Machine Learning for Centrifugal Pump Condition Monitoring Using Data from Variable Frequency Drive

Topias Turunen

School of Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 18.11.2022

Supervisor

Asst. Prof. Raine Viitala

Advisors

M.Sc. Jesse Miettinen

M.Sc. Timo Päivinen

Aalto University
School of Engineering

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Topias Turunen

Title Machine Learning for Centrifugal Pump Condition Monitoring Using Data from Variable Frequency Drive

Degree programme Master´s Programme in Mechanical Engineering

Major Mechanical Engineering **Code of major** ENG25

Supervisor Asst. Prof. Raine Viitala

Advisors M.Sc. Jesse Miettinen, M.Sc. Timo Päivinen

Date 18.11.2022 **Number of pages** 62 **Language** English

Abstract

Centrifugal pump maintenance that is not based on its true condition causes unnecessary waste of time and resources. Resources are wasted on premature maintenance operations and replacing still functional parts. Condition-based maintenance strategies, like predictive maintenance, could make maintenance more efficient as the right maintenance procedures can be done exactly when needed. With effective condition monitoring, the condition of the pump is known, its remaining operation time can be predicted, and the possible faults can be diagnosed. However, these strategies are not utilised because of their high upfront cost.

This thesis introduces a novel machine learning application for centrifugal pump condition monitoring and a fault diagnosis based on raw data from a variable frequency drive. A variable frequency drive as a source of data gives a possibility for sensorless condition monitoring. Furthermore, fault diagnosis can be automated by using deep learning methods.

A convolutional neural network type of deep learning model was trained to diagnose the pump conditions. For this thesis, two centrifugal pump fault conditions were studied: cavitation and abrasive metal-metal contact. In order to train the deep learning model, series of experiments were conducted to form a dataset. The dataset was comprised of variable frequency drive raw torque data under the chosen pump conditions.

The trained model achieved a weighted accuracy of 78.4%. The trained model's performance on abrasive metal-metal contact was excellent, whereas differentiating cavitation from normal samples needs improvement. The results demonstrate the possibility of diagnosing multiple pump failures from the variable frequency drive data.

Overall, the results of the thesis are promising. With improvement, this deep learning model could be implemented into variable frequency drive software or as an embedded device. This kind of condition monitoring system has the possibility to make condition-based maintenance more approachable and affordable.

Keywords condition monitoring, fault diagnosis, centrifugal pump, variable frequency drive, cavitation, shaft deflection, deep learning, convolutional neural network

Aalto University
School of Engineering

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Topias Turunen

Työn nimi Koneoppiminen keskipakopumpun kunnonvalvontaan käyttäen taajuusmuuttajadataa

Koulutusohjelma Konetekniikan maisteriohjelma

Pääaine Konetekniikka **Pääaineen koodi** ENG25

Työn valvoja Asst. Prof. Raine Viitala

Työn ohjaajat DI Jesse Miettinen, DI Timo Päivinen

Päivämäärä 18.11.2022 **Sivumäärä** 62 **Kieli** Englanti

Tiivistelmä

Keskipakopumpun Kunnossapito, joka ei perustu keskipakopumpun todelliseen kuntoon, aiheuttaa tarpeetonta ajan ja resurssien haaskausta. Resursseja tuhlataan ennenaikaisiin huoltotoimenpiteisiin ja ehjien komponenttien vaihtoon. Kuntoon perustuvilla kunnossapito menetelmillä, kuten ennustuvalla kunnossapidolla, on mahdollisuus tehdä kunnossapidosta tehokkaampaa. Kunnonvalvonnalla huoltotoimenpiteet voidaan suorittaa oikea aikaisesti. Tehokkaalla pumpun kunnonvalvonnalla pumpun todellinen tila tiedetään, sen jäljellä olevaa toiminta ikää voidaan arvioida ja sen vikoja pystytään määrittämään.

Tässä diplomityössä esitellään taajuusmuuttajan raakadataa hyödyntävä koneoppimissovellus keskipakopumpun kunnonvalvontaan ja vianmääritykseen. Taajuusmuuttaja tietolähteenä antaa edellytykset anturittomalle kunnonvalvonalle. Syväoppimisen avulla vianmääritys voidaan automatisoida.

Konvoluutioneuroverkkoon perustuva syvväoppimismalli koulutettiin määrittämään keskipakopumpun vikatiloja. Tässä työssä tarkasteltiin kahta keskipakopumpun vikatilaa: kavitaatio ja hankaava metalli-metalli kontakti. Neuroverkon koulutusta varten suoritettiin sarja mittauksia muodostamaan tietojoukko. Tietojoukko sisälsi näytteitä taajuusmuuttajan vääntömomenttidatasta valituista pumpun vikatiloissa.

Lopputuloksena saadun mallin painotettu tarkkuus oli 78.4%. Koulutettu malli luokitteli hankaavan metalli-metallikontaktin erinomaisesti, mutta kavitaation erottaminen normaalitilasta vaatii parannusta. Tulokset osoittivat, että on mahdollista määrittää useita vikatiloja taajuusmuuttajan datasta.

Kaiken kaikkiaan, tämän työn tulokset ovat lupaavia. Pidemmälle kehitettynä tämä syväoppimismalli voidaan ottaa käyttöön sisällyttämällä se taajuusmuutajan ohjelmistoon tai erilliseen sulautettuun laitteeseen. Koulutettuun malliin perustuvalla kunnonvalvonta metodilla kuntoon perustuvasta kunnossapidosta saadaan lähestyttävämpää ja edullisempaa.

Avainsanat kunnonvalvonta, vianmääritys, keskipakopumppu, taajuusmuuttaja, kavitaatio, akselin vääntymä, syväoppiminen, konvoluutio neuroverkko

Preface

This thesis was made at Aalto University in co-operation with Sulzer Pumps Finland Oy. From Sulzer, I would like to thank Timo Päivinen, Kalle Tiitinen, Anssi Lehtonen, and everyone from the research center. From Aalto University, I would like to thank my supervisor Raine Viitala and advisor Jesse Miettinen for their guidance. I am grateful for the possibility to work with you all.

I want to thank my parents, sisters, and friends. A special mention goes to Kaisa for your love and support. I would not have made it this far without you.

Otaniemi, 31.10.2022

Topias A. Turunen

Contents

Symbols and abbreviations

Symbols

a_n	n:th layer activation value
b	Bias
b_{2tot}	Impeller outlet width [m]
d_2	Impeller diameter [m]
e	Euler´s number [aprox. 2.71828]
E	Young´s modulus [N/m^2]
F	Load force [N]
f	Power supply frequency [Hz]
F_R	Radial force [N]
g	Gravitational acceleration [Nm/s^2]
$g()$	Activation function
H	Pump head [m]
I	Second momentum of the area [m^4]
k_R	Radial thrust factor
L	Loss function
l	Shaft overhang [m]
m_t	Momentum of the learning rate
n	Number of poles of in the electric motor stator
N	Rotational speed [rpm]
$n_{RMS,N}$	Root mean square of motor rotational speed [rpm]
P	Power [W]
p_{amb}	Upstream total pressure [Pa]
p_v	Vapor pressure [Pa]
Q	Flowrate [l/s]
S	Specific gravity
$T_{RMS,N}$	Root mean square of motor torque [Nm]
v	motor speed [rpm]
v_d	Shaft deflection [m]
v_t	Scaling factor
w	Trainable weight
W	Weight matrix
x	Input
X	Input matrix
y	Label
Y	Output matrix
z	Weighted sum
z_D	Difference between NPSH datum plane reference plane [m]
β	recall is deemed more important than precision
β_1	Momentum factor
β_2	Scaling factor factor
ϵ	Small value to avoid divisions with zero
η	Efficiency
η_{lr}	Learning rate

$\hat{y}$	Output label
$\hat{m}_t$	Bias correction for momentum
$\hat{v}_t$	Bias correction for scaling factor
ρ	Fluid density [kg/m^3]
θ	Trainable parameter

Abbreviations

AC	Alternating current
AdaGrad	Adaptive gradient
Adam	Adaptive moment estimation
AI	Artificial Intelligence
ANN	Artificial neural network
CNN	Convolutional neural network
DC	Direct current
DL	Deep learning
DNN	Deep neural network
ELU	Exponential linear unit
ICA	Independent component analysis
LDA	Latent dirichlet allocation
LReLu	Leaky rectified linear unit
ML	Machine learning
MLP	Multilayer perceptron
MSE	Mean squared error
NN	Neural network
NPSH	Net positive suction head
PCA	Principal component analysis
ReLu	Rectified linear unit
RMS	Root mean square
RMSprop	Root mean squared propagation
RNN	Recursive neural network
RUL	Remaining useful life
SELU	Scaled exponential linear uni
SGD	Stochastic gradient descent
SVM	Support vector machine
VFD	Variable frequency drive

1 Introduction

1.1 Background

In autumn 2022, Europe is in the middle of an energy crisis due to its reliance on Russian natural gas. Electricity shortages and power cuts during the upcoming winter are a real possibility. In Finland, Olkiluoto 3 power plant was meant to start production and alleviate the energy situation. When operational, Olkiluoto 3 would be the largest power plant in Nordic countries and would account for 10% of Finland's energy consumption during the coldest winter days [1]. However, the opening of the plant has been delayed to an undefined date due to damage detected in the feed water pumps [2]. With successful maintenance, the fault could have been detected earlier, the power plant could be operational as scheduled, and human suffering could be reduced.

As the previous example demonstrated, pumps play an important role in many processes and their malfunction can have grave consequences. Manufacturers have an incentive to maintain their pumps in working condition, as pump breaking down can cause crucial processes to shut down. To keep pumps operational, different maintenance strategies have been developed.

