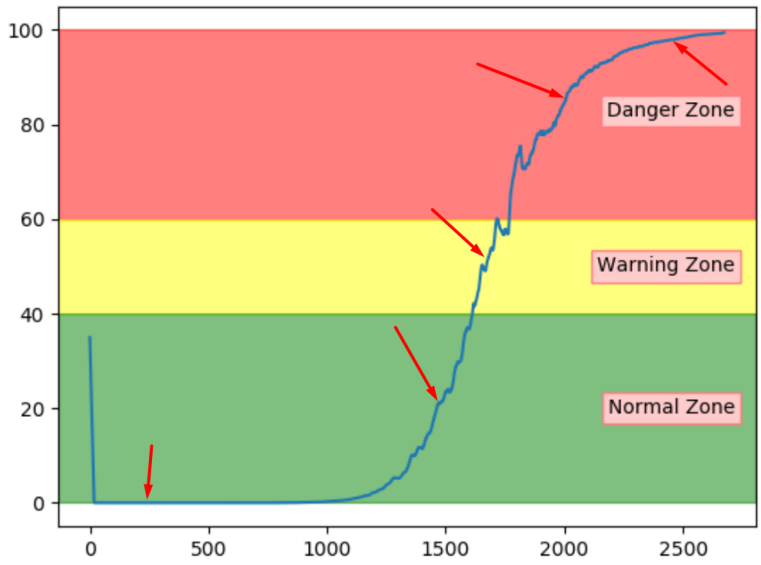


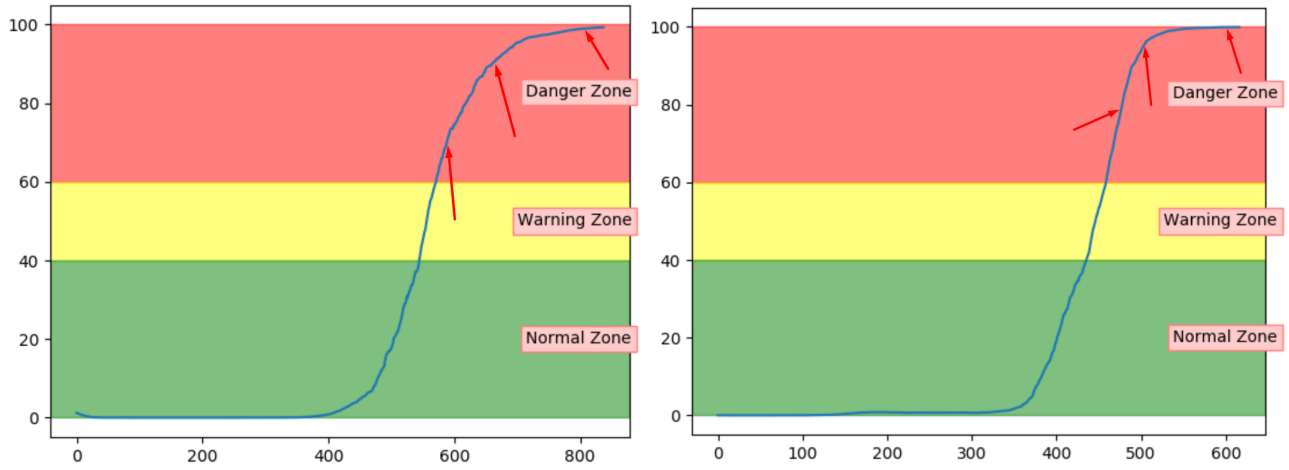

Figure 5.22: Selected test samples from another sequence with fault A

Table 5.11: Top sensors contributing to the anomaly score on samples from figure 5.22

| | **Sample #1** | **Sample #2** | **Sample #3** | **Sample #4** | **Sample #5** |
|---|---|---|---|---|---|
| **Top 1 Contributor** | S3: 27.85% | S6: 32.14% | S6: 32.92% | S6: 33.33% | S6: 33.71% |
| **Top 2 Contributor** | S1: 18.44% | S5: 18.75% | S5: 19.64% | S5: 19.14% | S5: 19.38% |
| **Top 3 Contributor** | S6: 16.87% | S1: 10.45% | S1: 12.70% | S1: 14.50% | S1: 17.73% |

So far, the proposed method to increase transparency seems promising. The next step is to see if a sequence with another fault type has a unique pattern of top contributing sensors. Figure 5.23 shows two sequences with fault type B, and the red arrows indicate the three samples that was investigated. Since the normal samples can be considered as insignificant only error samples are included.

Table 5.12 shows that the samples from the sequence in figure 5.23a indicate sensor S3 as the main contributor to the error. Similarly, table 5.13 shows that S3 was also the main contributor to the other sequence. In addition, the tables show that for both sequences, S5 was also contributing to the anomaly score. This can give service personnel an indication that the fault is somehow related to these two sensors.




(a) Sequence 1 (b) Sequence 2

Figure 5.23: Selected samples for sensor contribution from sequences with fault B

Table 5.12: Top sensors contributing to the anomaly score on samples from figure 5.23a

| | **Sample #1** | **Sample #2** | **Sample #3** |
|---|---|---|---|
| **Top 1 Contributor** | S3: 30.4% | S3: 28.50% | S3: 29.38% |
| **Top 2 Contributor** | S5: 14.58% | S5: 15.28% | S5: 16.11% |
| **Top 3 Contributor** | S1: 11.22% | S4: 11.83% | S4: 12.19% |

Table 5.13: Top sensors contributing to the anomaly score on samples from figure 5.23b

| | **Sample #1** | **Sample #2** | **Sample #3** |
|---|---|---|---|
| **Top 1 Contributor** | S3: 17.23% | S3: 20.41% | S3: 24.99% |
| **Top 2 Contributor** | S5: 17.10% | S5: 16.52% | S5: 14.56% |
| **Top 3 Contributor** | S1: 16.97% | S1: 14.51% | S1: 13.29% |

The proposed method has proved useful in increasing the transparency in the anomaly detection approach. It uncovered that each fault type has a unique pattern of top contributing sensors. These can be used to indicate why the system is behaving abnormally. This is discussed further in chapter 8. The results from the offline fault detection are presented in the next section.

## 5.4 Offline: Fault detection

The offline fault detection is as explained in section 4.4.1 about detecting the fault time step in a sequence. The method for detecting faults is based on finding the point where the reconstruction error accelerates the most in a sequence. It is described in more detail in section 4.4.1.

### 5.4.1 Results

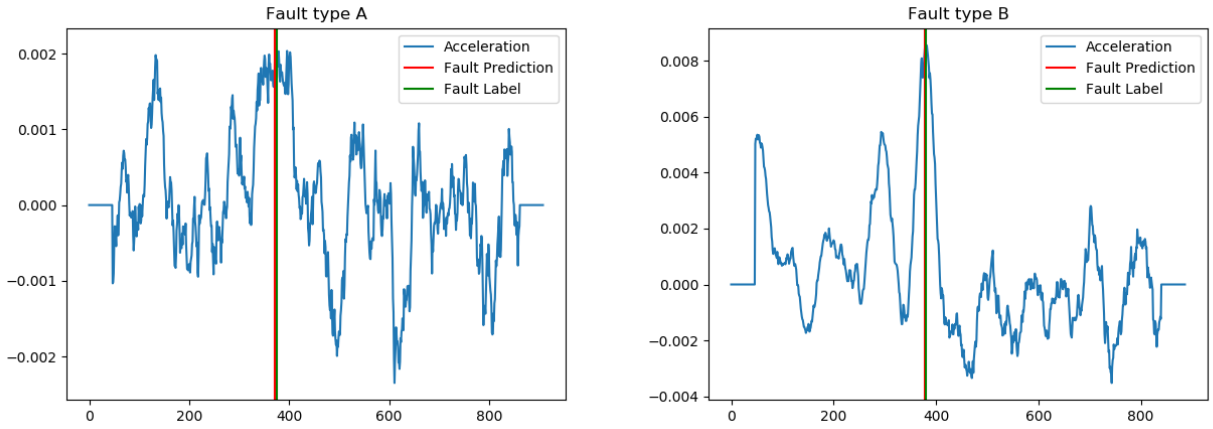
The configuration set was used to test different window sizes. The best performance was achieved with the same $w$ as Ellefsen et al. [106]. The window size was found by dividing the sequence length by 35. The results are measured with accuracy from equation 4.4. Table 5.14 shows the accuracy of the fault detection approach using the 6 different DL models on unseen test data. The leftmost column indicates which sequence was used (A equals sequence with fault type A). The results show that the VAE achieved the best results with an average accuracy above 99%. The other models got an accuracy between 72% and 91%. This indicates that VAE is the most promising model for this purpose.

Table 5.14: Accuracy on unseen test sequences for offline fault detection

| | **AE** | **SAE** | **VAE** | **DBN** | **LSTM** | **CNN** |
|---|---|---|---|---|---|---|
| **A5** | 99.82 | 40.12 | 99.51 | 40.12 | 40.12 | 99.63 |
| **A6** | 99.89 | 99.74 | 99.04 | 40.2 | 31.64 | 98.64 |
| **A7** | 42.89 | 42.89 | 98.86 | 47.59 | 42.89 | 42.89 |
| **B4** | 99.85 | 82.02 | 99.85 | 96.56 | 82.02 | 99.78 |
| **B5** | 99.91 | 99.47 | 99.56 | 98.22 | 99.73 | 99.38 |
| **B6** | 99.3 | 99.41 | 99.31 | 99.21 | 98.71 | 99.01 |
| **B7** | 99.88 | 99.88 | 98.81 | 97.51 | 98.22 | 86.12 |
| **C1** | 89.19 | 89.2 | 99.53 | 89.19 | 89.19 | 81.39 |
| **Average** | **91.34** | **81.59** | **99.31** | **76.08** | **72.82** | **88.36** |

The results indicate that the fault detection approach can recognize when a fault happened with high accuracy. The VAE detects the failure within 2 to 32 time units. The fault A sequences (A5-7 in the table) were detected with an error between 3 and 32 time units, while for fault B sequences (B4-7 in the table) the error was between 2 and 9 time units. The sequence with fault C was only 3 time units from the actual time of the fault. As stated earlier in this thesis, a common problem in the maritime industry is the lack of labels. This approach makes it possible to point out the time when a fault occurred, and therefore use it for labelling.

Figure 5.24a and 5.24b show examples of the fault detection on one sequence with fault type A and one with fault type B. The label is marked in green, while the prediction is marked in red. The figures show that the prediction is at the center of the highest acceleration peak, which is not far from the label.




(a) Sequence with fault type A

(b) Sequence with fault type B

Figure 5.24: Acceleration of the reconstruction error with label and predictions for two sequences

The presented results indicate that the fault detection algorithm proposed by Ellefsen et al. [106] worked in this particular problem. As suggested, the fault detection approach can be used for labelling or root-cause analysis. In this thesis, the argument for exploring this approach is labelling. In the prognostics case (chapter 7) an alternative labelling approach is researched, where the detected fault time-steps are used as a basis. The next case study is about diagnostics and identifying faults.

# Chapter 6

# Case B: Diagnostics

This case study is as described in section 4.4.2 about fault identification. The previous case study showed that it is possible to detect if the system deviates from normal condition and indicate why. This case study aims to identify faults and predict their severity. It has the goal of detecting what is wrong in the system. This can assist decisions on which maintenance actions should be taken. The idea with this case study was to see what can be done when historical data with fault labels are available. The case was explored with three different DL approaches and as two different types of problems. The first type is a multi-label classification variant where each fault type is either 0 or 1. This is referred to as fault identification. This is a normal diagnostics approach indicating what the fault is. Treating the problem as a regression problem is the second approach. This is referred to as severity prediction. For this approach, data was labelled with severity. It requires that the severity of fault types are logged. It can also be combined with binary labels. The obtained anomaly score from the previous case study was added as a feature to the data in this study.

Both types of problems were explored and compared with FNN, LSTM and CNN. Some of the model parameters were tuned using PSO, while others were selected manually. Section 6.1 and 6.2 describes these steps for each model, and present the results. This chapter presents the results and briefly discusses them, while chapter 8 discusses and evaluates them more thoroughly.

## 6.1 Fault identification

This section uses DL techniques to identify faults. It is important to note that this is not a multi-class classification problem, but a multi-label classification problem. The main difference is related to the output of the system. A multi-class problem tries to classify between $x$ classes, where the output of the model is obtained with a softmax activation function resulting in a summed output of 1.0. In a multi-label problem, each output is between 0.0 and 1.0, meaning that it can detect if several classes are present at the same time. To obtain an output between 0.0 and 1.0 for each fault type the *sigmoid*-activation function was used.

As described in section 4.4.2, the diagnostics problems were explored using sequences of normal data,

fault A and fault B. Fault C and D has few examples and was therefore not included. If more fault types are collected, it can be added to the problem as an extra class. For this study the models were used to classify if data is normal $[0, 0]$, suffers from fault A $[1, 0]$ or fault B $[0, 1]$. The most common loss function for multi-class and -label classification problems are called *binary cross-entropy*. It was used as the loss function for this experiment. If more classes are introduced, the categorical cross-entropy should be used. The performance was evaluated based on accuracy.

Initially, some manual experiments were done to find promising ranges of the hyper-parameters. A PSO optimization loop was used to find the best hyper-parameters and architecture of the models. The loop used the 5 first datasets with faults for training, the $6^{\text{th}}$ for validation and the $7^{\text{th}}$ for testing. Ideally, the optimization loop should have used all 7 splits, such that each set of hyper-parameters are properly evaluated. It would have been too time-consuming; hence, the simplified evaluation on one set was used. Table 6.1 presents the PSO specific parameters used for the optimization loops.

Table 6.1: PSO-specific parameters

| Parameter | Values |
|---|---|
| Inertia | 0.5 |
| Cognitive | 0.8 |
| Social | 0.6 |
| #Particles | 10 |
| #Iterations | 10 |

**FNN**

Some parameters were found through manual experiments to simplify the optimization loop. First, four optimizers were tested (AdaGrad, RMSProp, SGD, and Adam). They performed quite similar, but Adam was selected as it tended to give slightly better results. The output activation function was already decided to be *sigmoid* since the output is between 0 and 1 for each class. In a classification problem, this means that a threshold of 0.5 is used to determine the binary classification. It was found that using 4 layers, where the $4^{\text{th}}$ layer is the output layer was the most promising. The final layer has two nodes; one for each fault type. All weights in the network were initialized with Xavier weight initialization. Early stopping was used for preventing the network from over-fitting. Manual experiments indicated that training always converged before 30 epochs. This number was also used in the optimization loop. The remaining hyper-parameters that was adjusted with PSO is presented in table 6.2. The best result from the optimization loop is marked in the table.

Table 6.2: Hyper-parameters for FNN for classification

| Hyper-parameter | Values |
|---|---|
| Learning rate | 0.001, **0.0001**, 0.00001 |
| Batch size | **10**, 25, 50 |
| Units layer h1 | 10, **14**, 18, 22 |
| Units layer h2 | 10, 14, **18**, 22 |
| Units layer h3 | 6, **8**, 10, 12, 14 |
| Dropout | 0.0 - 0.3 (**0.15**) |
| Activation | sigmoid, **tanh**, relu |

### LSTM

Similarly, to the FNN, some of the parameters for LSTM was decided through manual experiments. Of the four optimizers, RMSProp performed the best. LSTM architecture is often combined with normal dense (FNN) layers to make the final decision. Experiments indicated that using two LSTM layers and two dense layers gave the best results. The final dense layer is the output layer. The LSTM layers used the default activation functions which is *tanh*, and the output layer used a *sigmoid* activation function. The activation function for the first dense layers were determined with PSO. Table 6.3 presents the parameters that are tuned with PSO loop, the best choice is marked.

Table 6.3: Hyper-parameters for LSTM for classification

| Hyper-parameter | Values |
|---|---|
| Learning rate | 0.001, **0.0001**, 0.00001 |
| Batch size | 10, 50, **100** |
| Units layer h1 | 10, **14**, 18, 22 |
| Units layer h2 | 10, **14**, 18, 22 |
| Units layer h3 | 6, **8**, 10, 12, 14 |
| Activation | Sigmoid, **tanh**, relu |
| Time window | **5**, 10, 15, 20, 30, 40, 50 |

### CNN

Obtaining a good architecture for the time-window CNN is harder since it has many parameters to tune. Manual experiments indicated that using two sets of convolutional and max-pooling layers in combination with three dense layers were most promising. Dropout layers were added between the dense layers. The RMSProp optimizer was selected since it was most promising. Table 6.3 shows the parameters tuned with PSO, and the best parameters are marked.

Table 6.4: Hyper-parameters for CNN for classification

| Hyper-parameter | Values |
|---|---|
| Learning rate | 0.001, **0.0001**, 0.00001 |
| Batch size | 10, 50, **100** |
| #Filters - Conv2D (1) | 4, 8, 12, **16** |
| Kernel - Conv2D (1) | 5-10 (**5**) |
| Kernel - Pool (1) | 2, **3**, 4 |
| #Filters - Conv2D (2) | 4, 8, 12, **16** |
| Kernel - Conv2D (2) | 5-10 (**6**) |
| Kernel - Pool (2) | 2, **3**, 4 |
| Units - Dense (1) | 10, **20**, 30 |
| Units - Dense (2) | 6, **10**, 14 |
| Dropout | 0.0 - 0.3 (**0.2**) |
| Activation | sigmoid, **tanh**, relu |
| Time window | 5, 10, **15**, 20, 30, 40, 50 |

The results from the models were evaluated with the k-fold cross-validation loop explained in section 4.4.2. The results were analyzed based on accuracy. Table 6.5 shows how each model performed averaged over the 7 splits.

Table 6.5: Results from fault classification

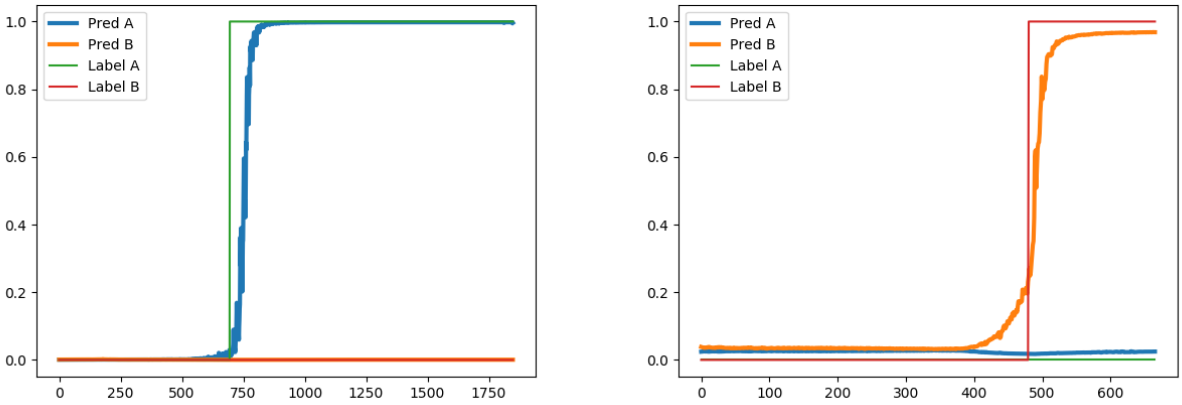
| | **Accuracy** |
|---|---|
| **FNN** | 0.9691 |
| **LSTM** | 0.9646 |
| **CNN** | 0.9674 |

The table shows that the models performed with an accuracy of around 0.96. This means that the models were accurately able to identify faults in the air compressor. The performance of the models can be ordered as FNN, CNN and LSTM. The accuracy of each split was investigated. Table 6.6 shows that the accuracy on most splits were between 0.95 and 1.0. All models performed the worst on split 3, with an accuracy between 0.88 and 0.92. This is discussed later.

Table 6.6: Results per split on fault classification

| | **Accuracy** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Models** | **Split 1** | **Split 2** | **Split 3** | **Split 4** | **Split 5** | **Split 6** | **Split 7** |
| **FNN** | 0.9745 | 0.9836 | 0.9027 | 0.9953 | 0.9788 | 0.9628 | 0.9856 |
| **LSTM** | 0.9724 | 0.9864 | 0.8890 | 0.9928 | 0.9758 | 0.9497 | 0.9859 |
| **CNN** | 0.9657 | 0.9925 | 0.9221 | 0.9928 | 0.9574 | 0.9565 | 0.9845 |

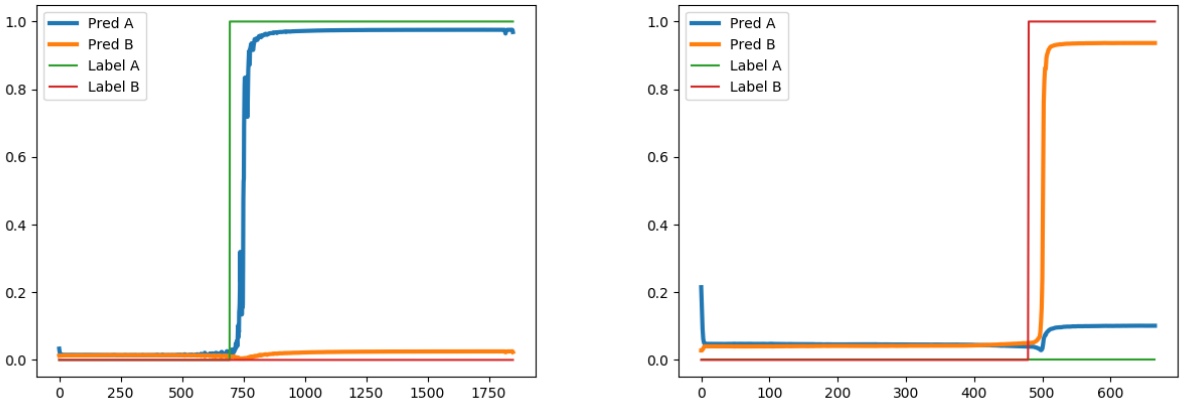
Figure 6.1a and 6.1b show the raw prediction probabilities for one sequence of fault type A and one with fault type B. In both sequences the FNN was quite accurately able to distinguish between normal operation and the different fault types.