The two common strategies are preventive and predictive maintenance, preventive being the more common approach. Preventive maintenance is a strategy where pump components are maintained and repaired ahead of time before they have time to degrade [3]. Maintenance procedures are scheduled based on approximations of the pump component service periods. Predictive maintenance is a technique designed to predict the remaining service time based on the measured condition of the machine. Ideally, this way the pump is maintained exactly when needed and procedures are scheduled so that they cause as few unnecessary expenses as possible. The main elements of predictive maintenance are data collection and preprocessing, condition monitoring, fault diagnosing, time to failure prediction, maintenance scheduling and resource optimization [3].

A major aspect of predictive maintenance is condition monitoring. In condition monitoring, condition parameters are measured and monitored to detect anomalies. Usual condition parameters for pumps are vibration, temperature, pressure, and power [4]. Changes in the condition parameter are indicative of a developing fault. Progression of change in the parameter is used to approximate the need for maintenance. The parameter monitoring can be done either with hand measurements or with permanently installed sensors.

Once an anomaly is detected, the fault needs to be diagnosed. Pumps are intricate machines and therefore have multiple fault states. These include mechanical faults such as axial misalignment, bearing component wear and metal contact. The fault can also be in flow such as cavitation, air bubbles, and turbulence. Some of these faults are easier to diagnose than others and take varying amounts of time to develop. Diagnostics is still usually done by experts by manually analysing the measurement data. Machine learning (ML) is a possible way to automate the diagnostic process.

ML is a form of artificial intelligence. ML models can extract patterns from raw

data [5]. These models must be first trained to recognize patterns with known data after which they can be used to make predictions based on new data. ML is not a recent invention, the first experiments with artificial neural networks (ANN) were conducted in the 1950s [5]. However, with the quite recent increase in computational power and the amount of data available thanks to the industrial internet of things there are more opportunities to take advantage of ML. Today ML methods are used in varying fields such as finance, computer vision, and advertising [5]. ML also offers useful applications for machine condition monitoring.

1.2 Research Problem

Even with maintenance techniques in use pumps still break down unexpectedly and from unknown causes. Both preventive and predictive maintenance have their weak points. Sometimes the methods used are unable to detect the fault. Also, without continuous monitoring, faults can develop so fast that there is no opportunity to detect them.

Preventive maintenance causes maintenance to be often performed prematurely [4]. Because the maintenance is not based on the true condition of the pump, healthy components are maintained and replaced. Unnecessary maintenance wastes time and resources [6]. It is approximated that as much as a third of all maintenance costs are wasted as a result of unnecessary maintenance [3].

Predictive maintenance techniques could improve the efficiency of pump maintenance. However, the reason why predictive maintenance is not widely adopted is the cost [3]. Installing numerous sensors, cabling, and computing devices to monitor the pump condition requires a substantial upfront cost. Additionally, training personnel to diagnose faults costs time and resources. Therefore, predictive maintenance faces the challenge of balancing cost with returns. However, the possible benefits of adopting predictive maintenance are substantial. Widely adapted predictive maintenance has the potential to make pumps more reliable, affordable and better for the environment.

ML methods have been proven to be effective for automating fault diagnosis process [7, 8, 9]. However, to implement ML methods large quantities of quality data are required. The availability of data itself is usually not a problem but the amount of correctly classified data is. The dataset also needs to be representative of the task it is used to learn. Representation of the data is important for minimizing the possibility that the model makes decisions based on irrelevant factors. With a bad dataset, the trained ML model will not generalize well to the real application. No algorithm can produce good results with a flawed dataset. This is usually referred "garbage in garbage out" [10].

1.3 Aim of the Research

The purpose of this thesis is to determine the accuracy of the best-performing neural network model for centrifugal pump condition diagnosis using data from VFD. The model will be trained using a dataset consisting of VFD raw torque data. The

dataset will be collected from a centrifugal pump under different operating conditions. The conditions include normal, cavitation, and abrasive metal-metal contact. The resulting model classifies data samples into these three classes.

The upfront cost of predictive maintenance can be mitigated by using a variable frequency drive (VFD) as the source of condition monitoring data. VFDs are widely used to control centrifugal pump rotational speed as they lower the pump's energy consumption significantly [11]. Because VFD is used to control the pump, data can be measured from it without sensors [12, 13]. Furthermore, the regularity of their use makes them a free source of data.

ML is a viable method for automating the diagnostic process. By automation, less personnel is needed for the analysis, analysis can be performed more frequently, and the faulty process can correct itself [7]. An ML model can be trained to diagnose pump faults based on raw data measured from VFD. The resulting ML model can be embedded on an on-board device, enabling automated real-time health monitoring.

1.4 Scope of the Research

Pumps have multiple possible fault states that could be studied. However, to keep the thesis scope manageable the study must focus only on a couple of pump faults. Furthermore, because of the machine learning method's tendency to require a large amount of data additional fault states increase needed measurements considerably. Consequently, this thesis focuses on only two fault states: cavitation and abrasive metal-metal contact. These pump faults were chosen as they are relatively easy to produce for measurements and traditional condition monitoring methods might not detect them. This is partly due to the momentary nature of these faults that are not always detected in scheduled measurements.

This study focuses on radial centrifugal pumps. Other types of pumps are excluded from the scope. Measurements are taken only from the VFD. Other sources of data for condition monitoring, such as vibration, temperature, and pressure sensors, are not covered in this work.

1.5 Methods

The thesis is comprised of two parts: a literature review and an experimental part. The aim of the literature review is to examine the state of the art and give context and direction for the experiment. In the experimental part data needed for the machine learning model is measured, and an ML model is trained and evaluated.

The literature review is done by using iterative process writing. Process writing starts with researching the subject and creating a preliminary structure for the study [14]. The structure will evolve iteratively as the knowledge of the subject deepens. The table of contents will guide the focus of the research and help to keep the study coherent. When researching the subject some topics might be added, removed, or modified depending on if they are deemed important and within the scope. The structure will always be kept up to date. This also means that the writing does not necessarily follow the same order as the table of content [14].

The experimental part of the study consists of gathering data from the pumps, building the machine learning model and evaluating the results. Data is measured from the VFD of the centrifugal pump in a test loop. Measurements will be taken in different conditions to diversify the data. To achieve this, different factors and components in the process are changed or modified. These factors include the pump, electric motor, variable frequency drive, pumping speed, coupler, and impeller. An ANN is created and trained using the collected data. Results from the model are evaluated using various methods. Used methods, test results and discussion of the experiment will be documented.

2 Literature Review

2.1 Variable Speed Driven Centrifugal Pump

In order to implement condition monitoring methods for a centrifugal pump, the operation of a variable speed driven pumping system must be understood. A typical variable driven centrifugal pumping system consists of a VFD, an induction motor, and the centrifugal pump that is installed into a process. The centrifugal pump is connected to the electric motor which speed is controlled with VFD. This kind of simple pumping system is presented in Figure 1. This chapter introduces these main parts of the pumping system.

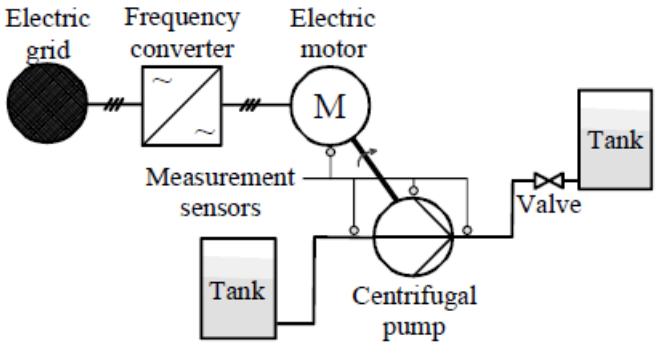

Figure 1: Variable speed driven pumping system [12]

2.1.1 Centrifugal Pump

Pumps are mechanical devices used to increase the pressure of a fluid. Usually, pumps are used to either move liquid in a pipeline from a lower reservoir to a higher reservoir or from lower pressure to higher pressure. Centrifugal pumps achieve this by using a continuously rotating shaft [15]. In centrifugal pumps, a rotating impeller generates a centrifugal force that adds energy to the fluid. Types of centrifugal pumps are radial, axial, and vertical centrifugal pumps. This thesis focuses on radial centrifugal pumps as they are the most common. However, it is likely that the results are applicable to other centrifugal pumps because of their similar working principle.

The main components of a centrifugal pump are an impeller, a volute casing, a shaft, bearings, and a shaft sealing system [15]. Figure 2 shows a cross-section and the main components of a typical centrifugal pump. The shaft transfers the power from an electric motor to the impeller. The impeller is a rotor with a certain amount of vanes that forces the pumped liquid into the volute casing. The shaft rotates on top of the bearings and is sealed with the shaft sealing system in order to avoid the liquid from leaking out or air getting in. Parts of the pump not shown in Figure 2 are a base plate on top of which the pump is installed, and a coupler that connects the pump to the electric motor.

Important parameters of centrifugal pumps are head H, efficiency η, power P_h, and net positive suction head ($NPSH$). These characteristics are often expressed

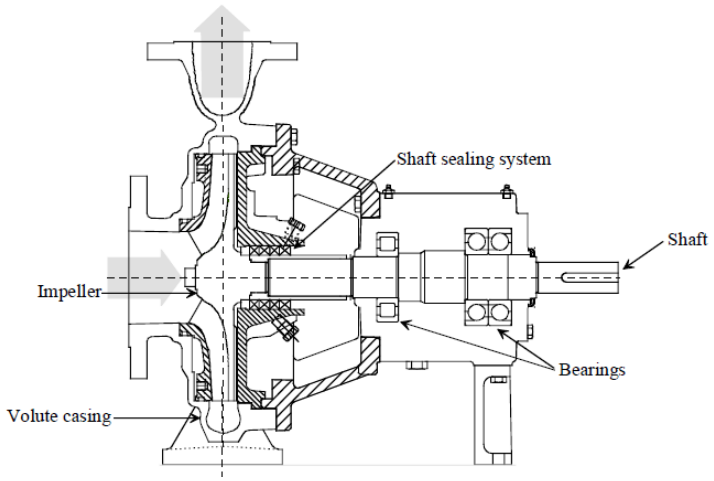

Figure 2: Cross-section of a typical centrifugal pump. Shaded arrows show the direction of the pumped fluid.

as a function of flow rate which is the volume of liquid pumped in a unit of time. The pump head is the specific energy increase expressed as equivalent liquid column height in meters. Efficiency is the ratio between the power pump provides to the liquid and the power pump gains from the electric motor. Power is the power needed to run the pump on a given operating point. $NPSH$ measures the pressure on the suction side and is discussed further in section 2.2.1 as it is closely related to a cavitation phenomenon. Pump performance is commonly expressed graphically with pump curves that depict these quantities. Figure 3 shows an example of how these parameters change as the flow rate increases.

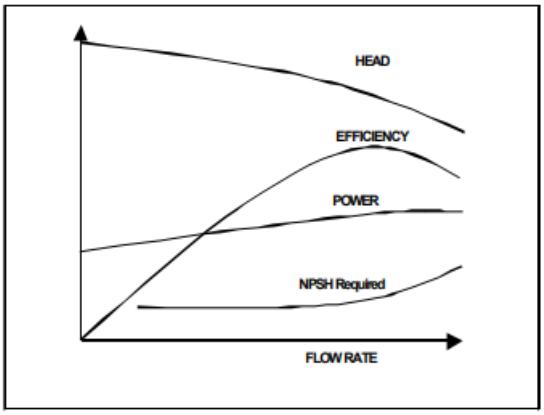

Figure 3: Rotodynamic pump performance curves [11]

Pump curves in Figure 3 have the same rotational speed. Changing the pump rotational speed affects the pump curves. The affinity laws describe how flow, head, and power change in proportion to rotational speed. The affinity laws are presented in equations (1), (2), and (3), where N is the rotational speed of the pump, Q is the flowrate, H is the head, and P is the power.

$$Q \propto N \tag{1}$$

$$H \propto N^2 \tag{2}$$

$$P \propto N^3 \tag{3}$$

The Equations (1)-(3) show that the flow rate is directly proportional to the speed, the head is directly proportional to the square of the speed, and the required power is proportional to the square of the speed. The efficiency of the pump is independent of rotational speed on large pumps [16]. In smaller pumps efficiency decreases as speed decreases. Affinity laws demonstrate that even small changes in speed give significant changes in head and power. Multiple pump curves with different speeds are usually displayed in the same graph as shown in Figure 4.

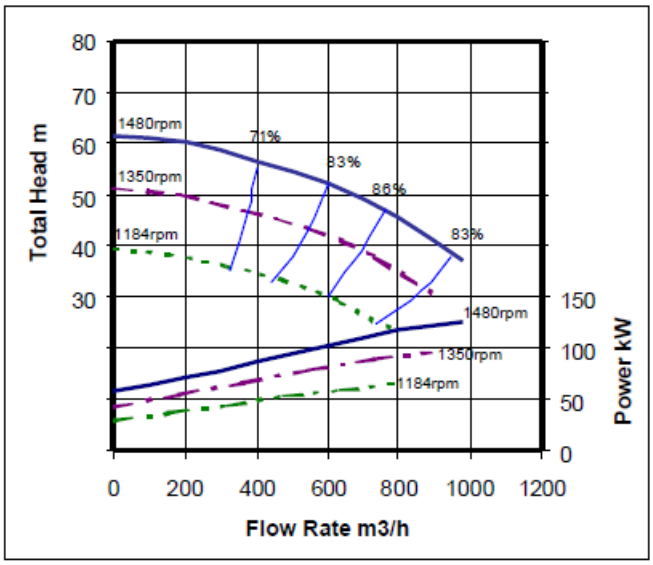

Figure 4: Pump performance speed variation. [11]

An operating point of a centrifugal pump depends on the system it is installed in. In a pumping system, the centrifugal pump head needs to overcome the systems head losses, which defines the operating point of the pump in a given system. The system head is a combination of a static head and a dynamic head. The static head is the height difference between a source and a destination reservoir and is indifferent to the flow. In a closed-loop system, there is no static head. The dynamic head is caused by friction in a pipeline and increases as the flow rate increases. Therefore, the dynamic head, and by extension, the system head, can be expressed as a function of the flow rate. The resulting plot is called a system curve. Figure 5 shows the pump and the system head curve. The operating point of the pump is where the system and pump head curve meet [11].

Often pumping systems require the operation point to be tuned in order to vary the flow rate or head. The operating point can be changed by either changing the

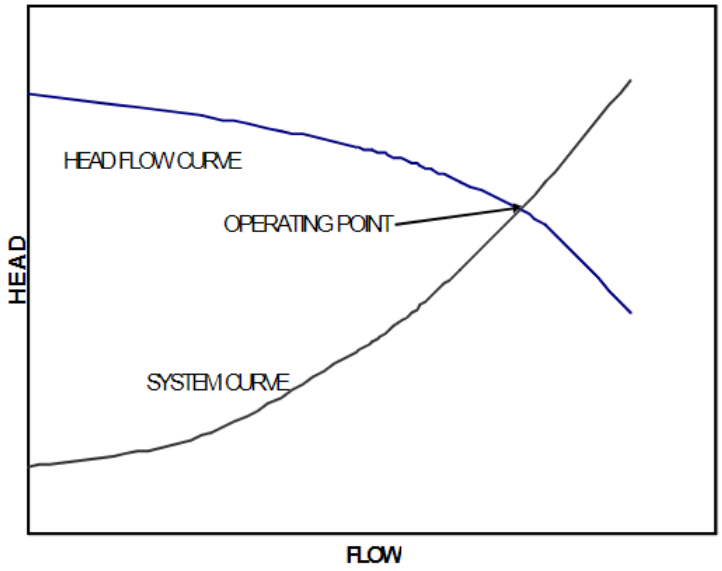

Figure 5: Typical rotodynamic pump head curve superimposed with the system curve [11]. The operating point is where the curves meet.

system curve or the pump curve. The system curve can be changed by changing the system dynamic head with, for example, control valves. The pump curve can be changed by varying the rotational speed, which affects the flow and the head as shown in Figure 4. Pump speed control is done with a VFD.

2.1.2 Variable Frequency Drive

A VFD is a solid-state power conversion system that is used to control the speed and torque of an AC motor by varying input voltage and frequency. The VFD is also known as an inverter, frequency converter, variable voltage, or variable frequency drive (VVVF).

In a pumping system, VFDs are used to reduce energy consumption and the need for maintenance. Because pumps usually need to operate in different operating points, the pump size is chosen for the largest possible load [11]. As a result, pumps are often oversized for their normal operation and therefore consume more energy than is necessary. However, with speed control, the centrifugal pump operating point can be adjusted to a more energy efficient point. As previously presented in section 2.1.1, affinity laws demonstrate that even small changes in speed affect the required power significantly. Therefore, using a VFD can bring meaningful energy savings. Additionally, operating a pump at optimal speed prolongs its lifespan and lessens the need for maintenance. Consequently, the use of VFDs in pumping systems is commonplace.

A VFD controls the motor speed by changing the frequency gained from the power supply [11]. This works, because the electric motor's synchronous speed depends on the power supply frequency and the number of poles. Equation (4) shows this

relationship, where v is the synchronous speed of the motor (rpm), f is the frequency of the power supply (Hz), and n is the number of poles of in the electric motor stator. The Equation shows, that by changing the frequency we can change the motor speed.

$$v = \frac{120f}{n} \tag{4}$$

A VFD is comprised of three parts: an input converter, a DC bus, and an output inverter [11]. Figure 6 shows the VFD circuit diagram and parts of the VFD. The input converter converts the supply voltage from AC to DC. The resulting DC current is uneven and is therefore smoothed with the DC bus. Lastly, the output inverter converts the DC supply back into AC with a given voltage and frequency that is fed to the motor. DC is turned into AC by pulse width modulation (PWM), where negative and positive DC legs are switched to simulate the required frequency and voltage (See bottom left of figure 6)

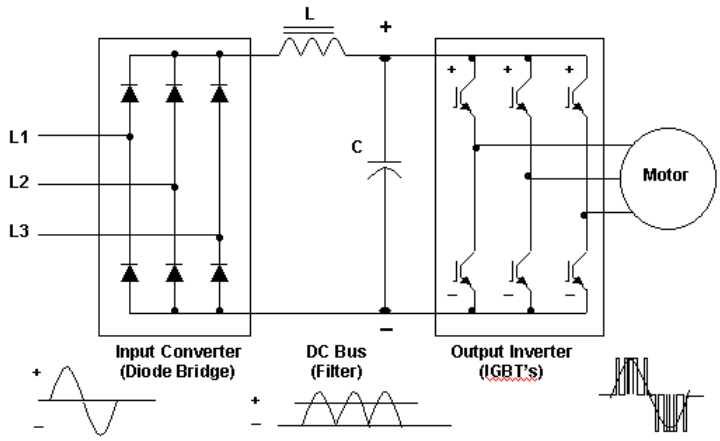

Figure 6: VFD diagram with the main parts and how the voltage changes [17].