(a) Fault type A (b) Fault type B

Figure 6.1: Fault classification based on FNN predictions

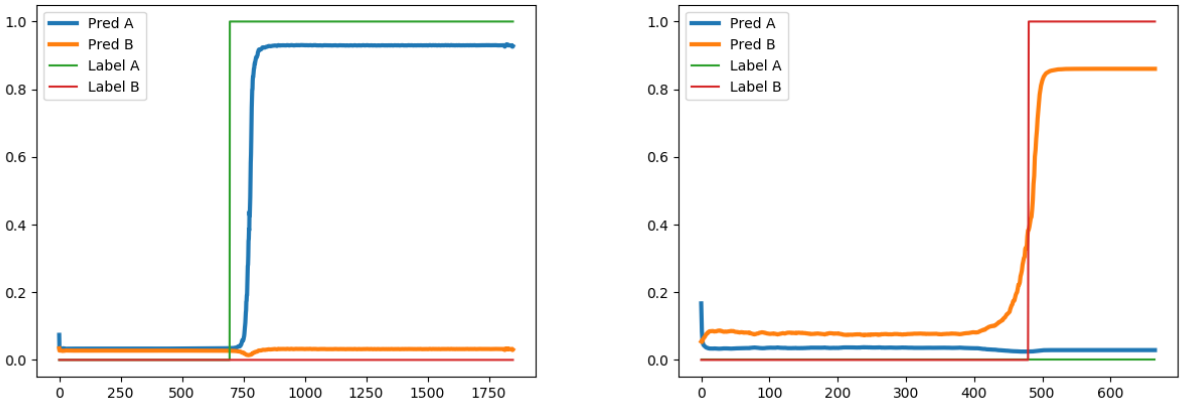
The LSTM performed similarly on the same sequences. This is indicated in figure 6.2a and 6.2b.




(a) Fault type A (b) Fault type B

Figure 6.2: Fault classification based on LSTM predictions

Figure 6.3a and 6.3b show that the CNN also achieved good results. Visually comparing the results showed that the raw predictions struggled to reach the level of 1.0. The classification threshold is 0.5, which means that it is considered a correct prediction anyway. It can still be interpreted as a prediction with less confidence.

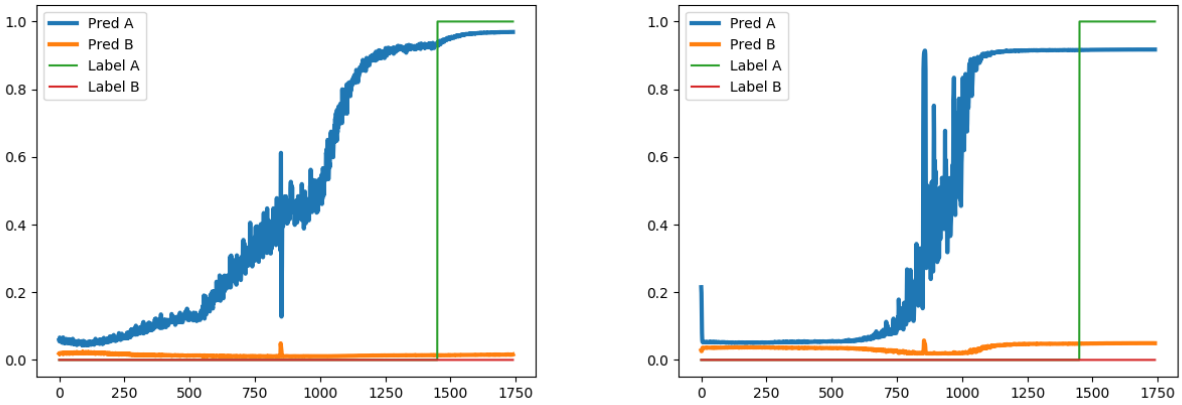



(a) Fault type A

(b) Fault type B

Figure 6.3: Fault classification based on CNN predictions

The figures has indicated results that are able accurately identify faults. The results on split 3 has received the lowest accuracy. Figure 6.4a and 6.4b show how the FNN and CNN predicted the faults on a sequence with fault A from split 3. Both figures indicated that the models predicted the fault too early. The results are therefore not satisfactory on split 3.




(a) FNN

(b) CNN

Figure 6.4: Fault classification on split number 3

Except for split 3, the models were able to accurately identify faults in the air compressor. Next, the problem is treated as a regression problem to predict the severity of the faults.

## 6.2 Severity prediction

For this approach, the goal was not only to identify the faults, but also to predict the severity of them. This means that each of the sequences were labelled with 4 steps; 0.0, 0.33, 0.67 and 1.0 as illustrated

in section 4.4.2. The output activation function was the identity function. Similar to the previous section, sequences of normal, fault A and fault B data are used. The performance of the models was evaluated based on MAE, which indicates how far from the label, the predictions in general are. The predictions were also analyzed visually.

For simplicity, the architectures obtained from the previous section were used. The output activation function are changed from sigmoid to identity. The models were trained with 30 epochs. K-fold cross-validation was used to evaluate the average performance of the models. Table 6.7 shows the performance of the models in the form of MAE. The models performed similarly. FNN achieved the best results by in general missing the label with 0.0626.

Table 6.7: Results from diagnostics with severity prediction

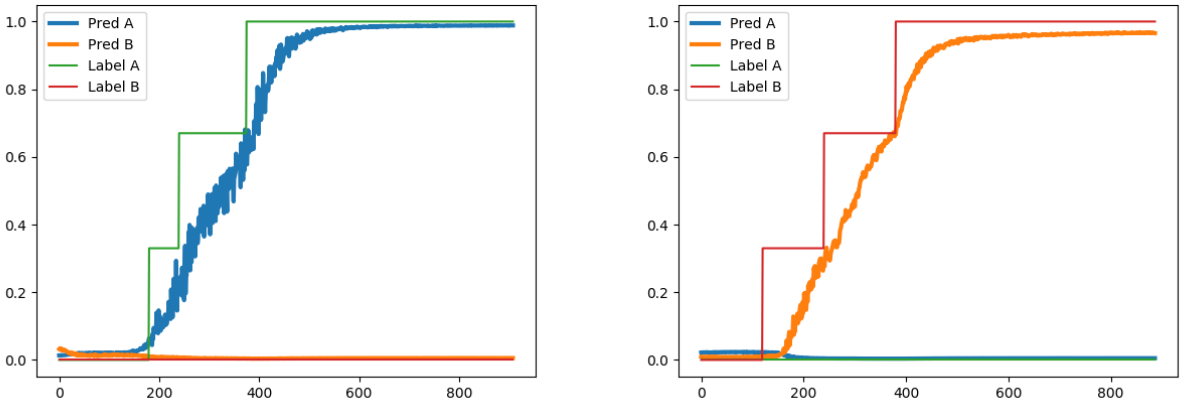
| | **MAE** |
|---|---|
| **FNN** | 0.0626 |
| **LSTM** | 0.0674 |
| **CNN** | 0.0753 |

Table 6.8 shows the MAE for each individual split. Similarly to the classification results, the table shows that the prediction on split 3 gives the worst results for FNN and LSTM. The CNN is performing better on split 3, but worse on split 6.

Table 6.8: Results from severity prediction on each individual split

| | **MAE** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Models** | **Split 1** | **Split 2** | **Split 3** | **Split 4** | **Split 5** | **Split 6** | **Split 7** |
| **FNN** | 0.0563 | 0.0474 | 0.1270 | 0.0463 | 0.0327 | 0.0679 | 0.0626 |
| **LSTM** | 0.0634 | 0.0560 | 0.1166 | 0.0572 | 0.0345 | 0.0602 | 0.0674 |
| **CNN** | 0.0570 | 0.0493 | 0.0851 | 0.0805 | 0.0471 | 0.1118 | 0.0967 |

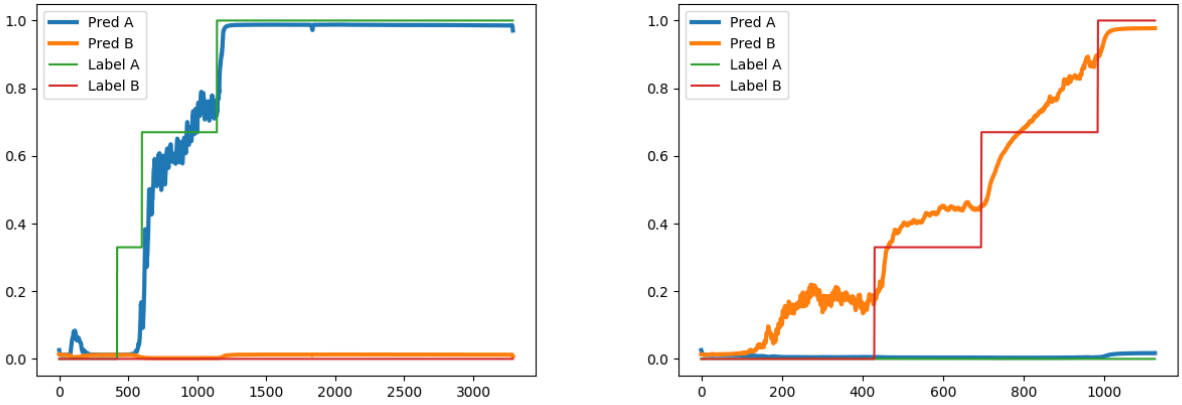
While the MAE gives information about the average difference from the label, visual analysis is important to inspect and understand the results. Figure 6.5a and 6.5b show how the FNN performed on the two sequences in split 1. The predictions was not able to predict the steps properly. It was on the other hand able to give an early indication of increasing severity, increasing from 0 to 1.




(a) Fault type A (b) Fault type B

Figure 6.5: FNN diagnostics predictions with regression approach

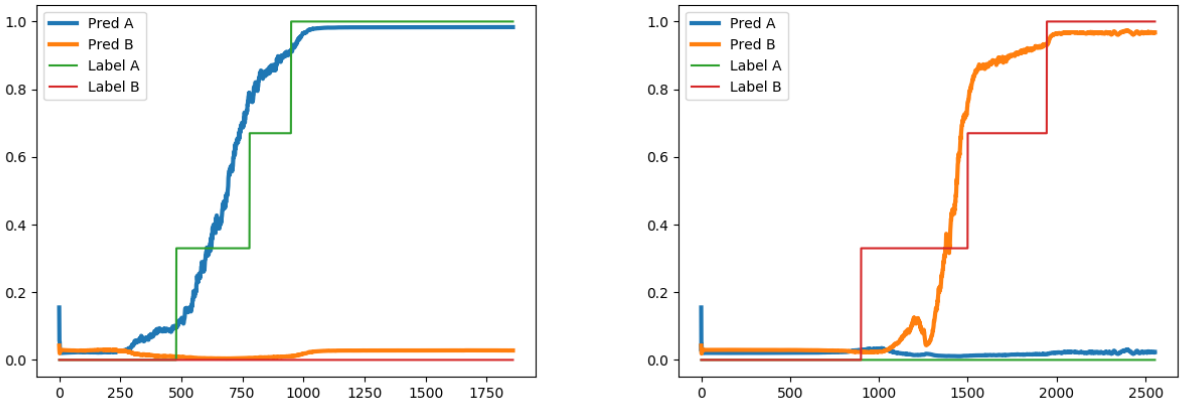
Figure 6.6a and 6.6b show that the LSTM also struggled to predict the steps. It was still able to give a clear indication of a progressing fault. The 100% severity was predicted accurately on both sequences.




(a) Fault type A (b) Fault type B

Figure 6.6: LSTM diagnostics predictions with regression approach

Prediction with CNN achieved comparable results with similar trends as the two other models. Figure 6.7a and 6.7b show the prediction on sequences with both fault types from split 1.

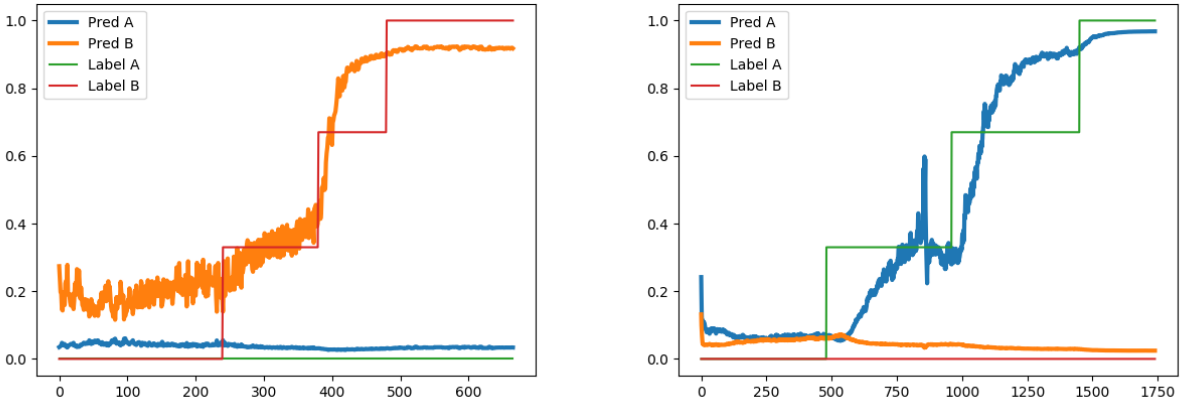



(a) Fault type A

(b) Fault type B

Figure 6.7: CNN diagnostics predictions with regression approach

The figures so far have shown promising results. The performance on the individual splits indicated that the models performed the worst on split 3. Figure 6.8a shows the prediction obtained from FNN on split 3 with fault type B. Figure 6.4b show predictions from CNN on split 3 with fault type A. The results were less accurate, but still able to give an indication of the severity of the faults. The predictions in figure 6.4a struggled to clearly indicate both the 0% and 100% severity level.




(a) FNN on fault B

(b) CNN on fault A

Figure 6.8: Severity prediction on sequence from split 3

Two different approaches for identifying faults in air compressors have been presented in this chapter. The first approach was able to identify the fault. The second approach was able to identify the faults and predict their severity. It gave an earlier indication of faults and fault types in the compressor. In the next chapter, prognostics are explored.

# Chapter 7

# Case C: Prognostics

A PHM system can have several features working together to provide insight into the condition. The case study on anomaly detection showed that it is possible to detect if the system is deviating from normal operating condition. The previous section presented alternatives for identifying faults and their severity. In this chapter, prognostics on air compressors are explored. As described in section 4.4.2, the case study explores predictions of RUL on air compressors. Three topics related to prognostics was researched. First, section 7.1 explores predicting RUL on air compressors with three different DL techniques, including deciding on labelling approach. Section 7.2 investigates transfer learning in PHM context, while section 7.3 explores how uncertainty bounds can be achieved.

## 7.1 Predict remaining useful life

The experiments with predicting RUL on air compressors were done based on sequences with fault A and B, as explained in section 4.4.3. These sequences start in normal operating condition and were introduced for faults with increasing severity. After the fault has reached 100% severity, the compressor runs until end-of-life. The end of a sequence is therefore considered the end-of-life. This is the event that is desired to predict time until.

### 7.1.1 Pre-processing

One of the most explored datasets for prognostics has more than 200 sequences of system failures. The available data in this research consist only of 14 sequences. In an attempt to increase the number of sequences, data augmentation was used as described in section 4.4.3. The argument for applying data augmentation is that it provides more examples available for each RUL level. It was used to divide each sequence into five new sequences. Instead of having 14 examples of RUL of 50, the dataset now contains $5 * 14 = 70$ examples.

As described in section 4.4.2, the piece-wise RUL labelling approach was used. To decide how long in advance the failures should be detected, discussions with domain-experts was done. According to

domain-experts, it was hard to set a limit since the faults are forced in an unnatural speed. Therefore, it was more convenient to look at the data to find suitable values. Table 7.1 shows the number of time units from 100% fault severity until end-of-life for the original sequences.

Table 7.1: Overview of RUL on original sequences

| Fault Type | Minimum RUL | Maximum RUL | Mean RUL |
|---|---|---|---|
| Fault A | 290 | 2145 | 1283 |
| Fault B | 113 | 609 | 289 |

For the augmented sequences, this will be divided by 5. This means that the minimum time between fault and failure for fault type B is around 23, and for fault A, 58. Predicting only 23 time units before failure is too little. Through experiments, it was decided to try to predict 100 time units in advance. This means that the linear label might start before 100% severity is reached, but it will give more time to make maintenance decisions. The original 14 sensor measurements and the obtained anomaly score were used as inputs in these experiments. The inputs were normalized using z-normalization (equation 4.2). The normalizer was only fitted with training data to avoid information leakage. The label was scaled to between -1 and 1 to simplify the prediction range. This makes the tanh activation function a perfect fit since its output is bounded between the same range. To make the predictions informative, they need to be re-scaled into its original scale, before presented to the user. Results from the models were evaluated in the original scale to have an informative measure of performance.

### 7.1.2 Model architecture & parameters

The hyper-parameters and architectures for each model were tuned similarly to what was done in the previous case study (section 6.1). The same PSO-specific parameters were used for these experiments. Every model was first tuned manually to find promising regions of architecture before running the optimization loops.

#### FNN

Among the four optimizers (SGD, RMSProp, Adam, and AdaGrad), Adam was selected as it gave good results in manual experiments. The experiments indicated that a FNN with three hidden layers was most promising. Dropout was used between the hidden layers. Table 7.2 shows the parameters that were tuned with PSO and the best parameters found are highlighted.

Table 7.2: FNN hyper-parameters for prognostics

| Hyper-parameter | Values |
|---|---|
| Learning rate | 0.001, **0.0001**, 0.00001 |
| Batch size | 10, **25**, 50 |
| Units layer h1 | 10, **20**, 30, 40 |
| Units layer h2 | 10, 20, **30**, 40 |
| Units layer h3 | 10, **20**, 30, 40 |
| Dropout | 0.0 - 0.3 (**0.2**) |
| Activation | sigmoid, **tanh**, relu |

**LSTM**

Through manual experiments, many architectures for the LSTM was tested. The most promising architecture was with three LSTM layers and two dense (FNN) layers. The last dense layer is the output layer. The RMSProp optimizer was selected based on its performance in manual experiments. The output activation function was as mentioned tanh. For the LSTM layers, the tanh activation function was used (default), but the activation function for the dense layers were tuned with PSO. Training was performed with 30 epochs and early stopping. Table 7.3 shows the parameters that were optimized with PSO, the best parameters found are highlighted in the table.

Table 7.3: LSTM hyper-parameters for prognostics

| Hyper-parameter | Values |
|---|---|
| Learning rate | 0.001, 0.0005, 0.0001, **0.00005**, 0.00001 |
| Batch size | 10, **40**, 70, 100 |
| Units layer LSTM1 | 10, 15, **20**, 25, 30, 35, 40 |
| Units layer LSTM2 | 10, 15, 20, 25, **30**, 35, 40 |
| Units layer LSTM3 | 10, 15, 20, 25, **30**, 35, 40 |
| Units layer Dense1 | 5, 10, 15, **20**, 25, 30 |
| Activation | Sigmoid, **tanh**, relu |
| Time window | 5, 10, 15, **20**, 30, 40, 50 |

**CNN**

As mentioned in section 6.1, CNN is a model with many parameters to tune. Manual experiments were performed to find a promising architecture. These experiments indicated that the most promising architecture was to use two sets of layers consisting of convolutional layer and max-pooling. Next, flattening and then three dense layers. The final dense layer is the output layer. A padding strategy called *same* was used for all convolutional and max-pooling layer, except the final max-pooling layer where valid was used. This means that the dimension is reduced instead of kept. Dropout was added between the two first dense layers. The RMSProp optimizer was selected since it showed the most promising results. The remaining parameters were tuned with PSO. Table 7.4 shows the parameters that were tuned, and the best parameters are highlighted.

Table 7.4: CNN hyper-parameters for prognostics

| Hyper-parameter | Values |
|---|---|
| Learning rate | 0.001, 0.0005, 0.0001, **0.00005**, 0.00001 |
| Batch size | 10, **40**, 70, 100, 130 |
| #Filters - Conv2D (1) | 4, 8, 12, **16**, 20 |
| Kernel size - Conv2D (1) | 4-10 (**5**) |
| Kernel size - Pool (1) | **2**, 3, 4 |
| #Filters - Conv2D (2) | 4, 8, 12, **16**, 20 |
| Kernel - Conv2D (2) | 5-10 (**4**) |
| Kernel - Pool (2) | 2, **3**, 4 |
| Units - Dense (1) | 10, **20**, 30 |
| Units - Dense (2) | 4, **8**, 12, 16, 20 |
| Dropout | 0.0 - 0.3 (**0.15**) |
| Activation | sigmoid, **tanh**, relu |
| Time window | 5, 10, **15**, 20, 30, 40, 50 |

### 7.1.3 Results

After the best architecture and hyper-parameters were found, each model was trained and evaluated using k-fold cross-validation. The performance of the models was mainly evaluated with MAE, which indicates the average error from the target. RMSE was also used, which punishes large errors more. Table 7.5 shows the MAE and RMSE for each of the models on the averaged performance from k-fold cross-validation.

Table 7.5: Results from RUL prediction using k-fold cross validation

| | **RMSE** | **MAE** |
|---|---|---|
| **FNN** | 14.04 | 9.27 |
| **LSTM** | 11.03 | 6.87 |
| **CNN** | 13.14 | 8.34 |

The results show that there were some large differences between the models. The LSTM was clearly performing the best with a MAE of 6.87. This means that the model on average predicted 6.87 time units from the target. The second best model was the CNN, and worst was FNN. This is interesting since the FNN performed the best on diagnostics. Table 7.6 shows the MAE on each individual split. The table indicates that the performance on split 3 was much worse than for the other splits. The LSTM achieved an average MAE of 22, and as high as 33 for FNN. Since both the diagnostics and prognostics experiments show that split 3 performs the worst, it can be assumed that the data was collected under quite different conditions or operation. This is discussed further in chapter 8.