2.2 Condition Monitoring Based on Variable Frequency Drive

Condition monitoring is the process of measuring a condition parameter of machinery over time, in order to identify a significant change which is indicative of a developing fault [4]. The usual parameters are vibration, temperature, and power. By monitoring these parameters over time, their development can be predicted. This way, condition monitoring allows faults to be anticipated before they develop into major failures. This is a type of maintenance technique known as preventive maintenance.

Condition monitoring is an important part of predictive maintenance. In predictive maintenance, the maintenance procedures are timed according to the true machine condition [18]. This is in contrast to preventive maintenance where maintenance is scheduled ahead of time, based on the average deterioration times of components. Monitoring the condition of the pump rules out unnecessary maintenance tasks and the procedures can be done when it has minimal impact. Other benefits include:

preventing unexpected failures, increased safety, positive environmental impact, and optimal spare part usage, all of which can offer significant savings.

The downside of deploying a condition monitoring system is the cost. The condition parameters are usually monitored with the use of sensors that require installation and cabling. Additionally, the measured data needs processing hardware, software, and personnel [19]. In addition to instrumentation, predictive maintenance requires personnel time for data measuring and analysis. Therefore, condition monitoring implementation costs can outweigh the benefits.

VFD as a source of condition monitoring data offers a possible cost-cutting measure. VFDs use and generate large quantities of data that could be used for condition monitoring [19]. As VFD can estimate motor rotational speed and torque, it offers a sensorless method for measuring condition parameters [12]. This data is free if the pump is already driven with a VFD and reduces the hardware requirement for condition monitoring.

The following sections discuss centrifugal pump faults this thesis focuses on. The focus is on the fault states as a phenomenon and how data from VFD is or could be used to detect them.

2.2.1 Cavitation

Cavitation is a common fault in pumps. Cavitation is a phenomenon where vapour bubbles form and collapse in the liquid due to reduced and subsequently increased pressure [20]. The collapse causes mechanical wear that damages the pump impeller, volute case, and pipes. Other pump components are also affected via vibrations. The formation of vapour bubbles is analogous to boiling. The boiling point of water depends on temperature and pressure. The lower the pressure, the lower the boiling temperature. While cavitation can also occur from an increase in temperature, the decrease in pressure is the more common cause.

The potential for cavitation can be expressed in terms of cavitation parameters. The typical cavitation parameter is $NPSH$. $NPSH$ is the inlet total head above the head equivalent to the vapour pressure relative to the $NPSH$ datum plane [21]. $NPSH$ datum plane is determined by the manufacturer, the common ones being the suction nozzle centerline or the impeller centerline. As the liquid enters the impeller the pressure drops. If the pressure is below the vapour pressure of the liquid, the pump cavitates. $NPSH$ can be calculated using Equation 5, where H is the pump total head, z_D is the difference between $NPSH$ datum plane and horizontal plane used as a datum for height measurement, p_{amb} is upstream total pressure, p_v is vapour pressure, ρ is the density of the fluid, and g is the acceleration due to gravity [21].

$$NPSH = H_1 - z_D + \frac{p_{amb} - p_v}{\rho g} \tag{5}$$

$NPSH$ is often expressed as available ($NPSH_A$) and required $NPSH$ ($NPSH_R$). $NPSH_A$ is the amount that the pump suction exceeds the vapour pressure and $NPSH_R$ is the suction needed to keep the amount of cavitation safe [11]. $NPSH_R$

is a pump property quoted by a pump manufacturer. It is measured by testing the pump under conditions of constant flow and observing the pump head as $NPSH$ is gradually reduced. NPSHR is often the same as $NPSH3$, ie. the $NPSH$ value where the pump head is reduced by 3%. However pump manufacturers can define NPSHR value differently, which is usually higher than $NPSH3$. The pump head - $NPSH$ curve is presented in Figure 7. First, the NPSH value is high. By the time the pump head is affected, the cavitation is fully developed.

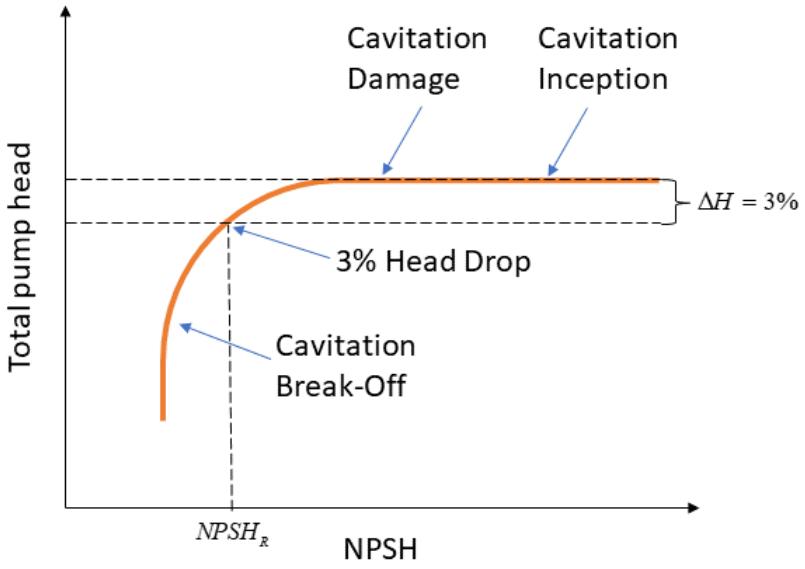

Figure 7: $NPSH_R$ for a given flow (Edited [22])

However, cavitation can happen and cause damage before the pump head drops. The first appearance of cavitation happens when the pressure at the eye of the impeller drops to or below liquid vapour pressure. This is called cavitation inception and does not affect the pump head at first but can cause damage [22]. On the other end, when the pump head has significantly reduced, is the cavitation break-off point. After cavitation break-off fully developed cavitation flow works as a cushion that diminishes damage. Therefore the maximum damage exists somewhere between inception and head breakdown. See Figure 7 for the points of cavitation inception, damage, and break-off.

Traditionally, cavitation is commonly detected with vibration sensors. Cavitation can be diagnosed as high-frequency broadband random vibration [20]. Additionally, cavitation can increase the amplitude of blade pass frequency and its harmonics. Cavitation can also be heard as a sound that resembles gravel passing through the pump. Therefore, acoustic sensors could also be used to detect cavitation [23].

There have been studies on detecting cavitation using VFD data. Ahonen et al. proposed a cavitation detection method in which rotational speed and torque in a normal state are measured and compared to new data [24]. In their method, root mean square values of rotational speed $n_{RMS,N}$ and torque $T_{RMS,N}$ were measured in normal operating state. Based on $n_{RMS,N}$ and $T_{RMS,N}$ a threshold value can be

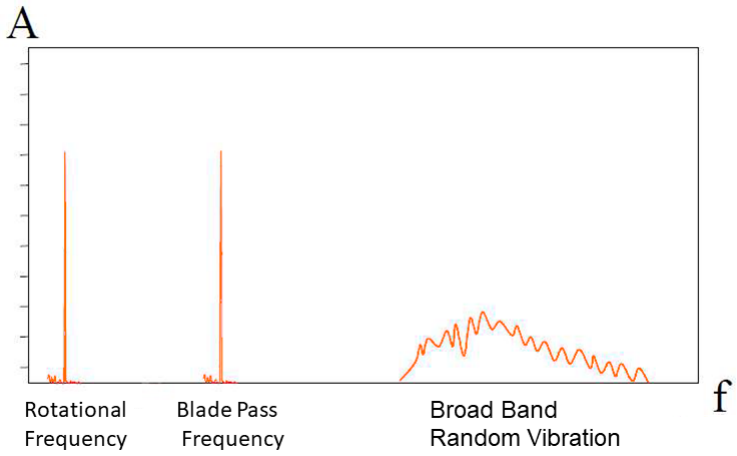

Figure 8: Cavitation frequencies [20].

determined over which the pump cavitates. An example of threshold value given in their article was 1.5-2 times the normal RMS value. Present RMS values can then be measured and compared to the threshold values. They found that with the right threshold values cavitation could be detected once it affects the pump head, and the occurrence of head-decreasing cavitation increased the time-domain variation of the rotational speed and the shaft torque estimates.

A similar method to the one proposed by Ahonen et al. is in use for ABB´s anti-cavitation software built into the VFD [25, 26]. Their algorithm compares a measured torque RMS value to a pre-measured nominal torque RMS value. When cavitation is detected the software automatically adjusts the pump speed or stops the motor altogether if cavitation continues for too long. The only installation that the software needs is setting the operating parameters.

2.2.2 Abrasive Metal-Metal Contact

Abrasive metal-metal contact, also known as a rotor-stator rub, is one of the common faults observed in rotating machinery occurring as a result of tight clearance between rotating and stationary components [27]. Rotor–stator rub degrades the pump performance and can lead to damage. The usual results are damaged shaft sleeves, premature bearing failure, premature mechanical seal failure and, at worst, shaft failure.

In rotor-stator rub, the rotor contacts the static part causing vibration and damage. The contact can happen at multiple parts of the pump. Usual contact locations are: bearing or seal, between wear ring and impeller, or between impeller blades and volute casing [27]. Common contact locations are shown in Figure 9. In addition to multiple locations, the cause of rubbing can differ. The usual causes are improper installation, load, or maintenance. Examples of causes are misaligned pump and motor shafts, bent pump shaft, and shaft deflection. However, this thesis focuses on motor-stator rubbing caused by shaft deflection.

Shaft deflection occurs when the velocity and pressure of a liquid being pushed

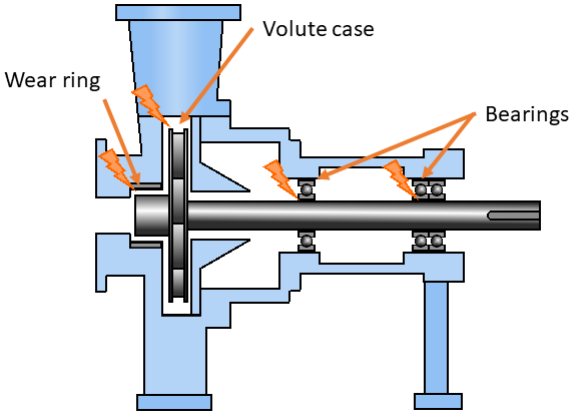

Figure 9: Contact locations on the pump.

by an impeller is not equal at all points around the impeller [28]. The difference in pressure causes radial forces that are enough to bend the pump shaft. The difference in pressure occurs when the pump operates outside of its design point. The pressure difference builds up depending on which side the operating point is from the design point. The directions are given in reference to the cutwater point, ie., the area just above the discharge. If the pump operates left from the design point the pressure increases in 180° to 270° area and radial force points in 60° from the cutwater. Conversely, pressure rises after cutwater if the operating point is on the right from the design point, and radial force points to 240° from the cutwater. Scratches can be detected from the static elements in the directions of the radial forces. Radial force directions and pressure differences are presented in Figure 10.

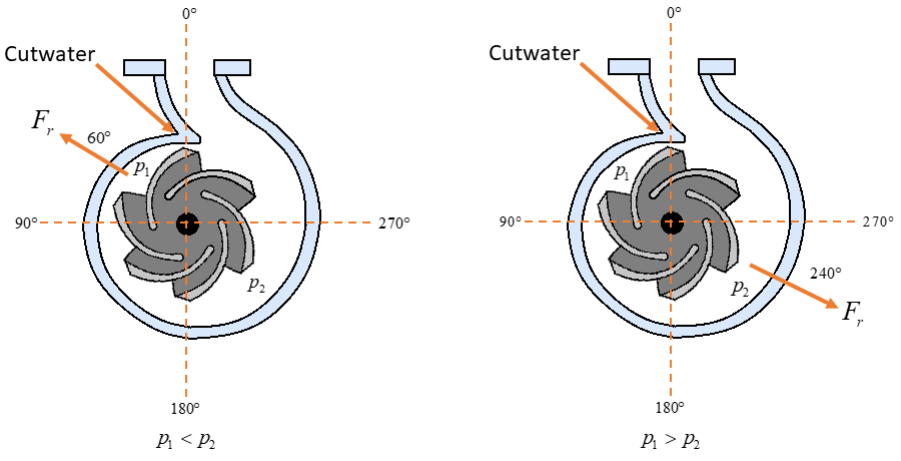

Figure 10: Radial force directions.

Radial force and shaft deflection can be calculated. Equation (6) shows the impeller radial force F_R, where k_R is the radial thrust factor, ρ is the density of the pumped liquid, H is head, S is specific gravity, d_2 is impeller diameter, and b_{2tot} is impeller outlet width [29]. The thrust factor k_R is dictated by rotating speed and

impeller type and is determined empirically.

$$F_R = k_R \rho g H d_2 b_{2tot} \tag{6}$$

The deflection v_d can be calculated with modified beam formula shown in Equation (7), where F is the load force, l is the shaft overhang, E is Young's modulus, and I is the second moment of area [30]. The contact happens when the deflection is larger than the clearance. Additionally, the looseness of the system has to be taken into account.

$$v_d = \frac{F l^3}{3EI} \tag{7}$$

Traditionally, rub can be detected with vibration measurements. Contact causes an increase in rotational speed frequency amplitude, its harmonics and high-frequency vibration [20]. It can be easily mixed with bearing faults as bearing component frequencies can also increase. Figure 11 demonstrates the usual vibration frequencies the contact causes in the frequency domain.

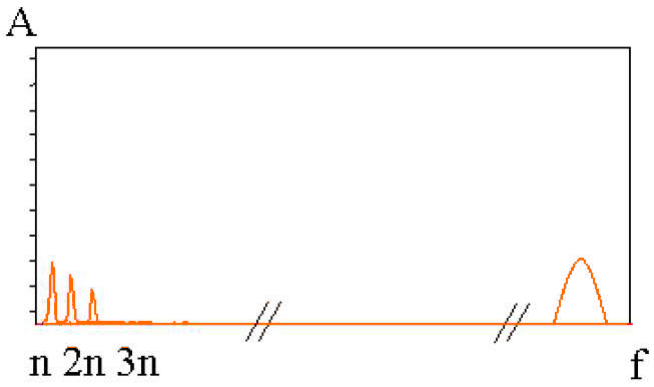

Figure 11: Abrassive metal-metal contact vibration frequencies [20].

There are no studies specifically on detecting abrasive metal contact from VFD data. However, studies on closely linked subjects point to the possibility of diagnosing contact faults. For example, a study by Wiedenburg et al. claims that a clear representation of the dynamic phenomena on the shaft can be calculated from current and voltage signatures [31]. Taking into account the specific type of contact shaft deflection causes, it is very likely to show in VFD data.

2.3 Artificial Intelligence

Machine learning (ML) discipline studies computer algorithms that can automatically learn and improve through experience. ML is regarded as a branch of artificial intelligence (AI) and encompasses numerous approaches and methods. The relationship between AI, ML, and other types of learning is presented in Figure 12 and will be explored in this subsection. Furthermore, this subsection reviews ML-based applications for machine health monitoring. The chapter encompasses reviews of

different AI-based health monitoring approaches and examples of where and how they are used.

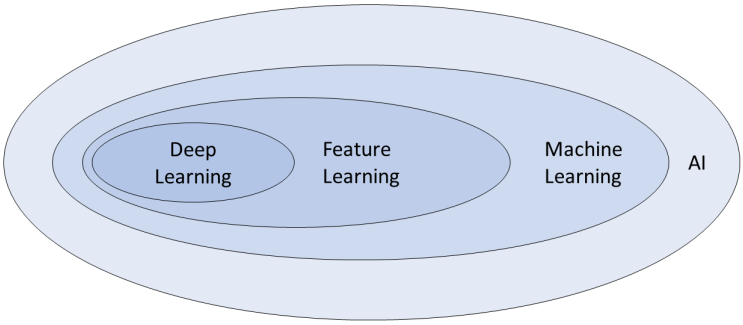

Figure 12: A venn diagram showing how deep and machine learning relates to AI (Modified [5])

AI-based machine health monitoring methods have been developed to automate the condition monitoring and diagnosis process. Numerous methods have been developed to achieve this goal, many of which implement ML, also known as data-driven approaches. With better access to data and improved computing power, ML methods have become increasingly feasible to implement. To help to analyze ML in the context of AI-based health monitoring, Figure 13 presents the different AI methods used for condition monitoring and how they are related in the hierarchy of AI.

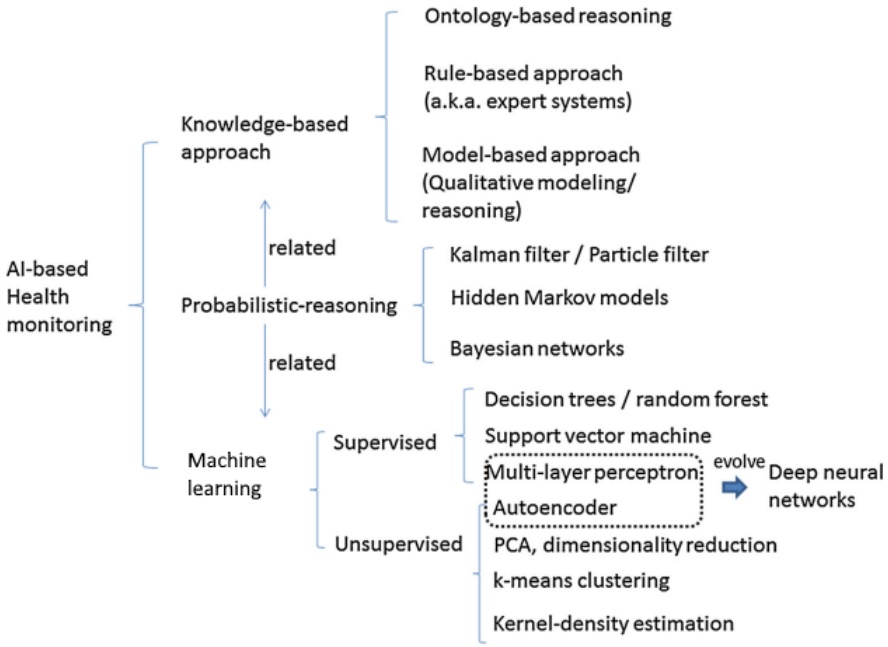

Figure 13: Categories of AI based condition monitoring methods [7]

The following sections explore the methods and concepts presented in Figure 13.

2.3.1 Knowledge-based Methods

Knowledge-based methods are an attempt to emulate human expert reasoning with software [7]. Figure 13 shows that ML approaches are defined distinctly from knowledge-based approaches. At simplest, the knowledge-based methods can be boiled down to if-else statements or flowcharts.

Knowledge-based condition monitoring approaches include based on mathematical models that describe real-life systems. Mathematical models use sensor data as an input for an existing mathematical model of the system and the system condition is attained as an output. Examples of knowledge-based methods are ontology-based reasoning [32], expert systems [33], and qualitative modelling [34]. The common factor between these methods is the need for knowledge of the monitored system. Implementing these methods need knowledge of key process parameters and how they influence the machine condition.

Knowledge-based methods can be effective and accurate for condition monitoring if mathematical and fault models are available [7]. For example, knowledge-based condition monitoring systems have been successfully used in digital twins, where a virtual copy is made from the model and its wear is emulated using collected sensor data [35]. However, there might not be enough expert knowledge available to implement these kinds of health monitoring systems. Knowledge-based approaches require a lot of human labour and vast knowledge of the application domain [36]. It may be the case that there are either no clear-cut rules to follow or accurate mathematical models available. In cases like these data-driven approaches or machine learning might be used.