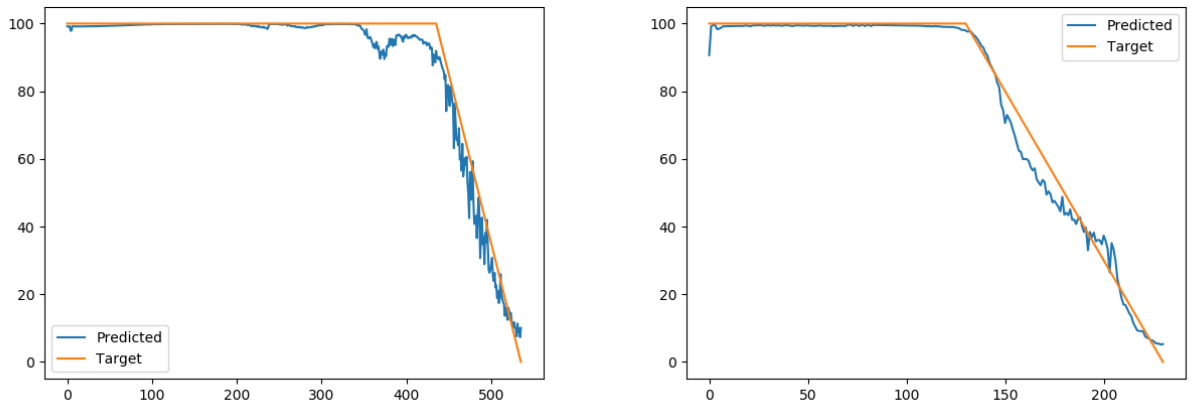
Table 7.6: Results from RUL predictions on each individual split

| | MAE | | | | | | |
|---|---|---|---|---|---|---|---|
| **Models** | **Split 1** | **Split 2** | **Split 3** | **Split 4** | **Split 5** | **Split 6** | **Split 7** |
| **FNN** | 4.00 | 7.16 | 33.10 | 2.42 | 6.93 | 4.45 | 6.85 |
| **LSTM** | 4.92 | 6.04 | 22.04 | 3.77 | 3.72 | 1.85 | 5.76 |
| **CNN** | 3.70 | 6.10 | 26.91 | 5.07 | 5.00 | 3.81 | 7.81 |

When excluding split number 3 the predictions from the LSTM model were on average 4.34 time units away from the correct RUL. So far, the RUL predictions have only been evaluated based on the performance measure. The predictions were also analyzed visually. This makes it possible to notice if the predictions were fluctuating, over-estimating, under-estimating, etc.

### FNN

First, the FNN predictions were explored. Split number 4 achieved a MAE of 2.42. Figure 7.1a and 7.1b shows the predictions on one sequence of each fault type from that split. The figures prove that the predictions were accurate and very close to the actual RUL of the compressor.

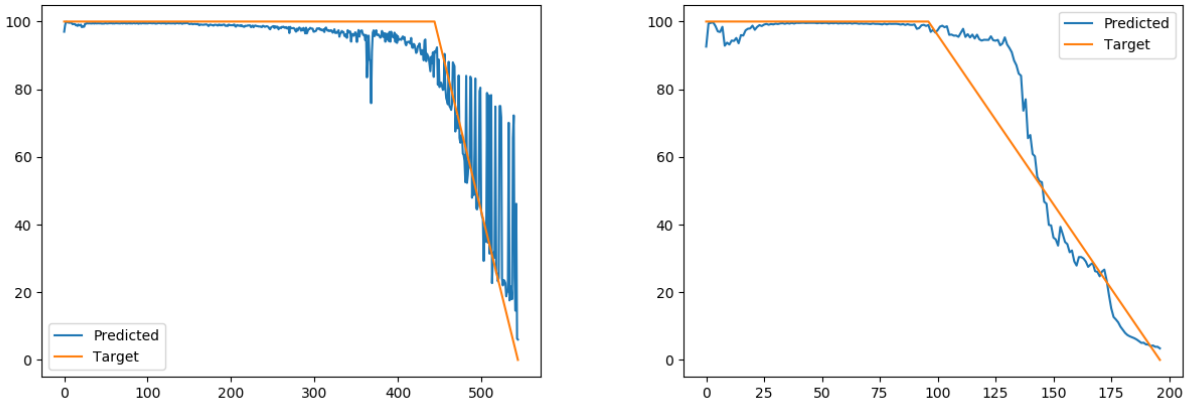



(a) Fault type A

(b) Fault type B

Figure 7.1: RUL prediction from FNN on split 4

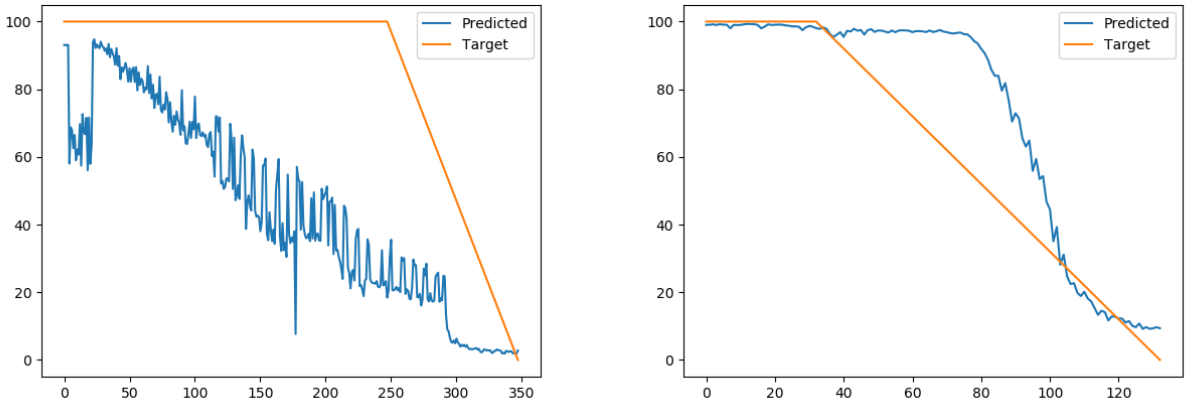
The FNN achieved variable results for the other splits. On split number 6, it achieved a relative low MAE, but as figure 7.2a indicates, the predictions on a sequence with fault type A from split 6 had much noise. Several of the predictions from FNN on sequences with fault type A have similar fluctuations. This could have been reduced by applying a moving average filter. Figure 7.2b shows that the predictions on a sequence with fault type B followed the target relatively good. It had less noisy, but were a bit late to start predicting the linear RUL.




(a) Fault type A (b) Fault type B

Figure 7.2: RUL prediction from FNN on split 6

As indicated earlier, predictions on split number 3 have performed much worse than the others. This is proved in figure 7.3a and 7.3b, which are predictions on a sequence with fault A and B from split 3. The predictions on the sequence with fault type A were far from the target and under-estimated the RUL by a lot. The other sequence was over-estimated large parts of the linear RUL.

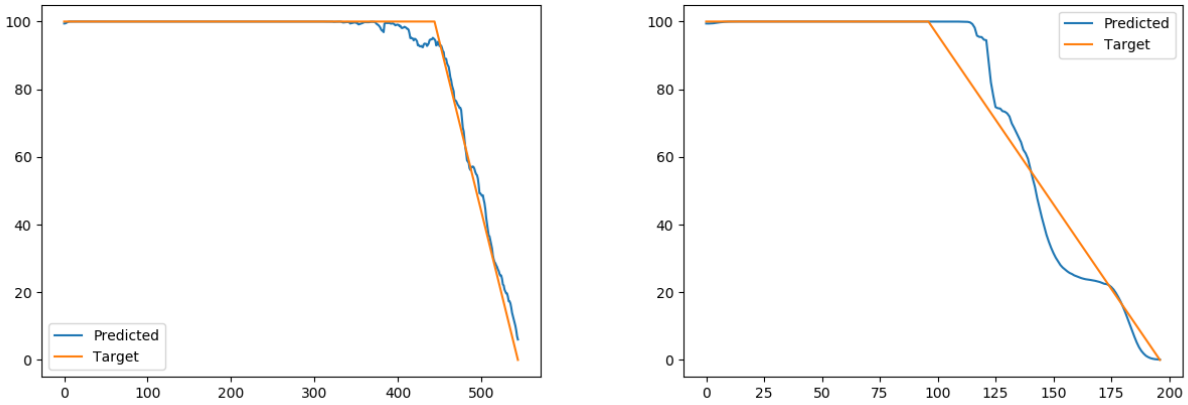



(a) Fault type A (b) Fault type B

Figure 7.3: RUL prediction from FNN on split 3

### LSTM

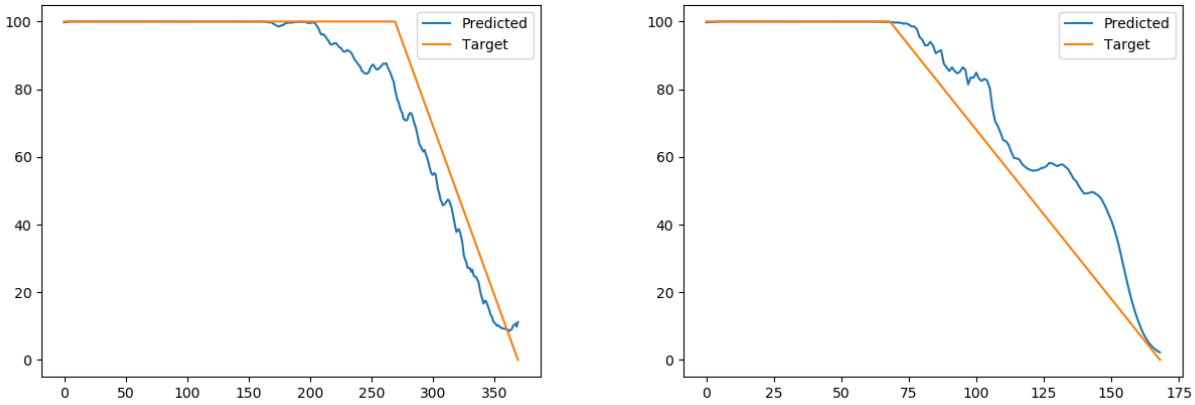
Predictions from the LSTM model achieved the lowest MAE and can be considered the best performing model. Figure 7.4a shows that the model were accurately predicting the RUL on a sequence with fault type A from split 6. The predictions from the same split, but on a sequence with fault B was not as accurate, but still a good prediction.




(a) Fault type A (b) Fault type B

Figure 7.4: RUL prediction from LSTM on split 6

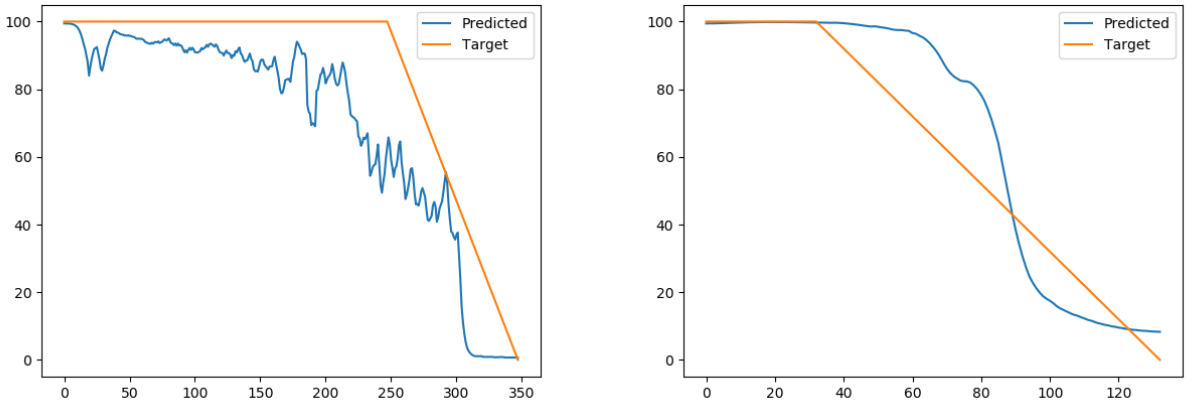
Predictions on split number 7 achieved a higher MAE than four other splits. The sequence with fault A slightly under-estimated the RUL (figure 7.5a), while the sequence with fault B is over-estimated a little. Under-estimation is considered better since the maintenance can be done before something breaks. Over-estimation may lead to failure before maintenance actions took place.




(a) Fault type A (b) Fault type B

Figure 7.5: RUL prediction from LSTM on split 7

As stated earlier, both the LSTM and FNN achieved the highest MAE on split number 3. Figure 7.6a shows that the LSTM predictions on the sequence with fault A were not as accurate as the ones seen so far. It was more accurate than FNN on the same sequence. The predictions under-estimated the RUL, but were at least able to indicate that the system was degrading in advance of a failure. The sequence with fault B over-estimated less than the FNN prediction, and even under-estimated the target towards the end-of-life. This is indicated in figure 7.6b.



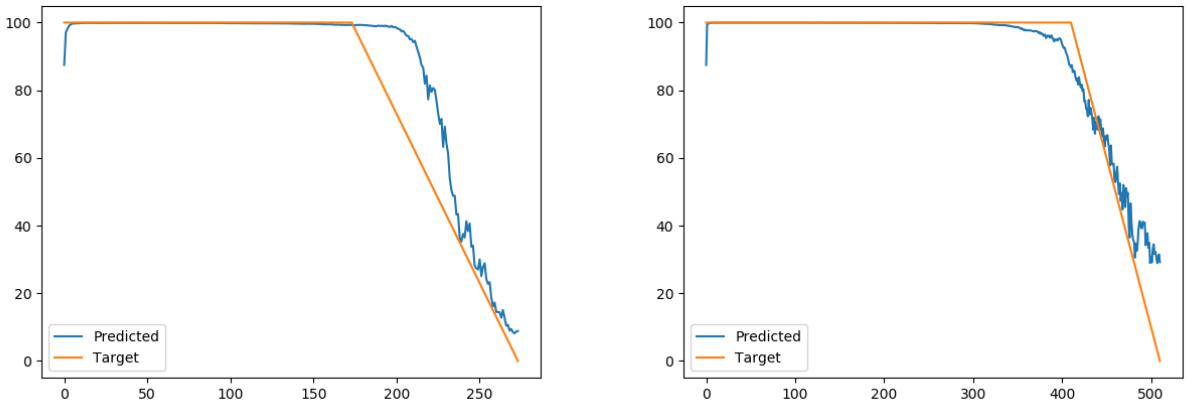

(a) Fault type A (b) Fault type B

Figure 7.6: RUL prediction from LSTM on split 3

The LSTM has proven superior to the FNN both when it comes to the score and the visual analysis. The LSTM achieved satisfactory results for all splits, except split 3. The results on split 3 were at least better for LSTM than for FNN.

### CNN

The MAE showed that the CNN performance was ranked between FNN and LSTM. The best CNN results were achieved on split 1. The prediction on a sequence with fault A and B from that split is showed in figure 7.7a and 7.7b, respectively. The results for fault A show that the prediction slightly over-estimated the RUL in early stages of degradation. The other sequence shows that the RUL was estimated accurately until about 30 time units was remaining.

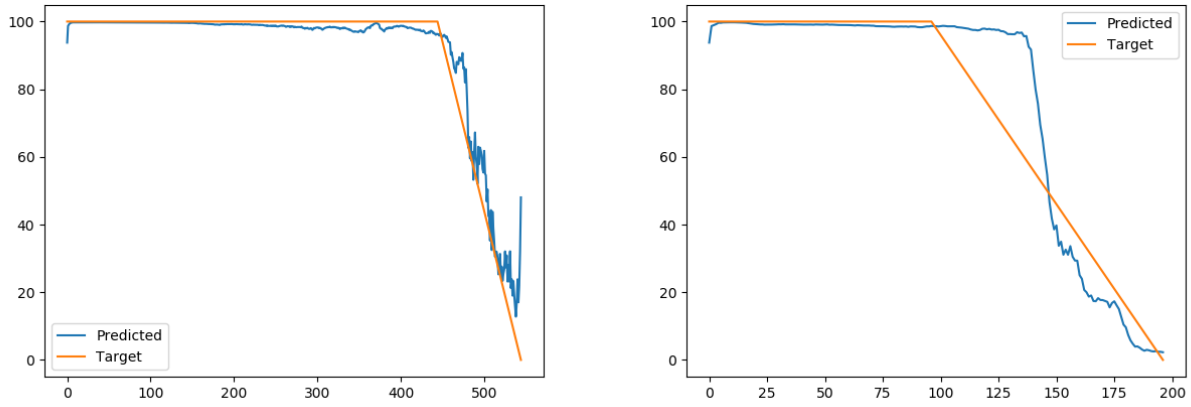



(a) Fault type A (b) Fault type B

Figure 7.7: RUL prediction from CNN on split 1

The CNN achieved varying results on split 6. The predictions on a sequence with fault A (fig 7.8a)

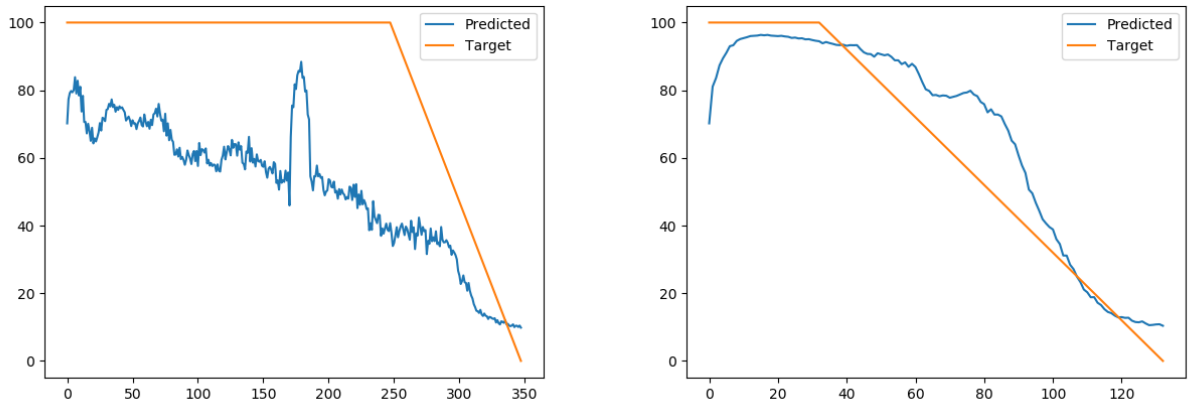
were following the target closely, but with some noise, especially towards the end-of-life. The model performed worse on the sequence with fault B, where the reduction in RUL was detected too late. It over-estimated the early linear phase of the prediction.




(a) Fault type A (b) Fault type B

Figure 7.8: RUL prediction from CNN on split 6

Figure 7.9a shows that the CNN also struggled with predicting the RUL on the sequence with fault A on split 3. The prediction under-estimated the target by a lot for almost the entire sequence. The results on the sequence with fault B were not as bad, but it over-estimated the RUL mid-sequence (figure 7.9b).




(a) Fault type A (b) Fault type B

Figure 7.9: RUL prediction from CNN on split 3

**Summary**

The results have indicated that there were considerable differences in the three DL models' performance. LSTM performed best and therefore considered the best model for predicting RUL, in this particular case. The predictions were accurate, and they were on average missing in the upper edge

of 6 time units from the target. This is promising results. So far, the RUL has been predicted with the commonly accepted RUL labelling approach called piece-wise linear RUL. As mentioned in section 4.4.3, new research proposed an alternative labelling approach referred to as the adaptive piece-wise linear RUL. This is investigated with LSTM in the next section.

### 7.1.4 Alternative labelling

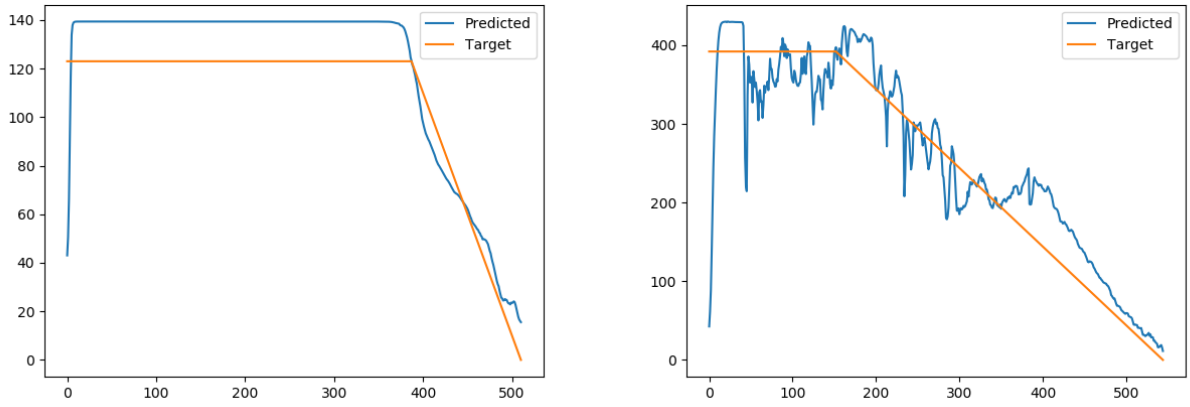
The state-of-the-art labelling approach proposed by Ellefsen et. al [123] were explored. The alternative labelling approach is described in more detail in section 4.4.3. The basic idea is that the RUL label is constant until a fault is detected; from that point, the RUL is linear. This means that the constant value is different between the sequences. The labelling is dependent on the offline fault detection that was explored in section 5.4. To explore the alternative labelling, the LSTM was used. It was selected due to its performance on the traditional labels. For this experiment, the same architecture for the LSTM was used, but since the upper limit of the RUL was so different for each sequence, the label were not normalized. Therefore, the output activation function was changed to the identity function. The activation function of the dense layer was changed to ReLU.