2.3.2 Machine learning

In contrast to knowledge-based methods, ML methods build a model to map the rules and correlations from the input space to the output space [37]. Therefore, ML methods do not need a hand-designed model based on prior knowledge of the underlying physics. Examples of ML models are decision trees, artificial neural networks (ANN), and support vector machines (SVM).

ML algorithms build the model based on training data that consists of labels and features [37]. Labels are classes or values the model is trained to output. Features are the model input which in classic ML are hand designed. Labels and features are discussed more in detail in section 2.4.1. An important part of ML is feature design. Feature design is the process of selecting the input features for the ML model. This step requires expert knowledge for selecting the relevant features from the measured data [36]. In pump condition monitoring, the selected features could be, for example, the rotational speed, temperature, and blade pass frequency.

Feature design often includes feature extraction. Feature extraction is done to reduce the amount of irrelevant or duplicate information. In feature extraction, hand-designed features are condensed into fewer features that retain the information of the original features. Methods for feature extraction are for example principle component analysis (PCA) [38], latent Dirichlet allocation (LDA) [39], and independent component analysis (ICA) [40]. The extracted features form a data representation

that is used to train the ML model. The success of machine learning models depends heavily on the data representation [5, 36]. If crucial features have been left out in the feature design, there is little hope for an accurate model.

Depending on if the labels of the data points are known the learning can be divided into supervised, unsupervised, and semi-supervised [37]. In supervised learning, the dataset is completely labelled. Therefore, the ML algorithm is trained with known inputs and desired outputs. Conversely, in unsupervised learning, there is no labelled training data. Usually, the reason why ML is not implemented is not the lack of data but the lack of labelled data. Data itself is usually plentiful, but the labelling requires dedicated tests or manual sorting. Using unsupervised learning the algorithm has to discover the similarities between data points. Pattern recognition can be the goal itself or just one step in the ML model. Semi-supervised learning is a mix of labelled and unlabeled data.

2.3.3 Deep Learning

A DL algorithm not only generates the model but also finds the features from the input data. DL methods evolved from ANNs which are now commonly known as shallow neural networks. Deep neural networks (DNN) consist of multiple connected layers of neurons. It is this deep layer structure that differentiates DL from ANN. DL encompasses several variants such as auto-encoders, multi-layer perceptron (MLP), convolutional neural networks (CNN), and recurrent neural networks (RNN). The multi-layer architecture makes the training process faster. Instead of hand-picked features, DNN uses raw data as input and learns to find the features by itself. This is called feature learning. Studies show that using feature learning often yields better results than hand-picking the features [5].

DL has many advantages when compared to more traditional ML methods. DL enables the AI system to adapt to new tasks with little human intervention, which is more difficult for classic ML [5]. Another benefit DL methods have is that the process of the feature extraction and training are interconnected and therefore can be easily optimized [36]. However, DL has its disadvantages. There seems to be a trade-off between model generality and accuracy [7]. DL models are often large and complex, which demand large amounts of data, computation time and resources for the training, and the final model processing time.

Figure 14 sums up the differences between rule-based systems, classic ML, and DL. In the Figure 14, the shaded boxes are the components that the ML algorithm is able to extract from the data. Figure 14 demonstrates that the further right ones need less human input. The amount of human input is inversely proportional to the amount of training data needed.

2.4 Data

ML model learns from the data which makes it a crucial component. ML models success depends heavily on the quantity and quality of the data. Usual problem with ML applications is the lack of labelled data. This chapter will focus on how the data

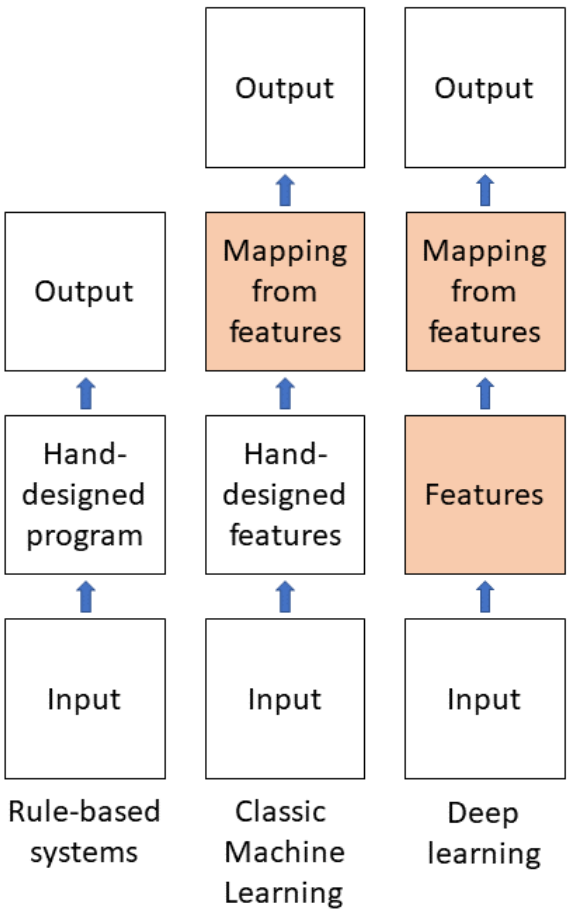

Figure 14: Flowchart of different AI systems. Shaded boxes are the components that the algorithm is able to extract from the data [5].

is structured, obtained, processed, augmented, and what are the characteristics of quality dataset.

2.4.1 Dataset

Dataset is a key component of any DL application. Dataset is a collection of data stored in a digital format that can be used to train the DL model. Data can be from one or multiple sources, for example sensors in different orientation. In supervised learning, dataset consists of feature and label pairs. Features are the input for the model and labels are the known desired output.

In machine condition monitoring, features are usually samples of time series data. Time series can be regarded as 1D data consisting of signal amplitudes measured at a regular frequency. In a dataset, the sample is represented as a vector referred to as a feature vector. Time series has a distinct property that it contains temporal dependencies that might show up in different points of time. This property makes learning more challenging [41].

Labels are the quantity of interest of the data point, or in other words, the property that the ML model is trained to compute [37]. Labels can have numerical or categorical values, which dictate the nature of the ML problem. ML methods aimed to output numerical labels are called regression methods and categorical labels classification methods.

Classification problems are often divided into binary, multi-class, and multi-label classification. In binary classification, the data point belongs to one out of two categories with a single label y being either 0 or 1. Multi-class classification is similar to binary, but there are more than two possible categories. Label values for multi-class classification are presented as $y = [1, 2, ..., K]$, where K is the number of different categories. If there is a possibility for a data point to belong to more than one category simultaneously, classification is said to be a multi-label classification. In multi-label classification several labels are used for different categories [37]. For example, in multi-label classification labels can be presented as $y_1, y_2, ...$, where label y_i represents a single category and is 1 if the data point belongs to it or 0 if it does not [42].

Before training the model dataset is split into three subsets: training, validation, and testing set. A training set is used as input in the training. A validation set is used in the training stage to monitor the progression of the training. This is done by giving the trained DNN validation set features as input and comparing the resulting output to the true label. The testing set is not used for the training at all and to give a final unbiased representation of the model's performance. This is called model loss and it will be further discussed in section 2.6.3. Once the loss is deemed low enough training can be stopped and the testing set is used to give a final evaluation of the model. The testing set is needed because the validation set is used in training and can therefore introduce bias to the model. Figure 15 visualises the dataset split.

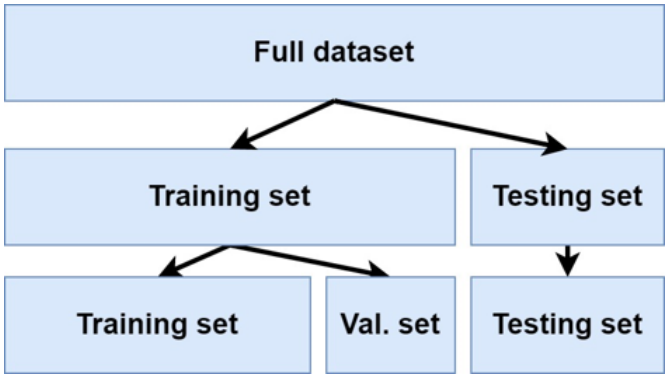

Figure 15: Dataset split.

2.4.2 Data Acquisition

Training a DL model needs a large amount of data. Dataset can be acquired from pre-existing data or be measured and recorded specifically for ML application. This

section discusses things to consider when forming the dataset. These include the number of samples needed, quality of data, dataset balance, and data representation.

DL models need large datasets to learn and perform well. There is no way of knowing beforehand how much data is needed for effective learning. Estimations for the amount of data needed can vary wildly. For example, Oyedare and Park estimate that for transmitter classification 1 000 to 30 000 samples are required per class [43]. Additionally, it is acknowledged that the more data the better and complicated applications need larger datasets [44]. This is always not the case and complicated problems require larger datasets. The number of trainable parameters can be changed with the model architecture and more samples can be generated with data augmentation. Model architecture and amount of trainable parameters will be discussed in section 2.5 and data augmentation in 2.4.3.

Having a quality dataset is a crucial component of any DL method. If the dataset has a large number of errors, outliers, and noise the trained model struggles to learn the true connection between input and output [45]. There are many reasons for poor quality data like broken sensors, human error, and corrupt data [7]. Therefore, dataset cleaning is a usual step taken before training the model. Cleaning includes discarding or manually fixing outliers and unusable data. Due to the large size of the dataset, cleaning is a time consuming process.

Another aspect to consider is the balance of the dataset [5]. Balance refers to the label distribution in the dataset. An unbalanced dataset has more samples from one label than another, which causes problems in the model training. For example, a dataset consisting of 90% of one label will achieve 90% accuracy if it always outputs that label. The easiest solution is to add more samples on the minority label, but this is not always feasible. Data for one label might be significantly harder to produce than others. Therefore, the dataset should be balanced by either augmenting more samples or giving weights for labels in a loss function.

An important aspect of the dataset is that the samples are representative of the new cases it is wanted to generalize into. A large dataset in itself does not help if the dataset does not represent the application it is used for [7]. Dataset can be non-representative if it is measured in exceptional or not varied enough conditions. For example, the pump assembly might be exceptional so that the trained model performs accurately only with that setup. Another example, data could be measured with a pump with the same rotational speed which might result in bad performance with any other speed. The non-representative dataset will lead to bad generalization even though the model might learn well on the dataset itself [45]. This is why possible variability and possible biases must be taken into account when gathering the dataset.

2.4.3 Data Augmentation

The goal of data augmentation is to artificially generate more data from existing samples. In practice, these methods take data points and transform them in some way. Ideally, the transformations do not affect the underlying properties of the data and therefore do not warp the model learning. Augmenting data lessens the risk of the model overfitting by increasing the number of data points. Data augmentation

methods also offer a way to mitigate dataset imbalance by oversampling the minority label data points [46]. For time series data, the usual augmentation methods include window slicing, window warping, flipping, and noise injection.

Window slicing is a data augmentation method where slices are extracted from the time series [47]. All the resulting slices have the same label as the sample they were extracted. To conduct window slicing the size of the windows and the step size must be determined. The step determines from where the next window starts as pictured in Figure 16. In order to generate more samples with the window slicing, the step size must be lower than the window size, which means that the resulting samples overlap. An additional benefit of using window slicing is that the length of the time series data does not matter, as the samples can be cut into equal lengths [47]. The ML model is trained using the slices.

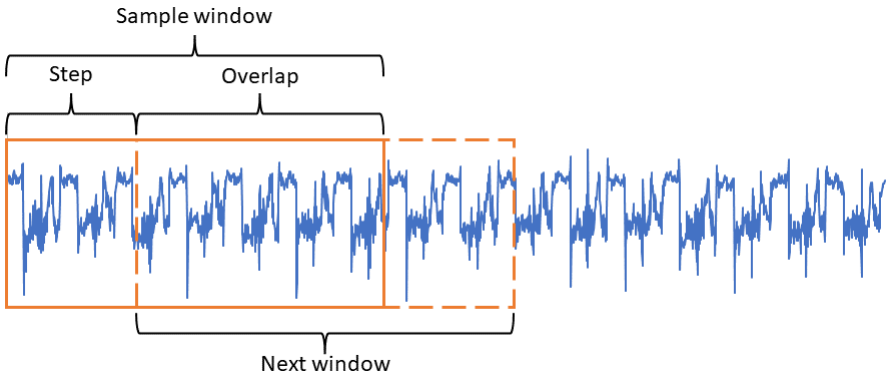

Figure 16: window slicing

Window warping is a time series specific augmentation method. In window warping the time series is either extended (upsample) or condensed (downsample) while the time range is kept the same [48]. Window warping is demonstrated in Figure 17. Up and downsampling changes the time series length which is why it is used in conjunction with window slicing in DL applications. Le Guennec et al. found that window warping improved CNN model performance on most UCR time series datasets [49].

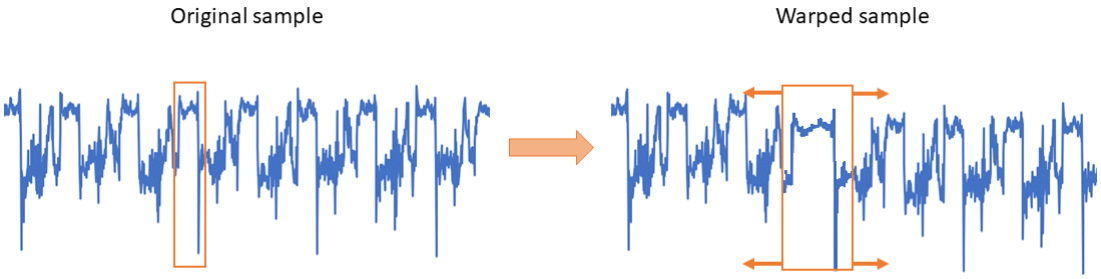

Figure 17: Window warping data augmentation method.

Flipping is a simple way to generate new data points. In the flipping data

augmentation method the original data is flipped by changing the sign of the original signal [48]. The label of the flipped data point is kept the same. Flipping is presented in Figure 18.

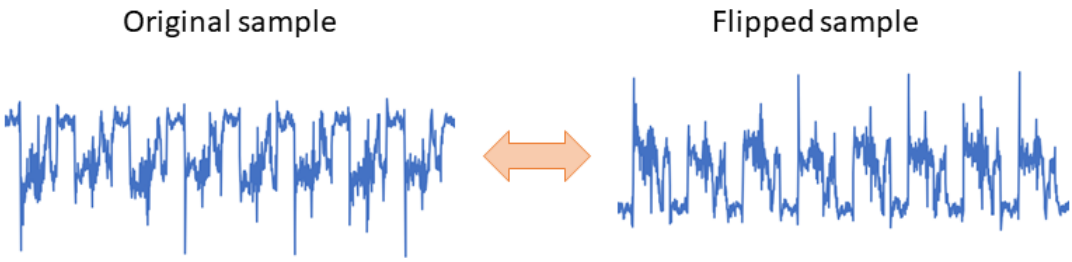

Figure 18: Flipping augmentation method.

Noise injection is a data augmentation method where noise is added in a random manner to the dataset. For example, data can be injected with gaussian noise, spike, step-like trend, and slope-like trend [48]. Great care should be taken when implementing noise injection. The application domain must be kept in mind when implementing noise injection. For example, if the data is expected to have periodic patterns the injected noise should not disturb it. Also, noise injection must be label invariant so that the model cannot predict the label based on the added noise [50].

2.5 Deep Learning Architectures for Condition Monitoring

There are many different DL architectures that have been developed for different kinds of situations and applications. Model selection is a crucial part of developing DL-based condition monitoring systems as other models are suited better to some applications than others. To help to understand and choose the right model for the given application, this chapter presents three DL models: MLP, autoencoder, and CNN.

2.5.1 Multilayer Perceptron

A MLP is a simple neural network, in which neurons are fully connected. MLP is a type of deep feedforward network. Deep feedforward networks are learning algorithms that loosely mimic brain neural structure [5]. Feedforward refers to the fact that the information flows from the input straight to the output. In contrast, a model with feedback connections, like RNN, the outputs are fed back to the model. Feedforward neural networks include a variety of DL models, like MLP, auto-encoders, and CNN. Therefore, most of the operating principles presented in this section also apply to other DL models. Variations of these basic structures will be presented in the model's own sections.

Deep feedforward networks are designed to approximate some function f^*. In the case of classification the function is $y = f^*(x)$, where x is the input and y is the class label. A feedforward network approximates some function f^* by defining a mapping

$\hat{y} = f(x;\theta)$, where $\hat{y}$ is the predicted label, and θ are adjustable parameters. The model learns the value of parameters θ that yield the best approximation.

In theory, a feedforward network with a single large enough hidden layer can represent any function [5]. However, the training does not always produce that function. Either the optimisation algorithm is unable to find the parameter functions or the training algorithm might choose the wrong function due to overfitting [5]. Deeper models reduce the number of training parameters.

Deep feedforward networks consist of different functions of a certain depth and width. Depth is the number of chained functions. For example, three functions in a chain form a network with a depth of 3: $f(x) = f_3(f_2(f_1(x)))$, where f_1 is the input layer, and the last layer. In this case, f_3 is the output layer. Layers between the first and output layer are called hidden layers. The width of the neural network is the dimensionality of the hidden layers.

In order to get good results with MLP the network needs to be quite deep which increases the required amount of training data significantly [5]. Because all the layers are connected, the number of trainable parameters grows significantly if the network is expanded. For example, Figure 19 a MLP with an input of size 3, two hidden layers of width 5, no bias term, and an input layer of width 3. This MLP has total $3 \cdot 5 + 5 \cdot 5 + 5 \cdot 3 = 55$ weights. If one more neuron is added to both hidden layers the total comes to 72, which is an increase of 30%. Often, the number of neurons and inputs is far larger.

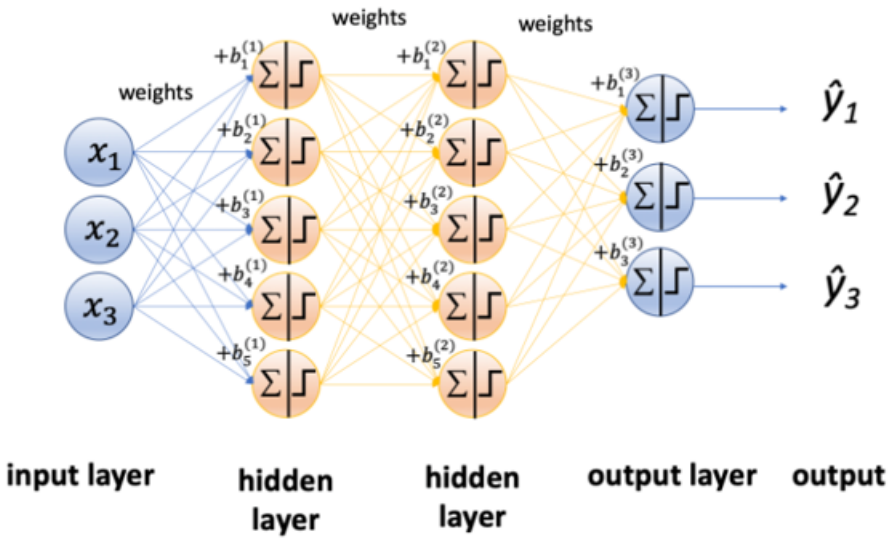

Figure 19: Deep neural network with four layers. The network has a dimensionality of five as dictated by the number of neurons in the hidden layer.