The new labelling approach was evaluated in the same way as the previous case. The average MAE and RMSE obtained were 37.32 and 51.38, respectively. This was much higher than the results obtained with the other labelling approach. It must be noted that the scores are not directly comparable. It does, on the other hand, indicate that the predictions were on average missing almost 40 time units from the label. Table 7.7 shows the results per split. These results were also much higher than the other labelling approach.

Table 7.7: Results on each split with alternative labelling

| | **MAE** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Models** | **Split 1** | **Split 2** | **Split 3** | **Split 4** | **Split 5** | **Split 6** | **Split 7** |
| **LSTM** | 27.57 | 15.43 | 4.91 | 37.19 | 108.12 | 30.22 | 37.77 |

Since the results were worse based on the scoring, the predictions were also analyzed visually. Figure 7.10a shows the prediction on a sequence with fault B, from split 1. The linear RUL was predicted accurately, while the constant level had an offset of 20 time units. Figure 7.10b shows a prediction on a sequence with fault A from split 5. The constant RUL-level was as high as 400 time units in advance of failure. Despite large fluctuations, the predictions were decreasing around the target relatively good. They can indicate the RUL four times longer in advance than the other labelling approach.

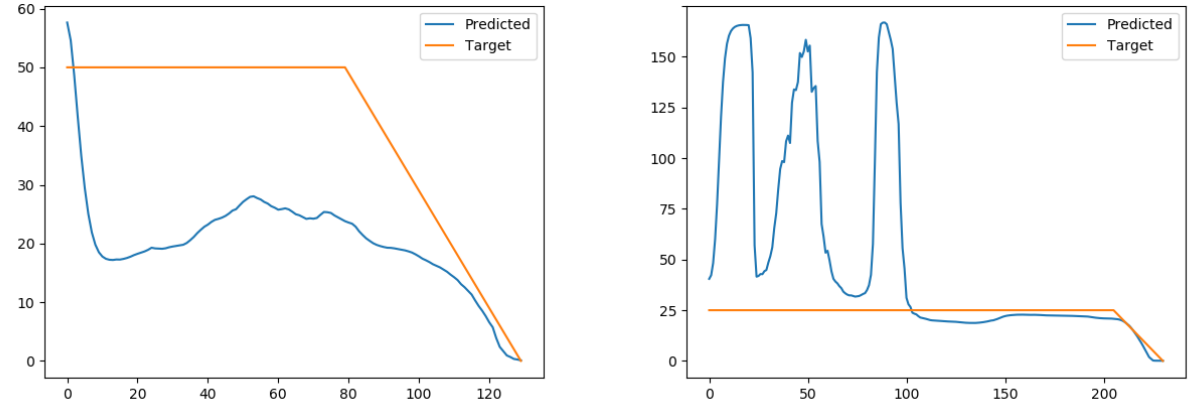



(a) Fault type B

(b) Fault type A

Figure 7.10: RUL predictions with adaptive piece-wise labels I

The two previous figures showed promising results. The two next figures show that other sequences were predicted with large errors. Figure 7.11a shows that a prediction on a sequence with fault type A under-estimated a lot. The prediction was not able to give any useful indication of the RUL. Similarly, figure 7.11b shows that the prediction on a sequence with fault B struggled to a find constant level. The prediction missed with more than 125 time units. The linear part of the RUL was predicted relative accurate, but it was only 25 time units in advance.




(a) Fault type A on split 2

(b) Fault type B on split 4

Figure 7.11: RUL predictions with adaptive piece-wise labels II

The results have shown varying performance with the adaptive piece-wise linear RUL label. The results were much more inaccurate than the traditional RUL labels and can be considered as not useful for this particular case. Service personnel will only have a value of the predicted RUL to decide from. If the value is 25, it can mean that the system is in normal condition (see figure 7.11b), while in other cases it is about to fail (see figure 7.10b). These results are discussed more thoroughly in section 8. In the next section, transfer learning is investigated to see if it can improve the RUL predictions.

## 7.2 Transfer learning

Transfer learning was researched to see if the results could be improved. If so, it contributes to reducing the required amount of run-to-failure examples. It was investigated with LSTM, which has performed the best in the previous prognostics experiment. Since the approach with adaptive piece-wise linear RUL labels were inaccurate, the traditional labels were used in these experiments. As mentioned in section 4.4.3, a model were trained that performed well on a much larger and popular dataset within prognostics research, called PHM08 (see section 4.2.2). The model was built to match the LSTM architecture that performed well in section 7.1, making it easier to use transfer learning.

The architecture and parameters of the LSTM model on the PHM08 dataset were decided based on experience and manual experiments. The best results were achieved using a model consisting of four LSTM-layers and two dense-layers. The number of neurons in each layer is described in table 7.8. The same setup as for the experiments in section 7.1 was used to train the model. This means that the labels were normalized between -1 and 1, the time window was 20, and activation functions for all layers were tanh. Since the number of inputs was not equal, the first LSTM-layer of the PHM08-model was not re-used.

Table 7.8: Architecture for the transferred model

| Layer | Units |
|---|---|
| LSTM layer 1 | 20 |
| LSTM layer 2 | 30 |
| LSTM layer 3 | 30 |
| LSTM layer 4 | 30 |
| Dense layer 1 | 20 |
| Dense layer 2 | 1 |

Transfer learning typically means to take parts of another trained network and re-use it in a new network either with untrainable or trainable layers. Several different architectures were explored. The architectures were based on:

- Re-using parts of the model, but make the layers untrainable.
- Re-using parts of the model, but make the layers trainable.
- Combining both untrainable and trainable layers.

In order to explore if transfer learning can contribute to improving the RUL predictions on the air compressor, several architectures were tested. The layers from the trained PHM08-model were used either untrainable or trainable. In the described architectures, a layer from the PHM08-model is referred to with *PHM-* plus the type of layer and the corresponding layer number (from table 7.8). An example is if the second LSTM layer is used, it is referred to as *PHM-LSTM-1*, while a new LSTM layer is simply referred to as LSTM.

Table 7.9, 7.10, 7.11 and 7.12 describes 8 different models that were tested to improve the predictions. Model 1, 2, and 5 uses several layers from the PHM08-model, but allows the transferred layers to be

trained. The pre-trained layers are in such cases used as a kind of weight initialization. The other models use a combination of new layers together with both trainable and untrainable layers. The reason some layers were untrainable is the idea that they might have been trained to find good and general features that the new model can benefit from.

Table 7.9: The first and second proposed model related to transfer learning

| Model 1 | # Nodes | Trainable | Model 2 | # Nodes | Trainable |
|---|---|---|---|---|---|
| LSTM | 20 | Yes | LSTM | 20 | Yes |
| PHM-LSTM-2 | 30 | Yes | LSTM | 30 | Yes |
| PHM-LSTM-4 | 30 | Yes | PHM-LSTM-4 | 30 | Yes |
| PHM-Dense-1 | 20 | Yes | PHM-Dense-1 | 20 | Yes |
| PHM-Dense-2 | 1 | Yes | Dense | 1 | Yes |

Table 7.10: The third and fourth proposed model related to transfer learning

| Model 3 | # Nodes | Trainable | Model 4 | # Nodes | Trainable |
|---|---|---|---|---|---|
| LSTM | 20 | Yes | LSTM | 20 | Yes |
| PHM-LSTM-2 | 30 | No | LSTM | 30 | Yes |
| PHM-LSTM-4 | 30 | No | PHM-LSTM-4 | 30 | No |
| Dense | 20 | Yes | Dense | 20 | Yes |
| Dense | 1 | Yes | Dense | 1 | Yes |

Table 7.11: The fifth and sixth proposed model related to transfer learning

| Model 5 | # Nodes | Trainable | Model 6 | # Nodes | Trainable |
|---|---|---|---|---|---|
| LSTM | 20 | Yes | LSTM | 20 | Yes |
| PHM-LSTM-2 | 30 | Yes | PHM-LSTM-2 | 30 | No |
| PHM-LSTM-4 | 30 | Yes | PHM-LSTM-3 | 30 | No |
| PHM-Dense-1 | 20 | Yes | PHM-LSTM-4 | 20 | Yes |
| Dense | 1 | Yes | PHM-Dense-1 | 20 | Yes |
| | | | Dense | 1 | Yes |

Table 7.12: The seventh and eight proposed model related to transfer learning

| Model 7 | # Nodes | Trainable | Model 8 | # Nodes | Trainable |
|---|---|---|---|---|---|
| LSTM | 20 | Yes | LSTM | 20 | Yes |
| PHM-LSTM-2 | 30 | No | PHM-LSTM-2 | 30 | No |
| PHM-LSTM-3 | 30 | No | PHM-LSTM-3 | 30 | Yes |
| PHM-LSTM-4 | 30 | No | PHM-LSTM-4 | 30 | Yes |
| PHM-Dense-1 | 20 | Yes | PHM-Dense-1 | 20 | Yes |
| Dense | 1 | Yes | Dense | 1 | Yes |

All the stated models were trained with the RMSProp optimizer with a learning rate of 0.00005. A batch size of 40 was used, and the models were trained over 40 epochs. The time window was

selected to be 20 time units. Results were evaluated based on MAE averaged over the 7 splits in k-fold cross-validation. Table 7.13 shows the score of each of the transfer learning models and the best model achieved in section 7.1, referred to as *original best*. The models that performed better than the original best are highlighted in the table.

Table 7.13: Results from RUL predictions with transfer learning models

| **Model** | **MAE** |
|---|---|
| Original best | 6.87 |
| Model 1 | 9.81 |
| Model 2 | 8.22 |
| Model 3 | 9.58 |
| **Model 4** | **6.78** |
| Model 5 | 8.41 |
| **Model 6** | **6.14** |
| Model 7 | 7.45 |
| **Model 8** | **5.92** |

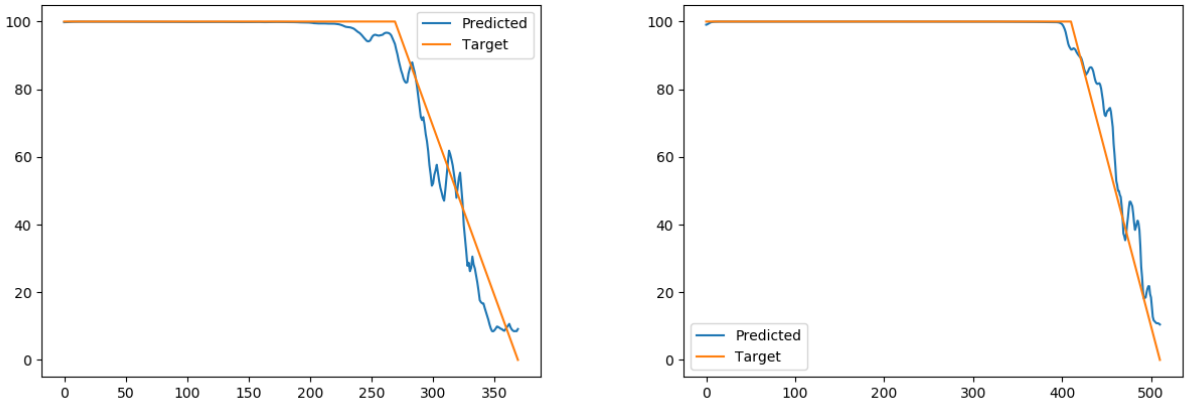
The results showed that three of the models performed better than the best model from section 7.1. Model 4 performed quite similar to it, while model 6 and 8 performed much better. These three models have in common that they have at least one untrainable layer from the PHM08-model. Model 6 and 8 also used several trainable layers from the PHM08-model. The difference between the two best models was that model 6 has two untrainable layers, while model 8 only has one. This indicated that having untrainable layers might force the network to reconstruct and benefit from good features in the transferred model.

The result from the best model (#8) for each individual split is presented in table 7.14. Compared to the original model, the transfer learning model performed similar for most of the splits, but there were large differences in split 2 and 3. The MAE was reduced a lot for split 3, while it increased a lot for split 2. It hard to identify why this happens, but it might indicate that the original model was over-fitted to some degree, while the transfer learning model generalizes better.

Table 7.14: Results from RUL predictions on each split with model 8

| | **MAE** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Split 1** | **Split 2** | **Split 3** | **Split 4** | **Split 5** | **Split 6** | **Split 7** |
| **Model 8** | 3.43 | 11.3 | 7.57 | 3.24 | 5.17 | 3.96 | 6.77 |

Figure 7.12a shows the prediction from model #8 on a sequence with fault type A from split 7. Figure 7.12b shows the prediction on a sequence with fault B from split 0. The predictions were following the target relatively good.




(a) Fault type A

(b) Fault type B

Figure 7.12: RUL predictions from transfer learning model

Split 3 resulted in quite bad predictions on the original model, but the best transfer learning model performed a lot better. Figure 7.13 shows the RUL prediction on the sequence with fault A from split 3. It shows that it was a lot better than the original predictions, but struggled to predict accurately close to end-of-life.

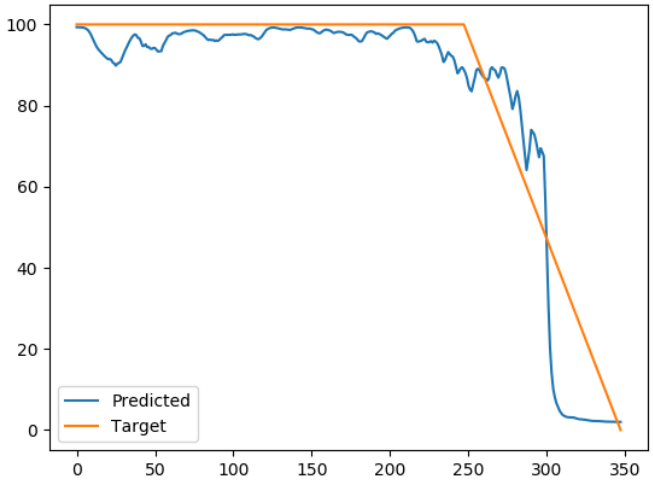


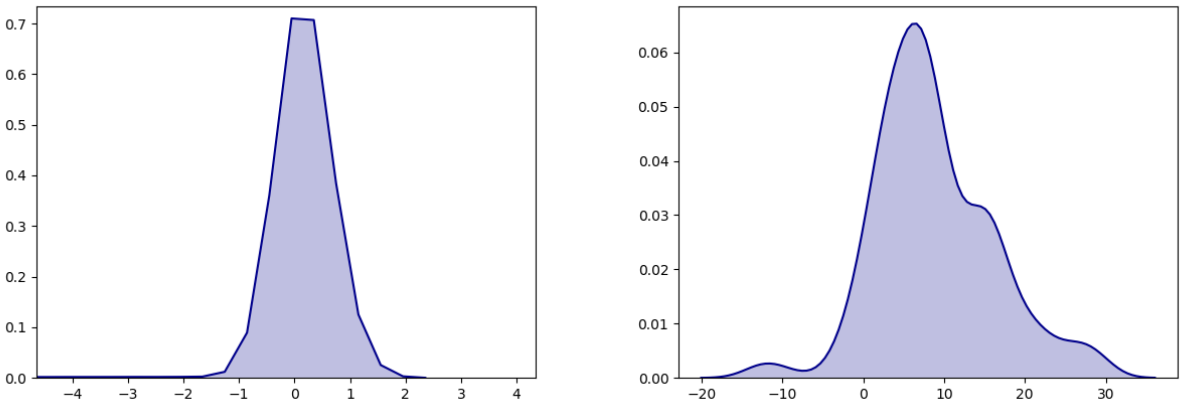

Figure 7.13: RUL predictions from transfer learning model on split 3

The results proved that using a model that was trained on sequences from a different system can contribute to improving predictions. Transfer learning in prognostics is promising and should be explored further.

## 7.3 Uncertainty

It is important to be certain that the predictions are correct and reliable. Companies might find it hard to trust a model giving an output like the RUL. In this section, a method for obtaining uncertainty on such predictions are proposed. This thesis has few run-to-failure sequences available, and the predictions so far have shown variable results on the available sequences. On one sequence a model can predict RUL to be 50, while in fact, it is around 30 (figure 7.5b). On another sequence, the RUL is predicted almost perfectly (figure 7.4a). When a potential user looks at RUL predictions, only the current and previous predictions are accessible. Since the prediction actually can vary a lot, the uncertainty can help to increase the trust in the predictions and show a more realistic picture.

The idea of the proposed method for obtaining uncertainty is that the prediction range can be divided into, for instance, 20 zones. A zone can be all predictions between 95 and 90, 90 and 85, and so on. For each zone, all predictions within it are compared to the target, and the error is stored. Each zone will eventually have a lot off errors stored, and a normal distribution of the errors is assumed. Therefore, predictions can be presented with not only a number representing the RUL, but also a probability distribution indicating the likelihood of the predictions. The distribution can also indicate if the current prediction is likely over- or under-estimating the RUL.




(a) Between 98 and 100

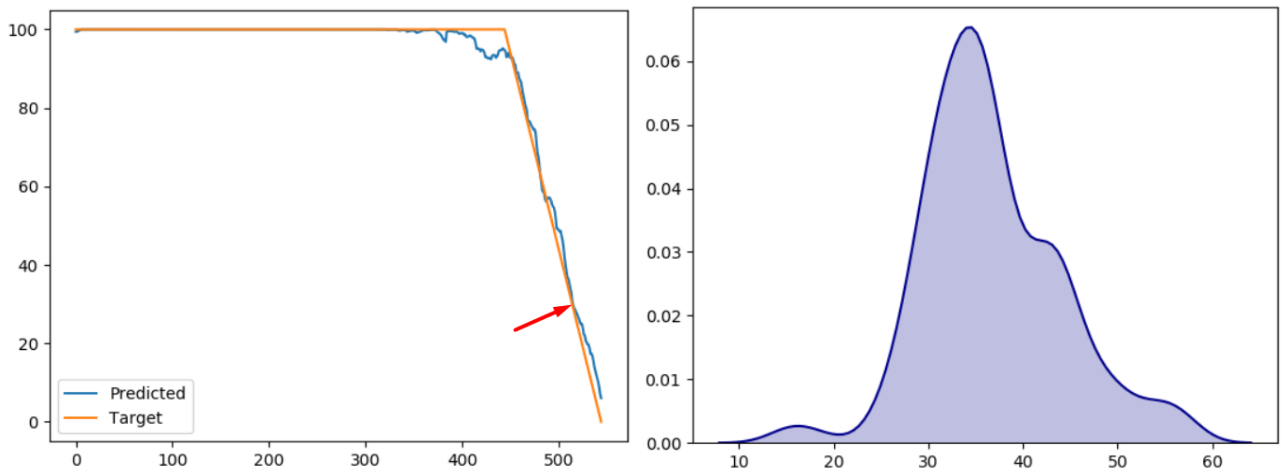
(b) Between 25 and 30

Figure 7.14: Error distributions from predictions in certain ranges

In the results so far, the accuracy has varied a lot between the individual splits. Only five of the best splits were chosen (1, 2, 4, 5, and 6) to show the principle of the uncertainty method. These were used for generating the distributions of the errors in each zone. It was used 20 zones as an example, they were equally distributed, except for the first two zones, which were between $[98, 100]$ and $[95, 98]$. The remaining zones were of range 5. Figure 7.14a shows the obtained error distribution in zone 1, which was for predictions between 98 and 100. The distribution shows that in general the predictions have an error of 0, but can miss slightly above and below the target. Figure 7.14b shows the error distribution of predictions between 25 and 30. This shows that the predictions in this zone commonly

under-estimated the RUL since the error often was above 0. It rarely over-estimated the RUL. This means that even though the prediction was perfectly fitting the label, the uncertainty of the prediction were large.

One sample from sequence A in split 6 was used to show a more specific example. Figure 7.15a marks a sample where the prediction is 28, which also is the target. This allows for combining the prediction of 28, with the probability distribution made from errors on predictions between 25 and 30. Figure 7.15b adds 28 to the distribution, hence, it shows the probability of the actual RUL based on previous predictions. The figure shows that based on historical predictions, the current prediction was likely to under-estimate the actual RUL. The figure indicates that when a prediction is 28, the actual RUL was commonly around 32-36.




(a) Sample prediction

(b) Error distribution from sample

Figure 7.15: RUL prediction with corresponding error distribution

In order to show the uncertainty on the same figures as the predictions, quantiles can be used. Quantiles are cut points that divide a probability distribution into intervals of equal probabilities. It is possible to select the number of quantiles and which quantiles based on the desired behaviour. For this case, quantiles based on the following percentages on both sides of the distribution were selected: 40%, 20%, 5%, 1%, 0.1%, and 0.01%. In addition, the maximum difference was used. The obtained quantiles were used to color the area around the prediction. A stronger color means a more probable prediction; weaker color means unlikely. The lightest limit is the maximum error. This is illustrated on a normal distribution in figure 7.16, where the color is darker closer to the average, which is marked with red.

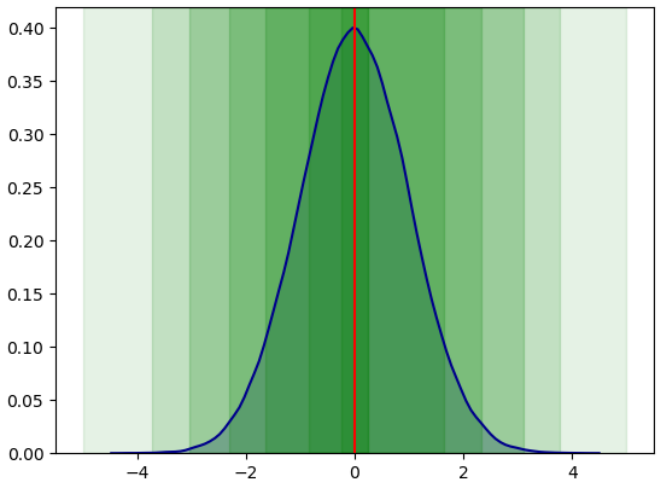
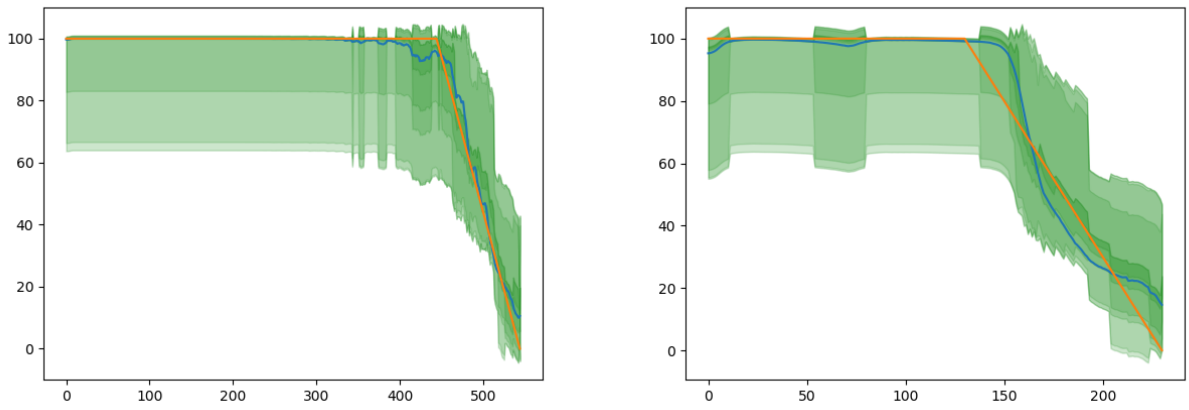



Figure 7.16: Normal distribution with quantiles marked by color

The principle with quantiles and coloring was used on a RUL prediction on sequence A in split 5, in figure 7.17a. The predictions were already accurate, but the uncertainty bounds add confidence to the predictions. The corresponding coloring shows that the prediction stays more or less in the zone of most confidence. The uncertainty bounds indicate that the air compressor might have longer or shorter RUL than the prediction itself indicates. When the prediction was around 20, the certainty bounds indicated that the prediction was most likely correct, but it could be longer until it fails. Figure 7.17b shows a prediction on a sequence from split 2. The prediction itself was less accurate than the previous. The colored area gives more confidence to the prediction. In most parts of the prediction, the darkest (most probable) area were close to the target.




(a) Fault type A from split 5

(b) Fault type B from split 2

Figure 7.17: RUL prediction with associated uncertainty bounds

In this section, a data-driven method for obtaining uncertainty related to RUL predictions have been proposed. Two alternatives for presenting the predictions with the associated uncertainties have also been proposed. The first alternative uses the prediction and corresponding probability distributions. The second alternative uses the prediction plot with colored quantiles to indicate the uncertainty. It can be tuned by adjusting the zones, the number of quantiles, and which quantiles. More data will make it more reliable and representable error distributions. The results and proposed methods are discussed in chapter 8.

# Chapter 8

# Discussion

In this chapter, the results from anomaly detection, diagnostics and prognostics are discussed. In addition, the maintenance strategy for air compressors and available data are evaluated and discussed. Appendix B presents a demo of how an air compressor supplier could use the explored cases for a PHM system. The demo is web-based and presents information about air compressors based on anomaly detection, diagnostics and prognostics.

## 8.1 Case A: Anomaly Detection

The anomaly detection case showed that it is possible to detect anomalous behaviour in air compressors. The case explored six different DL methods for two different purposes. First, to detect how much an air compressor deviates from normal condition. Secondly, to detect faults in historical data. Both were based on the reconstruction error. All six DL techniques (AE, SAE, VAE, DBN, ED-LSTM, ED-CNN) were able to reconstruct normal data with a relative low error. The reconstruction error increased when faults were introduced, and as they progressed. When the models were tested on sequences with faults, the reconstruction error started at a similar level as normal sequences, but increased as desired. The main difference between the models was the level of noise in the reconstruction error. Results from DBN had a lot of noise, while LSTM and CNN achieved results with little noise.

One of the main difficulties with the reconstruction-based approach was that it is challenging to tune. If a model achieves perfect reconstruction of normal data, it does not give insight into the performance on data with faults. If a hyper-parameter optimization loop is used uncritical for such a problem, a risk is that the obtained architecture allows the input to be copied to the output. This will achieve almost perfect reconstruction on both normal and fault data, which is not the desired behaviour. The architectures and hyper-parameters were therefore explored through trial and error. The manual process tested several architectures until the desired behaviour was achieved. If the process is to be automated a good evaluation method must be found.

### 8.1.1 Online: Anomaly score

In section 5.3, anomaly detection was explored towards giving a descriptive range of how much an air compressor deviates from normal operating condition. The proposed method transformed the raw reconstructions into a more descriptive range, referred to as the anomaly score. The reconstruction error was in different ranges for each of the DL models. It was therefore anticipated to be hard to find a common and descriptive scale of the reconstruction error. The proposed transformation contained three steps (scale, sigmoid transformation, scale), which was tuned based on a configuration set. It proved to give a suitable scale, giving an anomaly score which ranged from 0 (normal condition) to 100 (close to failure). The drawback with the approach is that it requires manual tuning, and is sensitive to changes. If little data is available to configure the transformation, an alternative is to use the reconstruction errors directly. The raw reconstruction error and its historical values can help to indicate anomalous behaviour. If it is increasing, it can indicate that there is something wrong with the system. The advantage of the anomaly score is that a single value is much more descriptive than the raw reconstruction error.

The range $[0, 100]$ was divided into three zones (normal, warning, and danger) to get more information from the anomaly score. These zones were tuned manually to classify normal data in the normal zone, and data with faults in the danger zone. While manually tuning such thresholds is a drawback, the advantage is the flexibility of tuning it to the desired behaviour. A potential user can decide if it is desired to get early warnings or wait until the anomalous behaviour is more guaranteed. The proposed anomaly score approach is not intended to be used purely as a classifier. The thresholds can, on the other hand, be useful for giving warnings and error.

One of the problems with the proposed anomaly score approach is that it is hard to evaluate. The models were therefore evaluated in two ways; visually and based on classifications from the thresholds. All evaluation was done on unseen data with four different fault types. Two of the fault types were not represented in either training or configuration. Visual inspection indicated that the anomaly score obtained from LSTM, CNN and VAE was the most promising. They were able to give a clear indication that the air compressor deviated from normal operational condition in reasonable time before failure. The three remaining models performed worse and struggled to give clear and consistent results. The classification was done on a set of partly randomly selected samples and supported that LSTM, CNN and VAE performed the best. Both VAE and LSTM achieved 100% accuracy. It is interesting that two out of the three best performing models use time windows, while the three worst works with non-sequential data. The VAE differs from the other models by modelling probability. The results indicate that VAE and ED-architecture LSTM are great choices for anomaly detection on air compressors.

The proposed method for making the anomaly score more transparent proved to indicate why an air compressor deviates from expected behaviour. A unique pattern of top contributing sensors for each fault type was detected. Service personnel can based on the top contributing sensors identify what the potential fault is or at least which parts of the system are causing the deviation. Increased transparency in anomaly detection makes such a feature much more useful. The contributing sensors have little significance when the anomaly score is in the normal zone. The contributing sensors should only be investigated in combination with an anomaly score outside of the normal zone. It is essential

to evaluate the scalability of the proposed method. The air compressor explored in the thesis has few sensors, but the method is developed to be able to scale to larger systems with several sub-systems. In such cases, it is possible to build a tree-structure describing the relation between sensors. Each sub-system can have its own branch. The method can, in such cases, be tuned to indicate the top contributing sub-system and the contributing sensors within them.

It is important to consider if the proposed method for anomaly score with transparency is useful. It has already proved to detect that the air compressor is deviating from normal condition and indicate why it deviates. The greatest advantage of the approach is that it is only trained on data in normal operating condition. It is beneficial to have a few run-to-failure examples to tune the model. If these are not available, the reconstruction error can be used directly until the required data is collected. This means that a company can easily start with this feature in a PHM system without having large quantities of labelled data. In addition, the approach proved to detect anomalous behaviour for unseen types of faults. This indicates that compared to fault diagnostics (case B), this approach can detect that something is wrong in the system, without having historical examples of that specified fault. A common problem in the industry, especially the maritime industry, is few run-to-failure examples and little labelled data in general. This approach can still be used in such cases. When more data is collected, the approach can be re-configured, improved, and evaluated. The anomaly score approach can answer the question: *"Is there something wrong with the air compressor?"*.

### 8.1.2 Offline: Fault detection

A sub-part of the anomaly detection case was to see if the time-step where a fault occurred could be detected, unsupervised. The suggested approach can be used for labelling data for diagnostics or prognostics, or it can be used to analyze data after a failure. The approach was evaluated based on how accurately the fault time-step can be detected. The obtained fault time-steps was used for experimenting with an alternative labelling approach for RUL predictions. This is discussed further in section 8.3.

The fault detection approach was explored with the reconstruction error from six different DL techniques. When evaluated on eight unseen sequences with faults, theVAE stood out with an impressive accuracy of 99.31%. Among the remaining models, only AE and CNN showed somewhat promising results, while the rest often missed the fault time-step with a lot. The VAE has performed best both in this case and the previous and are considered a great choice for both anomaly detection and fault detection.

The results indicated that where the reconstruction error was accelerating the fastest was where the fault happened. This might not be the case for all systems and faults, but it worked for this particular case. An alternative approach if the maximum acceleration is not sufficient, is to explore if a threshold for either acceleration or velocity of the reconstruction error can detect fault time-steps better. Such thresholds can, for instance, be based on the maximum acceleration and velocity in normal sequences.

## 8.2 Case B: Diagnostics

The diagnostics experiments from this thesis are not proposing anything new, but is included since it is an important part of a potential PHM system for air compressors. Two different diagnostics approaches were explored. The first showed that the three DL models were able to accurately able to identify faults. The second approach showed that an earlier indication of potential faults was possible by using the severity labels of the faults. In both approaches, the models performed quite similar, but FNN was slightly better than LSTM and CNN. Results on the severity prediction showed that the predictions were not able to predict the severity steps, but rather a gradually increasing severity increment from 0 to 1. This is a useful feature that can give an early indication of a fault that is progressing.

Both of the approaches are suitable to include in a PHM system for air compressors. The advantage of the severity prediction approach is that it gives an earlier indication of a fault that might progress. The disadvantage is that severity labels often are hard to obtain. It is assumed that for most cases, the fault identification methods is the most applicable. Both of the methods use activation functions which allows them to predict faults, even if several of them occur at the same time. Unfortunately, there was no data foundation to evaluate if several faults occurring at the same time could be detected. Both methods are easy to extend, by adding more fault types when historical data of them is available. Both approaches can answer if there is something wrong in the system and what is wrong.

Compared to the proposed method for anomaly detection, these experiments can more clearly identify what is wrong with the system. It is powerful to combine these features since diagnostics can recognize faults with historical examples, while anomaly detection can indicate faults without previous examples.

## 8.3 Case C: Prognostics

Prognostics is considered as one of the most important parts of a complete PHM system. In this thesis, several aspects of prognostics on air compressors were explored. The first parts looked at predicting RUL and comparing three DL techniques. A state-of-the-art labelling approach was also tested. Secondly, transfer learning was investigated to emphasize the typical problem of few run-to-failure examples. Finally, a data-driven approach for achieving uncertainty in RUL predictions was proposed.

### 8.3.1 RUL predictions

Prognostics was explored to predict RUL of air compressors. First, FNN, LSTM and CNN were compared to find which model was able to predict the RUL most accurately. In the diagnostics experiments, FNN was the best choice, but all three models performed quite similar. Prognostics had larger differences, where LSTM was the best, followed up by CNN. The time-based methods were superior to FNN. This indicates that capturing patterns across time is important in prognostics. The results were promising and obtained an average MAE below 7. Visual inspection of RUL predictions

also indicated promising results. The results from the individual splits showed that the performance was high for most splits, but one of the splits was deviating a lot. As mentioned earlier, this can suggest that the data in that split was collected under different conditions. Since this is relevant for both prognostics and diagnostics, this will be discussed further in section 8.4.

The best achieving model, LSTM, was used to explore the potential of the adaptive piece-wise linear RUL labelling approach. It is hard to compare the results between the two labelling approaches, but the original labelling achieved much lower MAE on the predictions. Visual inspection showed that for some sequences, the predictions followed the target accurately, while for others, it missed by a lot. One of the potential benefits from the adaptive approach is that it can predict different ranges in advance. For this particular problem that could be beneficial since failures due to fault type B, tend to fail faster than fault type A. The original research used this approach on a dataset where many sequences were available. This dataset might contain so few samples that the models were not able to generalize between the different constant levels. The alternative labelling approach should not be ruled out and should be tested further if more run-to-failure examples come available. The results proved that the common piece-wise linear RUL labelling approach achieved better results than the newer variant. The constant level of the piece-wise linear labels was chosen manually after some experiments. In the future, this level should be chosen based on how long in advance a failure should and could be predicted. Since the data contains failures that are forced faster than in real situations, the level must be reconsidered when data in a more realistic time scale is available.

The choice of using data augmentation to generate more sequences was based on having few run-to-failure sequences compared to the popular C-MAPSS dataset. Initial experiments showed improved performance when generating more sequences. Therefore, data augmentation was chosen to use, but in in the future the data augmentation in prognostics should be explored more thoroughly, and the performance with and without augmentation should be compared.

Architectures and hyper-parameters in both diagnostics and prognostics experiments were found by using a combination of manual experiments and PSO. The manual experiments aimed to narrow down the search space by finding a good number of layers for each model and the approximate layer sizes. Next, the PSO was used for tuning the hyper-parameters. The tuning process was not compared with other hyper-parameter optimization techniques such as grid search or GA. Therefore, there is no foundation for comparing PSO with other methods for hyper-parameter optimization. The approach did its intended job and was able to find parameters that achieved good results. The optimization loops were done by training on 5 datasets, validating on 1 and testing on 1. Optimally, the process should have used an approach similar to the k-fold cross validation used for evaluating the models. The current tuning process might risk to over-fit performance based on a single split. It was considered to implement full-scale optimization, but it was found to be too time-consuming to execute. In future projects, it should be considered to be included.

The failures in the available data are as mentioned in section 4.2.1 forced in an unnatural speed. Hence, the models from this research cannot be used directly when more realistic data is acquired. The results do indicate that LSTM is the most promising method and that the RUL can be predicted accurately. The patterns in the data are assumed to be quite similar, but forced faster than in the real fault situations. The principle of predicting the RUL on air compressors is important and can be

a part of a future PHM system. The prediction can help decide when maintenance should be done.

### 8.3.2 Transfer learning

As mentioned earlier, a common problem in prognostics is having few run-to-failure examples. Transfer learning was investigated to see if it had the potential to emphasize this problem. It has been explored and proved useful in many cases for image recognition, but it has not been researched much in prognostics. 8 different model architectures were explored to see if transfer learning was applicable in prognostics. Each model had one or more layer transferred from a pre-trained model from another dataset. Three of the models were able to improve the results, compared to the initial results. Model 4 performed quite similarly to the previously achieved results, while model 6 and 8 improved the results more. Both these models used four layers from the pre-trained model. Model 6 has the first layer untrainable, while model 8 has the two first untrainable. The models also used a new LSTM input-layer and a new Dense output-layer. A theory is that these models are performing better since they can keep some important feature detection from the pre-trained model, so the network can focus on learning to predict an accurate RUL based on those features. The model with only one untrainable layer performed the best. The results indicate that using transfer learning in prognostics can improve the results. This can have the impact that prognostics require fewer run-to-failure examples. This can make it easier for companies to start working towards prognostics.

One interesting difference between the result with and without transfer learning was the performance on the individual splits. The best transfer learning model performed a lot better on split 3 than the original models. It is hard to state why, but it might be that training the network on other, related data first, helps to generalize better. It might be that the models can learn more general features and therefore perform better on average.

In image recognition, it is often discussed which features are learned in a model, and transformed to another during transfer learning. Many suggest this is features for extracting information about vertical and horizontal edges, among other things. In prognostics, it harder to imagine what these features can be. It can be speculated if it learns some simple features that capture how the values changes or degrades during the time window. Transfer learning in prognostics should be investigated further in more specified research.

### 8.3.3 Uncertainty

In other problems like image recognition, predictions often have an accompanying probability which indicates how certain the predictions are. In prognostics with DL, a prediction is typically a single-valued prediction, and no information about the certainty is given. A single-valued prediction can give an illusion of certainty. Providing uncertainty bounds presents more realistic predictions. The results in section 7.1 showed that the accuracy of the RUL predictions can vary a lot. A challenge is to convince service personnel and customers that they can trust the predictions. Therefore, investigating how uncertainty bounds around RUL predictions can be obtained is both useful and desired.

The proposed method for finding uncertainty bounds around RUL predictions is data-driven and based

on finding how predictions commonly miss. The approach gave insight into the certainty of predictions and can help to make more confident decisions. One of the advantages of the method is that it is data-driven, meaning it can be learned from data and will become more accurate as more data is available. As more predictions are made, the distributions are more representative. Introducing such uncertainty bounds can hopefully make it easier to trust DL-based predictions. Even if a prediction is inaccurate, a decision taker can see if the air compressor is about to fail based on previous experience obtained from the data. Presenting more than a single-valued RUL prediction to a user is beneficial and increases the insight into the predictions. The proposed method seems promising and should be explored further when more data is available.

## 8.4 Data

As mentioned previously, the data in this thesis is based on faults generated in a much shorter time interval than in realistic cases with deployed compressors. The principle of the individual cases are still very relevant, but parts of it should be explored again when more realistic data is available. The faults introduced are realistic, but forced in a faster time horizon, meaning that the patterns are probably similar in more realistic cases, but stretched over a larger time window. All experiments so far have proved that they can find the patterns and make accurate predictions. Still, it is hard to say anything about how the performance will be compared to more realistic data.

Collecting data for PHM experiments can be a tedious process. Systems like air compressors can run for many years before any faults or failures occur. Therefore, Sperre Industri AS wanted to conduct this initial research with semi-realistic data to see the potential of PHM on their products. An alternative to only collecting real historical data is to combine it with a model-based approach. If an accurate simulator model of their compressor is developed, it could be possible to start collecting data from it. The data can contain vital information and capture close to real data. It might be able to give a great foundation for making predictions. When combined with real data, it can be explored if data from a potential simulator model can improve the results. It can also help to reduce the need for many run-to-failure examples.

Among the researched topics in this thesis, anomaly detection is the part that is easiest to implement before enough realistic data is collected. It needs no run-to-failure examples and can be continuously improved as more data becomes available.

Results from diagnostics and prognostics experiments have indicated that the performance has been quite similar on all sequences, except sequences in split #3 in the k-fold cross-validation. No clear reason for why the performance on those sequences was deviating from the rest has been found. The results do, on the other hand, indicate that there is something special with at least one of those sequences. It could have been collected over very different surrounding conditions, or something special could have happened with the air compressor for that particular sequence. The fact that the results differed so much for those sequences indicates that too little data is available, or at least to little variation in the data. It should be considered to collect several more sequences to see if the general performance of the predictions can be improved.

## 8.5 PHM for air compressors

The researched topics in this thesis can based on DL answer the three important questions about air compressors. Anomaly detection can answer if there is something wrong with the system. Diagnostics can say what is wrong. Finally, prognostics can say how long until the air compressor fails. These three aspects are important and useful for a PHM system and can improve the current maintenance strategy on air compressors.

It is hard to evaluate maintenance strategies for all air compressors and suppliers, but as the literature review indicated, most air compressor suppliers follow corrective and traditional preventive maintenance strategies. Using a maintenance strategy based on PHM with DL can improve these strategies. Most suppliers wait until a system fails to do maintenance and repairs. Prognostics with RUL predictions combined with diagnostics can know in advance that the system is degrading and why. This makes it possible to prevent unexpected standstills and initiate required maintenance actions before the system fails. Several of the suppliers change certain parts based on experience and mean-time-to-failure curves. This can lead to changing parts that show no sign of degradation, or being too late to change a part because it failed earlier than expected. Using a combination of prognostics and diagnostics can avoid changing parts too early since it is continuously monitoring the system and can indicate that the system is still working as expected. It will also avoid cases where parts are changed too late by capturing the actual condition of the compressor. A PHM system leads to better insight into the condition of the air compressor. It can monitor the system online and give warning that a compressor is starting to degrade, deviating from normal condition or a fault occurs. It can remotely give warnings to operators and stop compressor before any parts are broken. This might lead to stopping a progressing fault before it goes so far that it cannot be reversed. This will lead to saving money on spare parts and having a more reliable system with few unexpected standstills.

One example of how such a PHM system can improve the current maintenance strategy is by looking at Sperre Industri AS. Their service agreement offers customers replacement parts within 48 hours of a failure. This means that when something fails, a lot of resources are used on sending parts and fixing it. The customer can also experience an unexpected standstill due to the failure. If a PHM system could predict that a part will fail, they can send necessary replacement parts before the compressor fails. This can lead to preventing unexpected standstills and increasing reliability.

Most of today's maintenance strategies contain other types of routine work such as visual inspection, checking the oil level, and changing the oil. While a PHM system with diagnostics probably can detect low oil level or time for changing oil, these situations can be avoided by simply monitoring signals of oil quality and level. A PHM system can and should be able to detect degradation due to low oil level or bad oil. This with the presumption that historical data of such events are available. Anomaly detection could also indicate that the system is deviating from normal condition and for instance, show that oil quality is the main reason for the deviation.

A PHM system can improve the current maintenance strategy on air compressors. This thesis focused on the parts revolved around anomaly detection, diagnostics and prognostics. Data acquisition and decision support are two other important contents of a PHM system, which is out of the scope of this thesis. Therefore future work should investigate further how results from the three cases can be

transformed into useful decision support.

The experiments in this thesis focused purely on PHM based on DL. It is important to evaluate related advantages and disadvantages. Compared to traditional ML, the research done in this thesis is performed without doing any manual work related to feature engineering. The reconstruction error obtained from the anomaly detection has been added as a feature, but it is automatically obtained from the anomaly detection model. This means that using DL can reduce the required manual work. It also makes it less application dependent than PHM with more traditional methods. A disadvantage of using DL and other data-driven approaches to PHM is that it usually requires large amounts of historical data. This is often challenging to obtain, especially with labels. Model-based approaches have the benefit that they require no historical data, and can be developed only based on domain knowledge. In return, they are often time-consuming to develop. If a company has historical data available, the foundation for working towards PHM with DL is better. Even with unlabelled data, DL can be applied by starting with anomaly detection. One benefit of applying DL is that the results can improve over time. The product can be improved and extended as more data becomes available.

Suppliers of products like air compressors, often have several different variants of the product. In this thesis, only one of several potential products were explored. It is important to investigate the process of using DL-based PHM on a range of products. In some cases, the products might be similar. Therefore little data could be needed to re-train the models. This should be investigated further when data from several products are available. An advantage of using DL is that for other products, it should be enough to re-train and configure the models with the related data. Again, having enough historical data from the products might be challenging. If a model-based approach were applied for a range of products, much manual work would be necessary to develop an accurate model for each product. Developing accurate degradation models can be tedious and challenging. In cases where the products are similar, it can be possible to re-use parts of a model, which can reduce the required work.

# Chapter 9

# Conclusion

This research aimed to explore how DL can be used for a PHM system for air compressors. Based on experiments with anomaly detection, diagnostics and prognostics, it can be concluded that it has the potential to improve current maintenance strategies on air compressors. The three cases studied in the thesis complement each other and contributes to a powerful set of features. Research on prognostics showed that it is possible to predict when an air compressor will fail. Diagnostics showed that it is possible to identify faults and their severity. Anomaly detection proved that it is possible to get information about how much a system deviates from normal operating condition. This indicates the current health of the air compressor and why the behaviour is unexpected. A typical problem in the maritime industry is the lack of labelled data. This makes the anomaly detection a powerful method that can be used to obtain critical information about the system, even with a small data foundation. Both diagnostics and prognostics require labelled examples of faults and run-to-failure. The unsupervised approach for detecting faults can be used for obtaining labels.

All the proposed methods can work together to give valuable insight into the condition of an air compressor. The explored cases give a foundation where a company can start by implementing anomaly detection, and extend with diagnostics and prognostics features as more data is acquired. A demo was presented in appendix B, as a proof of concept of a PHM solution for the collaborating company's air compressors. The researched cases are implemented in the demo. Anomaly detection, diagnostics and prognostics can answer the three important questions about a system: *"Is everything going fine?"*, *"If not, what is wrong?"* and *"If something is wrong, when will it fail?"*. A PHM system as investigated in this thesis, can contribute to having the required maintenance strategy on vital equipment on board an unmanned autonomous ship. Next, the four research questions investigated in this thesis are repeated and answered.

**RQ1:** How can DL be used to detect abnormal behavior in air compressor systems?

Detecting abnormal behaviour in the air compressor system was done by exploring six different DL methods based on an ED-principle. The models were used to try to reconstruct raw input data through a lower dimensional latent space. The proposed method transformed the reconstruction error from the models into a descriptive range referred to as the anomaly score. Visual inspection showed

that the anomaly score from VAE and LSTM was performing best. They were able to give a strong indication of how much the air compressor deviated from normal operational condition. Both models were able to indicate highly anomalous behaviour in advance of failures. The same two models proved to accurately identify if a compressor was in normal or faulty operating condition. A method based on error contribution was proposed to increase the transparency of the anomaly score. It led to the anomaly score not only indicating how much the system deviates, but also which parts of the system are contributing to the deviation.

A newly proposed method for detecting faults in historical data with an unsupervised approach was also explored. The method is based on finding the maximum acceleration of the reconstruction error. When applied to the reconstruction error from six different DL techniques, the VAE achieved the most accurate results. The method detected faults with an accuracy of above 99%. This approach can be useful to obtain labels, which typically can be challenging to obtain.

**RQ2:** How can DL be used to identify faults in air compressor systems?

The diagnostics experiments presented two alternative approaches to identify faults in air compressors. The first approach was able to identify faults accurately. The second approach was able to identify the faults and predict their severity. The results indicated that predictions from FNN, LSTM and CNN performed quite similar. It is recommended to use the severity approach if the required labels are available. It can give an earlier indication of any potential problems. Severity labels are typically hard to obtain. Therefore, the fault identification is considered more applicable. The diagnostics experiments have little contribution to the overall research area of PHM, but are an important feature in a potential PHM system for air compressors. It was included to show potential usage when historical data is available.

**RQ3:** How can DL be used to predict the remaining time until failure, and how to emphasize the typical problem of few run-to-failure examples?

Three different topics were explored to answer this research question. First, experiments showed that LSTM was able to predict the remaining time until failure more accurate than CNN and FNN. The predictions achieved a MAE of 6.87, which means it on average misses the RUL with 6.87 time units. The results prove that LSTM can be considered a promising DL technique for predicting RUL. A newly proposed labelling approach was also explored with LSTM. The predictions gave variable results and did not perform satisfactorily. The approach should be explored further when more data is available.

Transfer learning in prognostics was explored to emphasize the typical problem of few run-to-failure examples. The popular C-MAPSS dataset was used to train the transferred model. Eight different architectures were proposed based on a combination of new and transferred layers. Three of these models were able to improve the predictions, and the best model reduced the MAE down to 5.92. Visual inspection showed that the predictions were accurately able to predict the time until the air compressor failed. Results from transfer learning were promising and proved to be useful in improving predictions and making the model generalize better. The predictions can, in other words, be improved without having more run-to-failure examples.

Predictions from ML typical give a single-valued output which gives an illusion of certainty. Results from the prognostics experiments showed that the accuracy from RUL prediction varied between sequences. A method for obtaining uncertainty was proposed to give service personnel a stronger foundation to make decisions. The method is a data-driven approach that finds uncertainty from historical predictions. It was promising to give more realistic predictions, which can lead to more qualified maintenance decisions.

**RQ4:** What are the advantages and disadvantages of using DL in a PHM system, and how does it improve the current maintenance strategy on air compressors?

Section 8.5 discussed the advantages and disadvantages of using DL for PHM for air compressors. The main disadvantage of using data-driven approaches is the need for data. To reduce this problem, parts of the thesis have focused on approaches that work with little data. Anomaly detection is less dependent on labelled data and does not require run-to-failure examples. In cases where little labelled data is available, it could be possible to start with an anomaly detection feature. The system can be extended with diagnostics and prognostics features as more data becomes available. Transfer learning seemed promising to reduce the need for these examples in RUL predictions. Some advantages of using DL is that it can reduce manual labor, find more complex patterns in data, and improve over time. Compared to model-based approaches, DL can learn the necessary patterns directly from data. DL is less application dependent than more traditional ML methods which often require much manual labor in the form of feature engineering.

Related work and research from this thesis have indicated that PHM with DL can improve the current maintenance strategy on air compressors. Instead of being oblivious to the state of the system, the methods within the individual cases can provide much information about the health of an air compressor. It can, therefore, contribute to deciding the required maintenance actions. Rather than facing unexpected standstills, a PHM system has the capability of predicting when something will fail and why. Maintenance actions can be performed before a failure, and parts can be sent in advance of failures. In addition, a PHM system can potentially avoid changing parts too early, since the parts can be changed when required instead of before. Next, the main contributions of this thesis are stated. The chapter ends with suggestions for future work.

## 9.1 Contribution

PHM is a much researched topic and of great interest to many industries and companies. This thesis is unique in the sense of its comprehensive exploration directed towards PHM for air compressors with DL. This research combined results from anomaly detection, diagnostics, and prognostics into a PHM demo for air compressors (Appendix B). Besides, several of the concepts proposed are general and can be adapted to other systems and industries. The main contributions are:

- Proposed a method for giving an informative scale to the reconstruction error referred to as the anomaly score. It provides descriptive information about how much a system deviates from

normal operating condition. The proposed method for increasing the transparency in anomaly scores gives insight into why a system deviates by indicating which parts of a system contributes to the deviation.

- Predicting RUL is much researched. This thesis explored the newly proposed labelling approach referred to as adaptive piece-wise linear RUL. The results were not satisfactory, and the findings indicate that the labelling approach requires more run-to-failure examples than the traditional piece-wise linear labels.
- Results indicate that the principle of transfer learning can be adapted to prognostics and RUL predictions. The approach was able to improve results where few run-to-failure examples are available. Models with one or two transferred, untrainable layers in combination with transferred, but trainable layers, can improve RUL predictions.
- Proposed a data-driven approach for obtaining uncertainty in RUL predictions based on errors from historical predictions.

## 9.2 Future work

The work in this thesis has presented proof of concept that DL has the potential to improve current maintenance strategies on air compressors. The list below presents potential future work related to this master thesis and its topic.

- The results in this thesis are based on data where faults have been forced and not degraded naturally. It is necessary to collect more **realistic data** to evaluate how the methods perform in more realistic situations. Future work could also see if building a **simulator model** of air compressors to collect data from, can reduce the requirement of realistic data. The combination of using semi-realistic, simulated, and realistic data is interesting. It can be beneficial if a simple simulator model can reduce the requirement of run-to-failure examples.
- The proposed method for detecting abnormal behaviour performed the best with VAE and LSTM. The method based on LSTM tries to reconstruct not only a set of sensors, but a set of sensors for a given time horizon. A potential future problem is if the system has faults that occur over quite **different time intervals**. This means that some abnormal behaviour only can be detected in a scale of a few seconds, while others are based on several hours. This should be investigated further.
- The transparency method proved that it could help to indicate where a potential fault is by finding the sensors which contribute the most to the reconstruction error. The air compressor system explored in this thesis has only 14 sensors. This makes it easy to indicate a connection between sensors and faults. The **scalability** of this method should be investigated further. A suggestion when the method is used for larger systems is to group sensor into a tree structure based on which parts they are related to. The method can then, instead of finding the top contributing sensors, find the top contributing parts and within them the top contributing sensors. This should be tested on an appropriate system.

- The **adaptive piece-wise linear RUL** did not perform satisfactorily on data from this thesis. The principle of predicting faults in different time horizons in advance can be beneficial. Therefore, it is suggested to investigate if the method performs better when more run-to-failure examples are available.
- This thesis proved that **transfer learning** has potential in prognostics and RUL predictions. The principle should be investigated further. In other types of problems such as image recognition, it is often speculated in what behaviour or functionality is being transferred in transfer learning. This should be investigated in prognostics as well. A question to investigate is what kind of features are being extracted through the untrainable, transferred layers.
- The proposed method for obtaining **uncertainty** is suggested to explore further. It currently seems promising to give a more realistic impression of the uncertainties involved in RUL predictions. It can be investigated if more data can provide smoother results, which is easier to understand for a potential user. It would be both educational and interesting to perform a study of how service personnel or potential users of such an application would react on RUL prediction with and without uncertainty.
- A really important aspect to PHM system is **decision support**. It has not been investigated in this thesis, but is suggested as important in future work. It should be explored how the results from anomaly detection, diagnostics and prognostics could give decision support or optimally automated decisions.

# Chapter 10

# References


[1] B. Batalden, P. Leikanger, and P. Wide. "Towards autonomous maritime operations". In: *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)* (2017), pp. 1–6. DOI: 10.1109/CIVEMSA.2017.7995339.

[2] Rolls Royce. "AAWA Position Paper: Remote and Autonomous Ships - The next steps". In: (2016). URL: http://www.rolls-royce.com/~/media/Files/R/Rolls-Royce/documents/customers/marine/ship-intel/aawa-whitepaper-210616.pdf.

[3] R. Kothamasu, S. H. Huang, and W. H. Verduin. "System health monitoring and prognostics - A review of current paradigms and practices". In: *Handbook of Maintenance Management and Engineering* (2009), pp. 337–362. DOI: 10.1007/978-1-84882-472-0_14.

[4] B. Dhillon. *Engineering Maintenance: A Modern Approach*. 2002, pp. 1–222. ISBN: 978-1-58716-142-1. DOI: 10.1201/9781420031843.

[5] R. K. Mobley. "Impact of Maintenance". In: *Maintenance Fundamentals*. 2004. Chap. 1, pp. 1–10. ISBN: 978-0-7506-7798-1. DOI: 10.1016/B978-075067798-1/50022-4.

[6] T. M. Allen. *U.S. Navy Analysis of Submarine Maintenance Data and the Development of Age and Reliability Profiles*. Tech. rep. 2001. URL: https://pdfs.semanticscholar.org/27c8/71b845f0c05fee98d018da6346452a081dd9.pdf.

[7] F. Camci, G. S. Valentine, and K. Navarra. "Methodologies for integration of PHM systems with maintenance data BT - 2007 IEEE Aerospace Conference, March 3, 2007 - March 10, 2007". In: (2007), IEEE; AIAA. DOI: 10.1109/AERO.2007.352917.

[8] A. Ismail and W. Jung. "Recent Development of Automotive Prognostics". In: April (2015).

[9] J. Lee et al. "Prognostics and health management design for rotary machinery systems - Reviews, methodology and applications". In: *Mechanical Systems and Signal Processing* 42.1-2 (2014), pp. 314–334. DOI: 10.1016/j.ymssp.2013.06.004.

[10] O. Geramifard et al. "Data-driven approaches in health condition monitoring - A comparative study". In: *2010 8th IEEE International Conference on Control and Automation, ICCA 2010* (2010), pp. 1618–1622. DOI: 10.1109/ICCA.2010.5524339.

[11] S. Yin et al. "A review on basic data-driven approaches for industrial process monitoring". In: *IEEE Transactions on Industrial Electronics* 61.11 (2014), pp. 6414–6428. DOI: 10.1109/TIE.2014.2301773.

[12] A. L. Ellefsen et al. "A Comprehensive Survey of Prognostics and Health Management Based on Deep Learning for Autonomous Ships". In: *IEEE Transactions on Reliability* (2019), pp. 1–21. DOI: 10.1109/TR.2019.2907402.

[13] D. Li et al. "Anomaly Detection with Generative Adversarial Networks for Multivariate Time Series". In: (Sept. 2018). URL: http://arxiv.org/abs/1809.04758.

[14] W. Sun et al. "A sparse auto-encoder-based deep neural network approach for induction motor faults classification". In: *Measurement: Journal of the International Measurement Confederation* 89 (2016), pp. 171–178. DOI: 10.1016/j.measurement.2016.04.007.

[15] B. Huang et al. "Review of Data-Driven Prognostics and Health Management Techniques: Lessions Learned From Phm Data Challenge Competitions". In: *Machine Failure Prevention Technology 2017* May (2017), pp. 1–17. URL: http://www.mfpt.org/MFPT2017/MFPT2017Proceedings/Paper_Huang_YaunDi.pdf.

[16] SPERRE. *Technical guide: Your Life Cycle Partner in air compressors*. 2018. URL: http://www.sperre.com/assets/downloads/Sperre-Technical-Guide_20180416_web.pdf (visited on 11/26/2018).

[17] I. Arel, D. C. Rose, and T. P. Karnowski. "Deep Machine Learning - A new Frontier in Artificial Intelligence Research". In: *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE* 5 (2010), pp. 13–18. DOI: 10.1109/MCI.2010.938364.

[18] M. Längkvist, L. Karlsson, and A. Loutfi. "A review of unsupervised feature learning and deep learning for time-series modeling". In: *Pattern Recognition Letters* 42.1 (2014), pp. 11–24. DOI: 10.1016/j.patrec.2014.01.008.

[19] K. Goebel. *Prognostics and Health Management - Kai Goebel*. 2017. URL: https://www.youtube.com/watch?v=1P36k2QfauY (visited on 02/21/2019).

[20] Engineering Toolbox. *Types of Air Compressors*. 2003. URL: https://www.engineeringtoolbox.com/air-compressor-types-d_441.html (visited on 02/18/2019).

[21] Noria Corporation. *Reciprocating Compressor Basics*. 2005. URL: https://www.machinerylubrication.com/Read/775/reciprocating-compressor (visited on 02/18/2019).

[22] British-Standard-Institution. *BSI-EN-13306:2010*. 2010.

[23] N. Zerhouni et al. "Prognostics and Health Management for Maintenance Practitioners-Review, Implementation and Tools Evaluation". In: *Article in International Journal of Prognostics and Health Management* 3 (2017), p. 60. DOI: 10.1016/j.euprot.2015.07.015.

[24] A. K. S. Jardine, D. Lin, and D. Banjevic. "A review on machinery diagnostics and prognostics implementing condition-based maintenance". In: *Mechanical Systems and Signal Processing* 20.7 (2006), pp. 1483–1510. DOI: 10.1016/j.ymssp.2005.09.012.

[25] P. W. Kalgren et al. "Defining PHM, a lexical evolution of maintenance and logistics". In: *AUTOTESTCON (Proceedings)* (2007), pp. 353–358. DOI: 10.1109/AUTEST.2006.283685.

[26] P. Lall, P. Gupta, and A. Angral. "Anomaly detection and classification for PHM of electronics subjected to shock and vibration". In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 2.11 (2012), pp. 1902–1918. DOI: 10.1109/TCPMT.2012.2207460.

[27] X. Li, Q. Ding, and J. Q. Sun. "Remaining useful life estimation in prognostics using deep convolution neural networks". In: *Reliability Engineering and System Safety* 172.December 2017 (2018), pp. 1–11. DOI: 10.1016/j.ress.2017.11.021.

[28] G. Vachtsevanos et al. "Fault Prognostics". In: *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. John Wiley & Sons, 2007. Chap. 6, pp. 280–350.

[29] M. Yasar and T. E. Lovett. "PHM decision support under uncertainty". In: *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM* 2016-Octob (2016), pp. 244–250. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85030257375&partnerID=40&md5=71b32052b9e723d5adcce2aaccc2684b.

[30] J. Liu, W. Wang, and F. Golnaraghi. "A multi-step predictor with a variable input pattern for system state forecasting". In: *Mechanical Systems and Signal Processing* 23.5 (2009), pp. 1586–1599. DOI: 10.1016/j.ymssp.2008.09.006.

[31] M. Daigle. *Model-Based Prognostics*. Tech. rep. 2014. URL: https://www.phmsociety.org/sites/phmsociety.org/files/Daigle-ModelBasedPrognostics-Tutorial-PHM2014_1.pdf (visited on 01/29/2018).

[32] P. Harrington. "Machine learning basics". In: *Machine Learning in Action*. Manning Publications, 2012. Chap. 1, pp. 3–17.

[33] M. Negnevitsky. "Artificial Neural Networks". In: *Artificial Intelligence: A guide to intelligent systems*. Third edit. Pearson, 2011. Chap. 6, pp. 164–218.

[34] J. D. Kelleher, B. M. Nameen, and A. D'Arcy. "Machine Learning for Predictive Data Analytics". In: *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. 2015. Chap. Chapter 1, pp. 1–19.

[35] F. Chollet. "What is Deep Learning?" In: *Deep Learning with Python*. Manning Publications, 2018. Chap. 1, pp. 3–23.

[36] X. Chen and X. Lin. "Big Data Deep Learning". In: *IEEE Access* 2 (2014), pp. 514–525. DOI: 10.1109/ACCESS.2014.2325029.

[37] F. Schroff, D. Kalenichenko, and J. Philbin. "FaceNet: A unified Embedding for Face Recognition and Clustering". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.

[38] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: (Sept. 2014). URL: http://arxiv.org/abs/1409.0473.

[39] D. Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961.

[40] W. Bao, J. Yue, and Y. Rao. "A deep learning framework for financial time series using stacked autoencoders and long- short term memory". In: *International Committee of the Red Cross* (2004), pp. 1–24. DOI: 10.1371/journal.pone.0180944.

[41] W. Lu et al. “A novel feature extraction method using deep neural network for rolling bearing fault diagnosis”. In: *Proceedings of the 2015 27th Chinese Control and Decision Conference, CCDC 2015* (2015), pp. 2427–2431. DOI: 10.1109/CCDC.2015.7162328.

[42] Y. Wu et al. “Remaining useful life estimation of engineered systems using vanilla LSTM neural networks”. In: *Neurocomputing* 275 (2018), pp. 167–179. DOI: 10.1016/j.neucom.2017.05.063.

[43] S. Marsland. “Introduction”. In: *Machine Learning: An Algorithmic Perspective*. Second Edi. CRC Press, 2015. Chap. 1, pp. 1–15.

[44] M. F. A. Hady and F. Schwenker. “Semi-supervised Learning”. In: *Handbook on Neural Information Processing*. Springer Berlin Heidelberg, 2013. Chap. 7, pp. 215–239. ISBN: 978-3-642-36657-4. DOI: 10.1007/978-3-642-36657-4{_}7.

[45] S. Sharma. *Activation Functions: Neural Networks*. 2017. URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (visited on 11/29/2018).

[46] S. Marsland. “The Multi-layer Perceptron”. In: *Machine Learning: An Algorithmic Perspective*. Second Edi. CRC Press, 2015. Chap. 4, pp. 71–111.

[47] L. Bottou. “Large-scale machine learning with stochastic gradient descent”. In: *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers* (2010), pp. 177–186. DOI: 10.1007/978-3-7908-2604-3-16.

[48] J. Duchi, E. Hazan, and Y. Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Jmlr* 12 (2011), pp. 1–40.

[49] T. Tieleman and G. Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31. (Visited on 05/26/2019).

[50] T. Bouda. *Day 69: rmsprop*. 2017. URL: https://medium.com/100-days-of-algorithms/day-69-rmsprop-7a88d475003b (visited on 05/26/2019).

[51] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: (Dec. 2014). URL: http://arxiv.org/abs/1412.6980.

[52] S. Marsland. “Neurons, Neural Networks, and Linear Discriminants”. In: *Machine Learning: An Algorithmic Perspective*. Second edi. CRC Press, 2015. Chap. 3, pp. 39–71.

[53] F. Chollet. “Fundamentals of machine learning”. In: *Deep Learning with Python*. Manning Publications, 2018. Chap. 4, pp. 93–117.

[54] F. Chollet. “Deep learning for text and sequences”. In: *Deep Learning with Python*. Manning Publications, 2018. Chap. 6, pp. 178–233.

[55] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memoory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. URL: http://didawiki.di.unipi.it/lib/exe/fetch.php/magistraleinformatica/aa2/lstm.pdf.

[56] F. A. Gers, J. Schmidhuber, and F. Cummins. “Learning to forget: LSTM”. In: (1999), pp. 1–19. DOI: 10.1162/089976600300015015.

[57] K. Cho et al. “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches”. In: (2014). DOI: 10.3115/v1/W14-4012.

[58] K. Greff et al. “LSTM: A Search Space Odyssey”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (Oct. 2017), pp. 2222–2232. DOI: 10.1109/TNNLS.2016.2582924.

[59] C. Olah. *Understanding LSTM networks*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 01/26/2019).

[60] Y. Lecun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[61] W. Liu et al. “A survey of deep neural network architectures and their applications”. In: *Neurocomputing* 234.October 2016 (2017), pp. 11–26. DOI: 10.1016/j.neucom.2016.12.038.

[62] Standford University: CS231a. *Introduction to Convolutional Neural Networks*. 2018. URL: https://web.stanford.edu/class/cs231a/lectures/intro_cnn.pdf (visited on 11/30/2018).

[63] D. Gilleman. *Convolutional Network (CNN)*. 2018. URL: http://www.deeplearningessentials.science/convolutionalNetwork/ (visited on 11/30/2018).

[64] Standford University: CS231n. *Convolutional Networks*. 2017. URL: http://cs231n.github.io/convolutional-networks/#overview (visited on 11/30/2018).

[65] F. Chollet. “Deep learning for computer vision”. In: *Deep Learning with Python*. Manning Publications, 2018. Chap. 5, pp. 119–177.

[66] F. Jia et al. “Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data”. In: *Mechanical Systems and Signal Processing* 72-73 (2016), pp. 303–315. DOI: 10.1016/j.ymssp.2015.10.025.

[67] X. Lu et al. “Speech enhancement based on deep denoising autoencoder”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* August (2013), pp. 436–440.

[68] J. Li, M.-T. Luong, and D. Jurafsky. “A Hierarchical Neural Autoencoder for Paragraphs and Documents”. In: (2015). DOI: 10.3115/v1/P15-1107.

[69] A. Krizhevsky and G. E. Hinton. “Using Very Deep Autoencoders for Content-Based Image Retrieval”. In: *ESANN 2011, 19th European Symposium on Artificial Neural Networks* (2011), pp. 27–29. DOI: citeulike-article-id:4640046.

[70] Y. Ma et al. “Deep Learning for Fault Diagnosis Based on Multi-sourced Heterogeneous Data”. In: *International Conference on Power System Technology* (2014), pp. 739–745.

[71] J. Jordan. *Introduction to autoencoder*. 2018. URL: https://www.jeremyjordan.me/autoencoders/ (visited on 11/29/2018).

[72] G. E. Hinton. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (July 2006), pp. 504–507. DOI: 10.1126/science.1127647.

[73] X. Qiu et al. “Ensemble deep learning for regression and time series forecasting”. In: *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIEL 2014: 2014 IEEE Symposium on Computational Intelligence in Ensemble Learning, Proceedings* (2014). DOI: 10.1109/CIEL.2014.7015739.

[74] P. Tamilselvan and P. Wang. "Failure diagnosis using deep belief learning based health state classification". In: *Reliability Engineering and System Safety* 115 (2013), pp. 124–135. DOI: 10.1016/j.ress.2013.02.022.

[75] G. E. Hinton, S. Osindero, and Y.-W. Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18.7 (July 2006), pp. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527.

[76] B. Chopard and M. Tomassini. "Particle swarm optimization". In: *Natural Computing Series* (2018), pp. 97–102. DOI: 10.1007/978-3-319-93073-2{_}6.

[77] C. Silva, G. M. Asher, and M. Sumner. "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks Venu". In: 2.1 (2002), pp. 1279–1284.

[78] P. R. Lorenzo et al. "Particle swarm optimization for hyper-parameter selection in deep neural networks". In: *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '17* (2017), pp. 481–488. DOI: 10.1145/3071178.3071208.

[79] X. C. Guo et al. "A novel LS-SVMs hyper-parameter selection based on particle swarm optimization". In: *Neurocomputing* 71.16-18 (2008), pp. 3211–3215. DOI: 10.1016/j.neucom.2008.04.027.

[80] A. P. Engelbrecht. "Particle Swarm Optimisation". In: *Fundamental of Computational Swarm Intelligence*. Wiley, 2005. Chap. 2, pp. 23–67. ISBN: 978-0-470-09191-3.

[81] P. Malhotra et al. "Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder". In: (2016). DOI: 10.1145/1235.

[82] L. Guo et al. "A recurrent neural network based health indicator for remaining useful life prediction of bearings". In: *Neurocomputing* 240 (2017), pp. 98–109. DOI: 10.1016/j.neucom.2017.02.045.

[83] B. Saha et al. "Prognostics Methods for Battery Health Monitoring Using a Bayesian Framework". In: *Instrumentation and Measurement, IEEE Transactions on* 58.2 (2009), pp. 291–296. DOI: 10.1109/TIM.2008.2005965.

[84] Y. G. Li and P. Nilkitsaranont. "Gas turbine performance prognostic for condition-based maintenance". In: *Applied Energy* 86.10 (2009), pp. 2152–2161. DOI: 10.1016/j.apenergy.2009.02.011.

[85] PHM. *IEEE International Conferance on Prognostics and Health Management*. 2018. URL: http://phmconf.org/phmtopics.html (visited on 10/04/2018).

[86] Y. Peng, M. Dong, and M. J. Zuo. "Current status of machine prognostics in condition-based maintenance: A review". In: *International Journal of Advanced Manufacturing Technology* 50.1-4 (2010), pp. 297–313. DOI: 10.1007/s00170-009-2482-0.

[87] Y. Li et al. "Dynamic prognostic prediction of defect propagation on rolling element bearings". In: *Tribology Transactions* 42.2 (1999), pp. 385–392. DOI: 10.1080/10402009908982232.

[88] C. H. Oppenheimer and K. A. Loparo. "Physically based diagnosis and prognosis of cracked rotor shafts". In: July 2002 (2002), pp. 122–132. DOI: 10.1117/12.475502.

[89] C. Byington and P. Stoelting. "A Model-Based Approach to Prognostics and Health Management for Flight Control Actuators"". In: *IEEE Aerospace conference* (2004), pp. 3551–3562. DOI: 10.1109/AERO.2004.1368172.

[90] K. L. Butler. "An Expert System Based Framework for an Incipient Failure Detection and Predictive Maintenance System". In: *Proceeding of the Int Conf on Intelligent Sys Application to Power Sys* (1996), pp. 321–326. DOI: 10.1080/002075400188933.

[91] T. Biagetti and E. Sciubba. "Automatic diagnostics and prognostics of energy conversion processes via knowledge-based systems". In: *Energy* 29.12-15 SPEC. ISS. (2004), pp. 2553–2572. DOI: 10.1016/j.energy.2004.03.031.

[92] S. S. Choi et al. "Development of an On-Line Fuzzy Expert System for Integrated Alarm Processing in Nuclear Power Plants". In: *IEEE Transactions on Nuclear Science* 42.4 (1995), pp. 1406–1418. DOI: 10.1109/23.467727.

[93] C. Frelicot and B. Dubuisson. "An Adaptive Predictive Diagnostic System Based on Fuzzy Pattern Recognition". In: *IFAC Proceedings Volumes* 26.2, Part 5 (1993), pp. 565–568. DOI: https://doi.org/10.1016/S1474-6670(17)48330-3.

[94] A. Ray and S. Tangirala. "Stochastic modeling of fatigue crack dynamics for on-line failure prognostics". In: *IEEE Transactions on Control Systems Technology* 4.4 (1996), pp. 443–451. DOI: 10.1109/87.508893.

[95] J. Yan, M. Koç, and J. Lee. "A prognostic algorithm for machine performance assessment and its application". In: *Production Planning and Control* 15.8 (2004), pp. 796–801. DOI: 10.1080/09537280412331309208.

[96] S. Zhang and R. Genesan. "Multivariable trend analysis using neural networks for intelligent diagnostics of rotating machinery". In: *Trans ASME J Eng Gas Turbine Power* 119 (1997), pp. 378–384.

[97] R. C. M. Yam et al. "Intelligent predictive decision support system for condition-based maintenance". In: *International Journal of Advanced Manufacturing Technology* 17.5 (2001), pp. 383–391. DOI: 10.1007/s001700170173.

[98] C. Byington, M. Watson, and D. Edwards. "Data-driven neural network methodology to remaining life predictions for aircraft actuator components". In: *IEEE Aerospace Conference Proceedings* 6 (2004), pp. 3581–3589. DOI: 10.1109/AERO.2004.1368175.

[99] S. A. Yoon et al. "Semi-supervised Learning with Deep Generative Models for Asset Failure Prediction". In: *CoRR* abs/1709.0 (2017). URL: http://arxiv.org/abs/1709.00845.

[100] S. Khan and T. Yairi. "A review on the application of deep learning in system health management". In: *Mechanical Systems and Signal Processing* 107 (2018), pp. 241–265. DOI: 10.1016/j.ymssp.2017.11.024.

[101] D. Park, Y. Hoshi, and C. C. Kemp. "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder". In: *IEEE Robotics and Automation Letters* 3.3 (July 2019), pp. 1544–1551. DOI: 10.1109/LRA.2018.2801475.

[102] P. Malhotra et al. "Long Short Term Memory Networks for Anomaly Detection in Time Series". In: (2015).

[103] P. Malhotra et al. "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection". In: *CoRR* abs/1607.00148 (2016). arXiv: 1607.00148. URL: http://arxiv.org/abs/1607.00148.

[104] W. Yan and L. Yu. "On Accurate and Reliable Anomaly Detection for Gas Turbine Combustors : A Deep Learning Approach". In: *PHM Conference* (2015), pp. 1–8.

[105] J. An and S. Cho. "Variational Autoencoder based Anomaly Detection using Reconstruction Probability". In: 2 (2015). DOI: 10.1007/BF00758335.

[106] A. L. Ellefsen et al. "An Unsupervised Reconstruction-Based Fault Detection Algorithm for Maritime Components". In: *IEEE Access* 7 (2019), pp. 16101–16109. DOI: 10.1109/ACCESS.2019.2895394.

[107] D. F. Wulsin et al. "Modeling electroencephalography waveforms with semi-supervised deep belief nets: fast classification and anomaly measurement". In: *Journal of Neural Engineering* 8.3 (June 2011), p. 036015. DOI: 10.1088/1741-2560/8/3/036015.

[108] H. Zenati et al. "Efficient GAN-Based Anomaly Detection". In: (Feb. 2018). URL: http://arxiv.org/abs/1802.06222.

[109] S. K. Lim et al. "DOPING: Generative Data Augmentation for Unsupervised Anomaly Detection with GAN". In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, Nov. 2018, pp. 1122–1127. ISBN: 978-1-5386-9159-5. DOI: 10.1109/ICDM.2018.00146.

[110] E. Balouji et al. "A LSTM-based deep learning method with application to voltage dip classification". In: *2018 18th International Conference on Harmonics and Quality of Power (ICHQP)*. IEEE, May 2018, pp. 1–5. ISBN: 978-1-5386-0517-2. DOI: 10.1109/ICHQP.2018.8378893.

[111] D. Xiao et al. "Fault Diagnosis of Asynchronous Motors Based on LSTM Neural Network". In: *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*. IEEE, Oct. 2018, pp. 540–545. ISBN: 978-1-5386-5380-7. DOI: 10.1109/PHM-Chongqing.2018.00098.

[112] J. Liu et al. "Fault Detection for Gas Turbine Hot Components Based on a Convolutional Neural Network". In: *Energies* 11.8 (Aug. 2018), p. 2149. DOI: 10.3390/en11082149.

[113] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu. "Time Series Classification with Multivariate Convolutional Neural Network". In: *IEEE Transactions on Industrial Electronics* (2018), pp. 1–1. DOI: 10.1109/TIE.2018.2864702.

[114] W. Zhao et al. "Fault diagnosis for centrifugal pumps using deep learning and softmax regression". In: *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)* 2016-Septe (2016), pp. 165–169. DOI: 10.1109/WCICA.2016.7578673.

[115] N. K. Verma et al. "Intelligent condition based monitoring of rotating machines using sparse auto-encoders". In: *PHM 2013 - 2013 IEEE International Conference on Prognostics and Health Management, Conference Proceedings* (2013), pp. 1–7. DOI: 10.1109/ICPHM.2013.6621447.

[116] G. Liu, H. Bao, and B. Han. "A Stacked Autoencoder-Based Deep Neural Network for Achieving Gearbox Fault Diagnosis". In: *Mathematical Problems in Engineering* 2018 (July 2018), pp. 1–10. DOI: 10.1155/2018/5105709.

[117] H. Shao et al. "Rolling bearing fault diagnosis using an optimization deep belief network". In: *Measurement Science and Technology* 26.11 (Nov. 2015), p. 115002. DOI: 10.1088/0957-0233/26/11/115002.

[118] M. Ma et al. "Bearing degradation assessment based on weibull distribution and deep belief network". In: *2016 International Symposium on Flexible Automation (ISFA)*. IEEE, Aug. 2016, pp. 382–385. ISBN: 978-1-5090-3467-3. DOI: 10.1109/ISFA.2016.7790193.

[119] V. T. Tran, F. Althobiani, and A. Ball. "An approach to fault diagnosis of reciprocating compressor valves using Teager-Kaiser energy operator and deep belief networks". In: *Expert Systems with Applications* 41.9 (2014), pp. 4113–4122. DOI: 10.1016/j.eswa.2013.12.026.

[120] L. Liao and H. i. Ahn. "Combining deep learning and survival analysis for asset health management". In: *International Journal of Prognostics and Health Management* 7.Special Is (2016). DOI: 10.1109/CBMS.2010.6042612.

[121] Repository NASA Ames Prognostics Data. *NASA Prognostics Center - Publications associated with Datasets*. 2018. URL: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/publications/#turbofan (visited on 04/05/2019).

[122] F. O. Heimes. "Recurrent Neural Networks for Remaining Useful Life Estimation". In: *Prognostics and Health Management, 2008. PHM 2008. International Conference on* (2008), pp. 1–6. DOI: 10.1109/PHM.2008.4711422.

[123] A. L. Ellefsen et al. "Validation of Data-Driven Labeling Approaches Using a Novel Deep Network Structure for Remaining Useful Life Predictions". In: *IEEE Access* (2019), pp. 1–1. DOI: 10.1109/ACCESS.2019.2920297.

[124] M. Yuan, Y. Wu, and L. Lin. "Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network". In: *AUS 2016 - 2016 IEEE/CSAA International Conference on Aircraft Utility Systems* (2016), pp. 135–140. DOI: 10.1109/AUS.2016.7748035.

[125] A. L. Ellefsen et al. "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture". In: *Reliability Engineering & System Safety* 183 (Mar. 2019), pp. 240–251. DOI: 10.1016/j.ress.2018.11.027.

[126] S. Zheng et al. "Long short-term memory network for remaining useful life estimation". In: *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*. 2017, 88–95. ISBN: 9781509003822. DOI: 10.1109/ICPHM.2017.7998311.

[127] A. Z. Hinchi and M. Tkiouat. "Rolling element bearing remaining useful life estimation based on a convolutional long-short-term memory network". In: *Procedia Computer Science* 127 (2018), pp. 123–132. DOI: 10.1016/j.procs.2018.01.106.

[128] Y. Zhang et al. "Long Short-Term Memory Recurrent Neural Network for Remaining Useful Life Prediction of Lithium-Ion Batteries". In: *IEEE Transactions on Vehicular Technology* 67.7 (July 2018), pp. 5695–5705. DOI: 10.1109/TVT.2018.2805189.

[129] A. Zhang et al. "Transfer Learning with Deep Recurrent Neural Networks for Remaining Useful Life Estimation". In: *Applied Sciences* 8.12 (Nov. 2018), p. 2416. DOI: 10.3390/app8122416.

[130] G. Tang et al. "Prediction of bearing performance degradation with bottleneck feature based on LSTM network". In: *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, May 2018, pp. 1–6. ISBN: 978-1-5386-2222-3. DOI: 10.1109/I2MTC.2018.8409564.

[131] J. S. L. Senanayaka, H. V. Khang, and K. G. Robbersmyr. "Autoencoders and Recurrent Neural Networks Based Algorithm for Prognosis of Bearing Life". In: *2018 21st International Conference on Electrical Machines and Systems (ICEMS)*. IEEE, Oct. 2018, pp. 537–542. ISBN: 978-89-86510-20-1. DOI: 10.23919/ICEMS.2018.8549006.

[132] Giduthuri S. B., P. Zhao, and X.-l. Li. "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life". In: (2016), pp. 214–228. DOI: 10.1007/978-3-319-32025-0.

[133] J. Deutsch and D. He. "Using Deep Learning-Based Approach to Predict Remaining Useful Life of Rotating Components". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48.1 (2017), pp. 11–20. DOI: 10.1109/TSMC.2017.2697842.

[134] Z. Tian. "An artificial neural network method for remaining useful life". In: (2012), pp. 227–237. DOI: 10.1007/s10845-009-0356-9.

[135] M. C. Carnera. "Selection of diagnostic techniques and instrumentation in a predictive maintenance program. A case study". In: *Decision Support Systems* 38.4 (2005), pp. 539–555. DOI: 10.1016/j.dss.2003.09.003.

[136] S. Maurya et al. "Fusion of Low-level Features with Stacked Autoencoder for Condition based Monitoring of Machines". In: *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE, June 2018, pp. 1–8. ISBN: 978-1-5386-1165-4. DOI: 10.1109/ICPHM.2018.8448969.

[137] B. Ma, Y. Zhao, and Z. Jiang. "Application of Variational Auto-Encoder in Mechanical Fault Early Warning". In: *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*. IEEE, Oct. 2018, pp. 1263–1268. ISBN: 978-1-5386-5380-7. DOI: 10.1109/PHM-Chongqing.2018.00221.

[138] Sperre Industri AS. *Sperre Compressors*. 2018. URL: http://www.sperre.com/about-sperre (visited on 11/26/2018).

[139] Portland Compressor. *Air Compressor Maintenance*. 2018. URL: https://www.portlandcompressor.com/compressor/maintenance.aspx (visited on 11/26/2018).

[140] Arizona Pneumatic. *Air Compressor Maintenance*. 2018. URL: http://www.arizonapneumatic.com/air-compressor-maintenance.html (visited on 11/26/2018).

[141] A&W Compressor & Mechanical Services. *Preventative Maintenance Compressor*. 2018. URL: http://www.awcompressor.com/preventative-maintenance-compressor.html (visited on 11/26/2018).

[142] Zabatt Power Systems. *Routine Compressor Maintenance*. 2018. URL: https://static1.squarespace.com/static/54c2bf5fe4b0d95d7952701d/t/54fdd3bfe4b03c4958845519/1425920959160/zabatt-compressor-maintenance.pdf (visited on 11/26/2018).

[143] M. Mazanec. *Air Compressor Maintenance*. 2017. URL: https://www.compressorworld.com/blog/air-compressor-maintenance/ (visited on 11/26/2018).

[144] Quincy Compressor. *Common Mistakes in Air Compressor Maintenance*. 2018. URL: https://www.quincycompressor.com/common-mistakes-air-compressor-maintenance/ (visited on 11/26/2018).

[145] IAC. *Compressed Air System Maintenance*. 2018. URL: https://www.iacserv.com/air_compressor_predictive_maintenance.html (visited on 11/26/2018).

[146] Atlas Copco. *Cost Saving Opportunities: Maintenance*. 2018. URL: https://www.atlascopco.com/en-us/compressors/wiki/compressed-air-articles/compressor-maintenance (visited on 11/26/2018).

[147] Industrial Compressor Solutions. *Preventative vs. Predictive Maintenance*. 2018. URL: https://www.industrialcompressorsolutions.com/articles/preventative-vs.-predictive-maintenance (visited on 11/26/2018).

[148] M. Bacidore. *Pdm applications to improve compressed air efficiency*. 2014. URL: https://www.plantservices.com/blogs/plant-ambassador/pdm-applications-to-improve-compressed-air-efficiency/ (visited on 11/26/2018).

[149] A. Saxena et al. "Damage propagation modeling for aircraft engine run-to-failure simulation". In: *2008 International Conference on Prognostics and Health Management*. IEEE, Oct. 2008, pp. 1–9. ISBN: 978-1-4244-1935-7. DOI: 10.1109/PHM.2008.4711414.

[150] C. Voskoglou. *What is the best programming language for Machine Learning?* 2017. URL: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7 (visited on 11/28/2018).

[151] Pandas. *Pandas: Python Data Analysis Library*. 2018. URL: https://pandas.pydata.org/ (visited on 03/16/2019).

[152] Scikit-learn. *Scikit-learn: Machine Learning in Python*. 2018. URL: https://scikit-learn.org/stable/ (visited on 03/16/2019).

[153] TensorFlow. *TensorFlow*. 2018. URL: https://www.tensorflow.org/ (visited on 03/16/2019).

[154] Pytorch. *Pytorch*. 2018. URL: https://pytorch.org/ (visited on 03/16/2019).

[155] Pywarms. *Pyswarms*. 2017. URL: https://pyswarms.readthedocs.io/en/latest/ (visited on 03/16/2019).

[156] J. H. Metzen. *Variational Autoencoder in TensorFlow*. 2015. URL: https://jmetzen.github.io/2015-11-27/vae.html (visited on 03/16/2019).

[157] M. Rastogi. *Deep-Belief-Network-Pytorch*. 2018. URL: https://github.com/mehulrastogi/Deep-Belief-Network-pytorch (visited on 03/16/2019).

[158] A. Elsheikh, S. Yacout, and M.-S. Ouali. "Bidirectional handshaking LSTM for remaining useful life prediction". In: *Neurocomputing* 323 (Jan. 2019), pp. 148–156. DOI: 10.1016/j.neucom.2018.09.076.

# Appendix

- Appendix A - Project proposal
- Appendix B - Demo PHM solution
- Appendix C - Research paper abstracts

# Appendix A: Project proposal

NTNU – Ålesund
Norwegian University of
Science and Technology

**03.12.2018**
**MSc in Simulation and Visualization**

# Prognostics and Health Management for Air Compressors based on Deep Learning techniques

## Introduction

Prognostics and Health Management (PHM) solutions are becoming increasingly popular as a part of predictive maintenance strategies. Such solutions can contribute to optimized maintenance management, thus reduce both costs and unexpected standstills. Maintenance is often a large portion of a company's expenses. Therefore, it is a topic relevant for improvements. While traditional maintenance strategies aim to repair something after it fails or do maintenance on fixed intervals, PHM solutions try to capture the condition and do maintenance when it is necessary and most convenient. The ultimate goal of PHM is zero-downtime performance. In recent years Deep Learning (DL) has proved its use in many industries and been able to solve complex problems, also within the field of PHM.

The thesis will explore how a PHM solution based on DL techniques for air compressors can be developed, and the pros and cons of such a system. Sperre Industri AS will be a partner through the thesis, by providing both domain knowledge and access to experimental data from their compressors.


Figure 1: A compressor from Sperre Industri AS

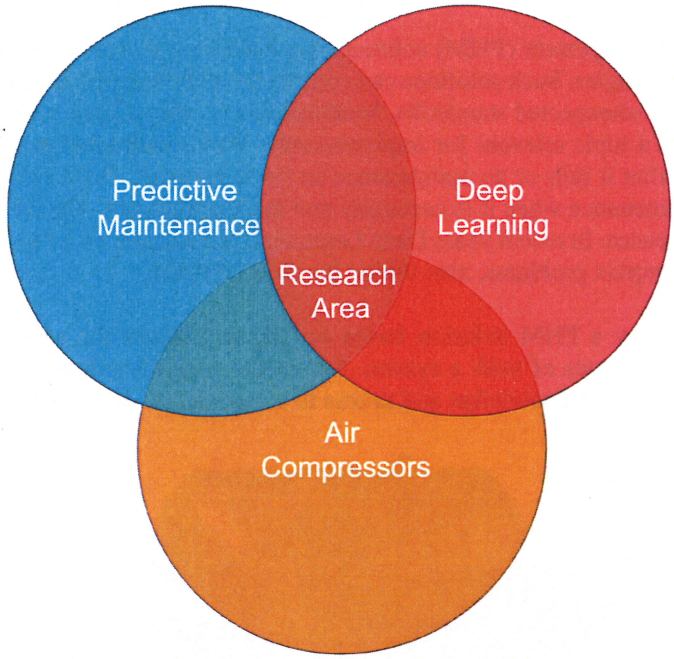
## Motivation

Traditional maintenance strategies such as corrective and preventive maintenance are still used in most industries. Predictive maintenance is becoming a popular supplement or replacement to the traditional strategies. In addition to increasing reliability of a system, this can reduce the environmental footprint, which is in focus globally.

Automating systems, especially towards autonomous vehicles has received a lot of attention in recent years. In order to achieve fully automated systems, PHM solutions are important. This is because predicting the remaining useful life (RUL) makes it possible to schedule maintenance procedures to the most convenient time. DL is considered a promising research area for PHM solutions. The motivation for defining exploration around DL techniques is that systems based on traditional methods are highly application dependent. DL techniques can on the other hand result in a more flexible solution that easily can be fitted to new systems. This can hopefully contribute to improved maintenance for air compressors.

## Scope

The scope of the project is to explore DL techniques for PHM on air compressor systems. The techniques are used to recognize patterns in data and contribute to determining when to do maintenance. Experiments are conducted on an air compressor from Sperre Industri AS, but the thesis aims to provide suggestions on flexible methods to use in a PHM concept. The scope lay within the boundaries of predictive maintenance, deep learning and air compressors.




## Objectives

The research goal is as follows:

- **Goal**: *Explore approaches and show proof of concept of how DL can be used in a PHM solution to detect anomalous behaviour, identify faults and predict future failures on air compressors.*

The stated research goal includes exploring if a PHM solution based on DL techniques can improve the current maintenance strategy on air compressors. The main part revolves around exploring how DL can be used for anomaly detection, diagnostics and prognostics. In this context, prognostics is the prediction of RUL, diagnostics is identifying faults, while anomaly detection is to detect if the system deviated from normal operating condition. The techniques should be implemented as proof of concept to show that this is feasible on one of Sperre's compressors. The thesis will try to answer the below research questions (RQs):

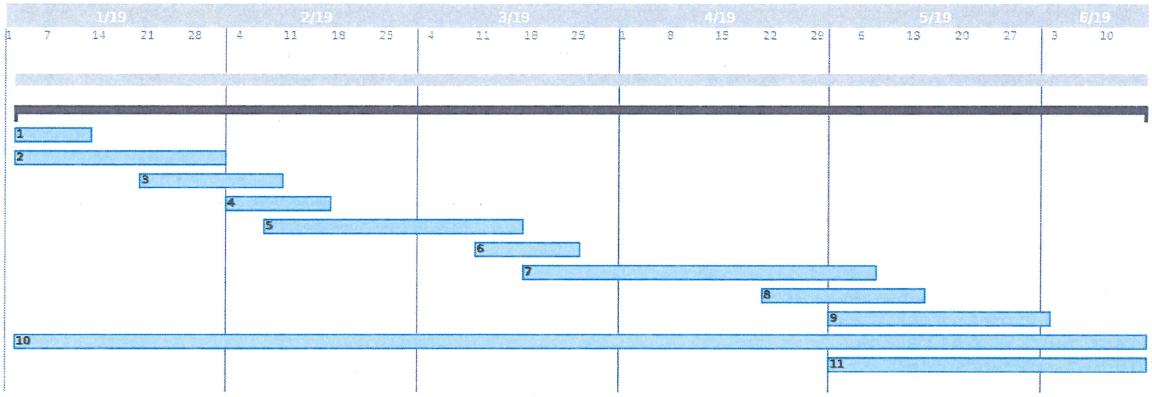
- **RQ1**: How can DL be used to detect abnormal behavior in air compressor systems?
- **RQ2**: How can DL be used to identify faults in air compressor systems?
- **RQ3**: How can DL be used to predict the remaining time until failure, and how to emphasize the typical problem of few run-to-failure examples?
- **RQ4**: What are the advantages and disadvantages of using DL in a PHM system for air compressors, and how does it improve the current maintenance strategy on air compressors?

**Milestones**:
The milestones, tasks and schedule are preliminary and will be adjusted in the future.

Tasks:

1. Identification and review of research questions, scope and objectives.
2. Literature review on PHM solution based on DL and predictive maintenance in general.
3. Decide on which DL techniques to test for Diagnostics and Prognostics
4. Define experiment case and collect relevant data from laboratory
5. Run experiments on Anomaly Detection
6. Run experiments on Diagnostics
7. Run experiments on Prognostics
8. Analysis of results
9. Analysis and discussion of a PHM solution based on the proposed techniques
10. Writing report
11. Evaluation

**Schedule**




The scope may prove to be different than initially anticipated. It is subject to approval from the supervisor, topics may be added or deleted from the list above or reduce in extent.

Magnus Gribbestad
Magnus Gribbestad
Student – Simulation and Visualization
PHONE (+47) 41569516
E-mail: magnus@gribbestad.no

Ibrahim A. Hameed
Main supervisor – NTNU

André L. Ellefsen
Supervisor – NTNU

Vladimir Krivopolianskii
Supervisor - Sperre Industri AS

# Appendix B: Demo of PHM solution

In this thesis, anomaly detection, diagnostics and prognostics on air compressors has been explored. Each of these serves the purpose of protecting the system and making it more reliable. In this appendix, a demo of how these features can be put together to a PHM system for air compressors is presented. A simple web-based demo is developed using Python and a framework for building web applications called *dash*. Since the thesis is in collaboration with Sperre Industri AS, they will be used as an example. The demo will hopefully inspire and show the usefulness of the cases studied in this thesis.

Sperre has their equipment both on land and on ships sailing around the world. As mentioned earlier, their current maintenance strategy means they need to send replacement parts anywhere in the world, within 48 hours. In their case, it could be useful to have a map that gives an overview of all of their products around the world and their current state. The map can, for instance, show the state with a sphere colored by either the anomaly score or the RUL. Hovering over a sphere can give extra information such as the name of the ship, RUL, anomaly score, which parts are most likely to fail and so on. Figure 1 shows the map which is implemented in the demo, where four example vessels are added.

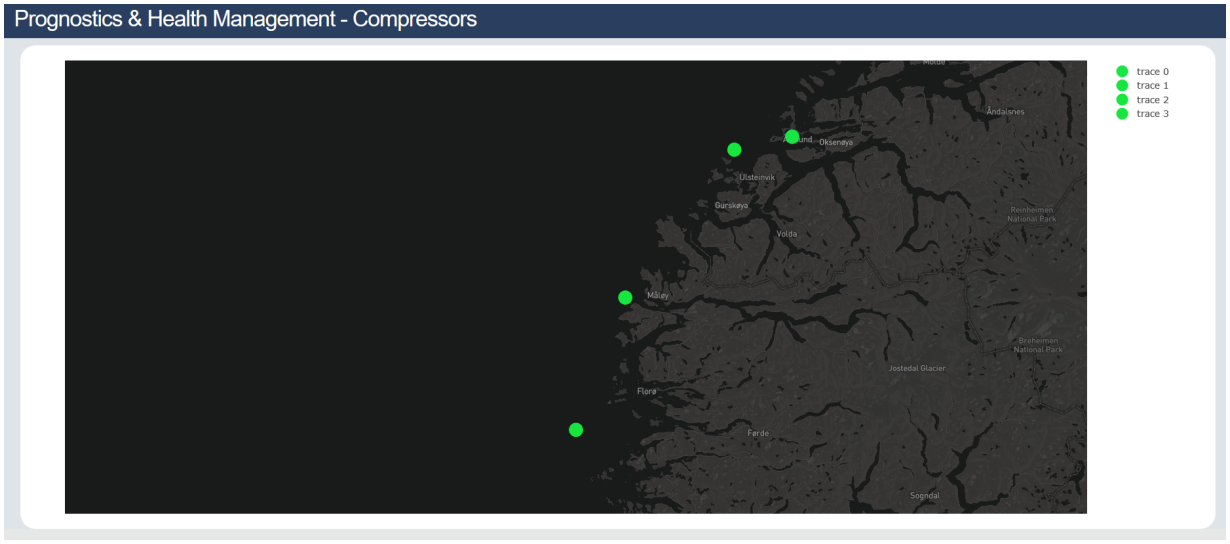



Figure 1: Map in demo PHM-application

Since they have many products, a map is not sufficient. A complete system could, for instance, only display ships where the current status is concerning and where maintenance soon is necessary. This

would filter out healthy products and give a clearer overview of future maintenance. In addition, an alarm list can be included which orders products after the need for maintenance. The list should include essential information such as potential faults, location, and potential shortcuts for maintenance orders and decision support.

If the overview information is not sufficient for making decisions, the product can be accessed by clicking in either the list or the map. This will take the user into another part of the web site, which shows more detailed information about the health of the air compressors. The proposed overview page is shown in figure 2. It is divided into three parts, one for each case in this thesis.

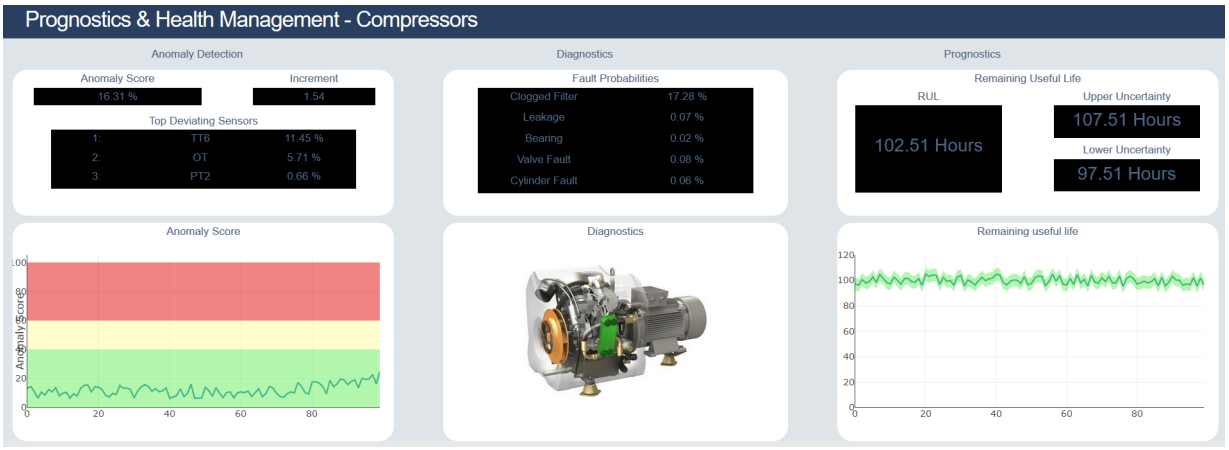



Figure 2: Health overview in demo PHM-application

The first and leftmost section of the web site includes information gained from anomaly detection on the air compressor. Figure 3 shows how this will look when an air compressor is starting to deviate from normal condition. The lower part of the section shows a plot of the anomaly score of the system and how it has evolved. It is divided into normal, warning, and danger zone as proposed earlier in section 5. In addition, the upper section shows the anomaly score in numbers, information about the potential rate of change, and which sensors are contributing to the system deviating from normal condition.

The middle section gives insight into the health of the air compressor based on diagnostics experiments. Figure 4 shows a suggestion of how this section can look. The upper section can list potential faults in an order based on severity. For this demo, random faults are added in the list. When the probability or severity of a fault increases over a certain threshold, the PHM-system can give an alarm. The lower part can show a 3D-model of the product and mark the related parts in color to indicate where the fault is. For this demo, a random image of a compressor from Sperre is used, and a random part is colored. It is included to show the principle, but for a real application, the corresponding compressor model should be used, and the correct parts should be highlighted.

Finally, the rightmost part of the application shows information related to prognostics. Figure 5 shows how the demo proposes to include this information. The upper part can show the value of the RUL prediction, with some corresponding upper and lower uncertainty, for instance, the 10% quantiles. In addition, the lower section includes a graph showing the RUL prediction over time with


Figure 3: Anomaly detection in demo PHM-application

the corresponding uncertainty bounds.

This appendix aimed to give an example of how the individual cases from this thesis could be included in a PHM system for air compressors. It is not a working version, and it only displays values obtained from the individual cases on one of the available sequences. The information showed in the figures is, therefore, realistic data, but not evaluated live. This is done for simplicity. When deploying such a system, it needs to be decided if the analysis is to happen on-site or on one general service center. In order to obtain results from a DL model, they need to be deployed. This can be done by using web servers and accessing them through API's.

A complete PHM system for air compressor should include more functionality and information. It is out of the scope of this thesis, but will be mentioned briefly. A PHM system could have the option to inspect trends of historical data individually. One of the most important features that should be included in such a system is decision support. It can propose which maintenance action should be taken, and if the system is very reliable, it could even automatically order and take maintenance decisions.

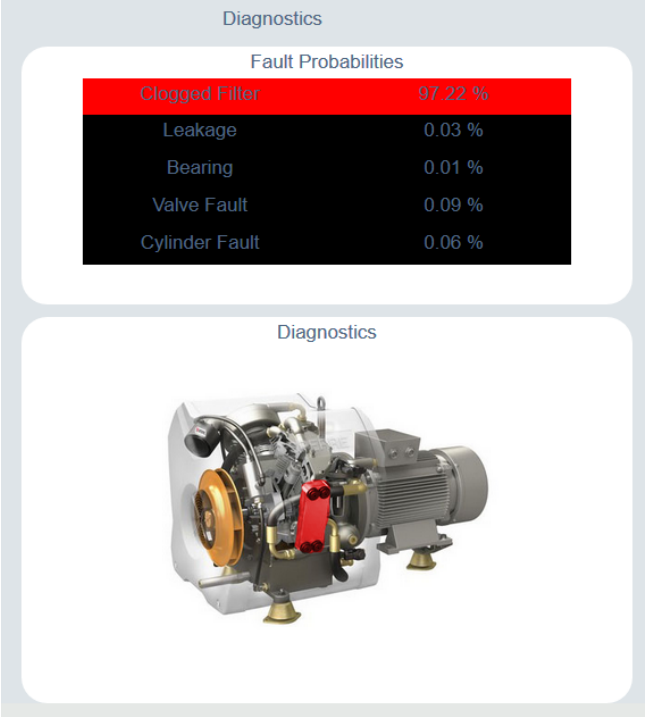



Figure 4: Diagnostics in demo PHM-application

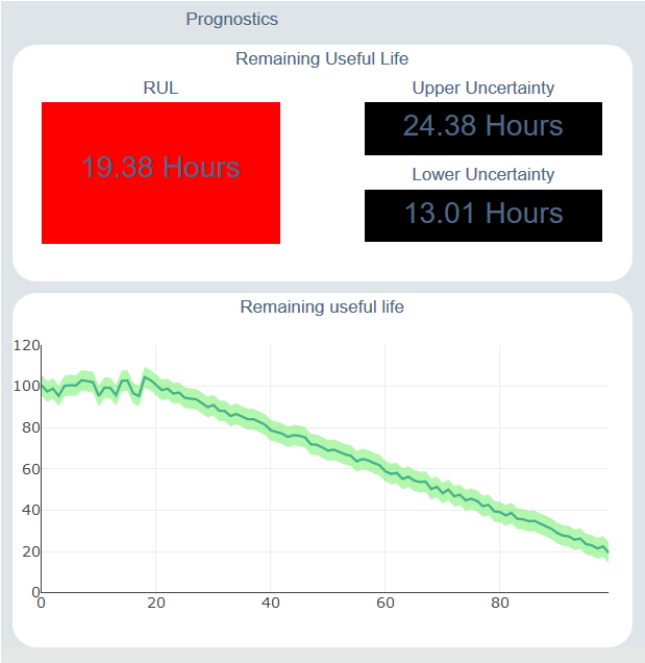



Figure 5: Prognostics in demo PHM-application

# Appendix C: Research paper abstracts

This appendix contains the abstract of three proposed research papers based on this thesis. It is planned to continue the work with these papers.

- C1: Anomaly detection
- C2: Transfer learning for prognostics
- C3: Uncertainty in remaining useful life predictions

**C1 - Anomaly detection**

# Reconstruction-based anomaly detection with increased transparency for air compressors using deep learning

Magnus Gribbestad
Department of ICT and Natural Sciences
Norwegian University of Science and Technology
Aalesund, Norway
Email: magnus@gribbestad.no

Anomaly detection is an important topic that has been much researched. A typical problem in industry is having little labelled data and few run-to-failure examples, which makes it challenging to develop prognostics and health management systems (PHM) with accurate features for fault identification and failure prediction. Certain machine learning (ML) approaches to anomaly detection has the strength that they only require normal data, which reduces the need for historical data with fault labels. In this work, several reconstruction-based deep learning (DL) approaches are explored towards detecting anomalies in air compressors. Anomalies in such systems are not point-anomalies, but instead an increasing deviation from normal condition as the system degrades. This work proposes a method to give a descriptive range on the deviation based on the reconstruction-based techniques. Most anomaly detection approaches are considered black box models, only indicating an anomaly or not. This study proposes a method for increasing transparency of reconstruction-based anomaly detection to indicate which parts of a system contribute to the deviation from expected behaviour. The results show that the proposed methods can detect abnormal behaviour in air compressors accurately, and indicate why it deviates. It was able to detect faults without having historical examples of them. The proposed method for anomaly detection with increasing transparency can be a useful feature to include in a PHM system since it can detect deviations and the reason for them.

**C2 - Transfer learning for prognostics**

# Transfer learning in prognostics: Improving remaining useful life predictions for air compressors

Magnus Gribbestad
Department of ICT and Natural Sciences
Norwegian University of Science and Technology
Aalesund, Norway
Email: magnus@gribbestad.no

Predicting remaining useful life (RUL) is a desired feature in prognostics and health management (PHM) systems since it can contribute to increasing the reliability and reduce the number of unexpected failures. A challenge with such a feature is that it requires many run-to-failure examples, which typically are hard to obtain. Transfer learning has been used for several areas of application to emphasize the problem of few training instances. The principle of transfer learning is to reuse parts of a machine learning (ML) model that has been trained for a related problem, to improve the performance in another. It has been successfully applied to many applications related to image classification, but it has received little attention towards prognostics. This study focuses on transfer learning to improve RUL predictions on air compressors with a limited number of run-to-failure examples. First, a vanilla long short-term memory model without peephole connections was built and trained on the popular C-MAPSS dataset. Next, several architectures based on transferred layers from that model were explored. The results indicate that using new layers in combination with both untrainable and trainable transferred layers can improve the RUL predictions and increase generalization. This has the potential to make prognostics in PHM more accessible for companies.

**C3 - Uncertainty in remaining useful life predictions**

# A data-driven approach for uncertainty in remaining useful life predictions

Magnus Gribbestad
Department of ICT and Natural Sciences
Norwegian University of Science and Technology
Aalesund, Norway
Email: magnus@gribbestad.no

A successful prognostics and health management (PHM) system should be able to predict the remaining useful life (RUL) of a system. Typically, such predictions are represented by a single value indicating the remaining time until system failure. A problem with single-valued predictions is that they give an impression of certainty. Presenting a prediction of 51 remaining hours until failure to service personnel will let them believe that the prediction is accurate. Often the predictions for such cases are much less confident than they express. Therefore, uncertainty boundaries is a feature in demand, which can give more realistic predictions. This study proposes a data-driven method for obtaining uncertainty related to the predictions based on distributions of errors in previous predictions. Two different methods for presenting the uncertainty bounds have been used. The results indicate that this method can show more realistic predictions that can contribute to more qualified maintenance decisions. One of the main benefits of the proposed method is that it is purely data-driven, which similarly to deep learning reduces manual labor.