Layers in DNN consist of neurons that contain the trainable parameters [5]. A neuron calculates a weighted sum z of the input and computes an activation value with an activation function g. In some cases, the neuron can also have a bias term b. The activation value is calculated with Equation (8).

$$a = g(z) = g(b + w_1x_1 + w_2x_2 + w_3x_3) \tag{8}$$

The neurons weights and biases are the trainable parameters that are optimized during the training phase [5]. The input of each neuron depends on the DL model's architecture. In MLP the layers are fully connected and therefore the neuron receives all the previous layer's outputs as its input. Output a of the neuron is known as the activation value and is calculated with an activation function that is given the weighted sum of the inputs and bias. Figure 20 shows the function of the neuron.

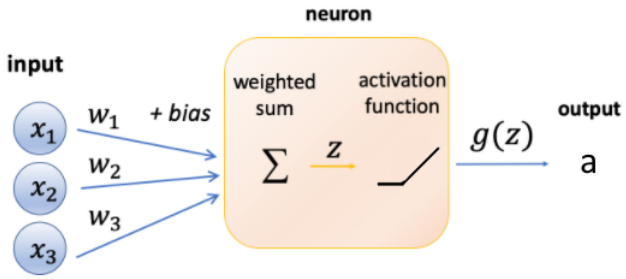

Figure 20: Neuron with three input values. Each of the input values has a trainable weight associated with it. The neuron calculates the weighted sum and activation value. The activation value is sent to the next layer or as an output.

Activation function g is chosen when constructing the DL model. In the hidden layers, the choice of activation function affects the learning of the model. In the output layer, the choice affects what type of values the model outputs. Therefore, activation functions are a crucial part of neural network design. Usually, all the hidden layers use the same activation functions [5, 42].

There are numerous activation functions and variations thereof. However, this study will focus only on the most common ones, namely identity, binary step, sigmoid, rectified linear unit (ReLU), and softmax function. Gradients of the activation functions are also important in the training phase because the model's weights are updated based on them [51]. The design of hidden units is an active area of research and does not have many guiding principles in place [5]. This makes selecting the activation function difficult as there is usually no way of knowing in advance which performs best in a given application.

The simplest activation function is an identity function, where the output is the weighted sum of the input [42]. Because the identity functions gradient is always one, the model can not update the weights, ie. learn, during the training. Therefore, the identity function is seldom used in hidden layers. However, the identity function can be implemented in an output layer for regression problems.

$$f_{identity}(x) = x \tag{9}$$

A binary step activation function gives only 1 or 0 as the activation value. This mimics the neurons of the brain that either fire or not. The binary step formula is

presented in Equation (10). The binary activation function is a natural choice for a binary classification output layer [42]. Similarly to the identity function binary step is not used in hidden layers as its gradient is always zero.

$$f_{step}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{10}$$

A common activation function is a sigmoid function. It is a continuous version of the binary step function and yields values between 1 and 0. The sigmoid function can be calculated using Equation (11). In contrast to identity and binary step functions, the sigmoid is a non-linear function, i.e., it has a varying gradient. The varying gradient makes it possible to train the model using a backpropagation algorithm, discussed more in section 2.6. Due to this, the sigmoid function can be used in hidden layers. A downside of a sigmoid function is that it suffers from vanishing and exploding gradients. The vanishing gradient occurs when in training a model the gradient terms are smaller than 1. In a deep architecture, these small gradients are multiplied many times resulting in smaller values that tend towards zero [52]. The opposite happens when the gradients are larger than one in which case the multiplied value tends to infinity i.e. the gradient explodes. This can lead to poor performance or stop the model from learning altogether. Therefore, the sigmoid function is not recommended for feedforward network's hidden layers but can be used in RNN hidden layers and as a binary classification output layer [5]. A common variant of the sigmoid function is the tanh activation function that is scaled so that it is symmetric around the origin [42].

$$f(x)_{sigmoid} = \frac{1}{1 + e^{-x}} \tag{11}$$

A ReLU activation function is currently the recommended activation function for most feedforward neural network hidden layers [5]. The ReLU function was designed to address the problem of vanishing and exploding gradients. The ReLU activation function is shown in equation (12). As can be seen from Equation (12), the neuron only activates if the input is positive, which makes it more computationally efficient than the sigmoid function. This also addresses the vanishing and exploding gradient problem as ReLU keeps the gradient either as 0 for negative or 1 for positive inputs. However, this introduces another problem called dying ReLU, where once the gradient gets a value of zero the neuron dies and will never activate again. To combat this, variations of the ReLU have been designed. These variations are leaky ReLU (LReLU), exponential linear unit (ELU), and scaled exponential linear unit (SELU) [52]. All these variants give small non-zero gradients for negative inputs to keep them from dying.

$$f_{ReLu}(x) = max(0, x) \tag{12}$$

A softmax activation function is used for multiclass classification output layer [42]. It can be thought of as a combination of multiple sigmoids that outputs vector values that sum to 1. The softmax activation function is presented in equation (13).

In a multiclass classification output layer, there are as many neurons as there are possible classes. The softmax returns the probability for a data point belonging to an individual class.

$$f_{softmax}(x) = \frac{e^x}{\sum e^x} \tag{13}$$

There are some studies on MLP for condition monitoring. MLP performs well on classification tasks and is therefore often used as a classifier while another method is used for feature extraction [53]. Feature extraction method can be another DNN like CNN, examples of which are presented in section 2.5.3. The following paragraphs introduce some applications of MLP for condition monitoring.

A study by Mishra and Huhtala presented an MLP-based fault detection method for an elevator system. The MLP was given 12 extracted features which derived from sensor data on motion and vibration of the elevator. The MLP itself consisted of two hidden layers each containing 20 neurons. The model detected a fault with 99% accuracy.

Orrú et al. used an MLP for a centrifugal pump fault prediction [54]. The approach aimed to predict the pump's fault progression. Dataset was collected from a centrifugal pump operating at a refinery and was measured from eight different sensors. From the sensor data, eight features were extracted to describe the remaining useful life. The study compared SVM and MLP methods classification performance on the measured dataset. The final accuracy the MLP achieved was 98.2%.

Al Tobi et al. used MLP for centrifugal pump fault diagnosis based on vibration data [55]. In their study, MLP was used for classification and feature extraction was done for both methods with discrete wavelet transform. A genetic algorithm was used to optimise the number of hidden layers and neurons. Algorithms were trained to classify six conditions, imbalance, bearing, cavitation, impeller, looseness, and normal. The best MLP model achieved 100% accuracy with 30 normalized features, from the feature extraction. Interestingly the MLP performance worsened with additional features.

2.5.2 Autoencoder

An autoencoder is a type of neural network that condenses its input and then replicates the input to its output [56]. This is done in order to find and extract compressed data representation from a given data set. Autoencoders are unsupervised learning methods as only input is needed in the training. However, the way the autoencoder uses its input could be regarded as it having labels, which is why autoencoders are sometimes called self-supervised methods. Autoencoders are an early attempt at feature learning, from which feature learning methods evolved [7]. Figure 21 presents a typical architecture of an autoencoder.

As seen in Figure 21 autoencoders are comprised of an input layer, an output layer, and one or more hidden layers. The structure forms an hourglass shape with a bottleneck in the middle. The first half of the hidden layers is the encoder and the later half is the decoder. The bottleneck makes it intentionally hard to

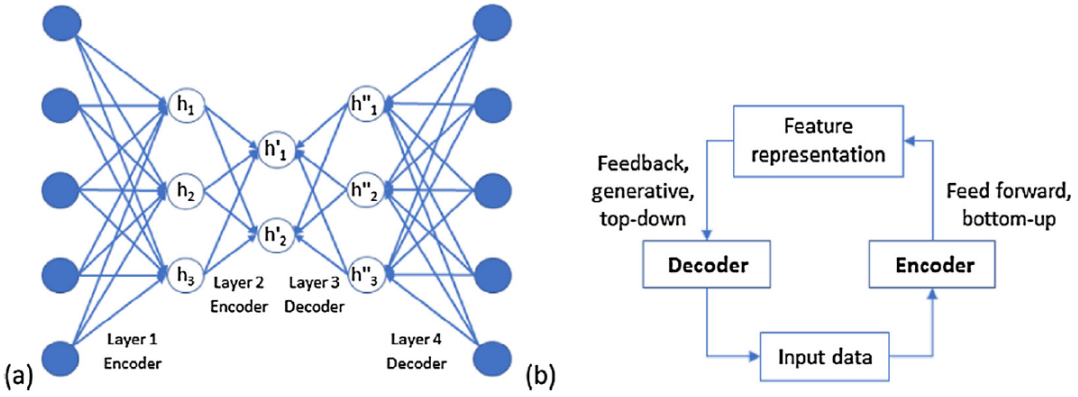

Figure 21: Architecture and operation of autoencoder [7]

reconstruct the input to form a representative compression. After the training, the resulting compressed representation can be found at the bottleneck. To improve performance, multiple autoencoders can be stacked back to back. This is usually done when the relationship between the dataset and features is complex. When the autoencoder is trained the decoder part of the neural network can be discarded. This way autoencoders can be used as the initial step in a deep learning architecture or as a feature design method for classic ML [57]. In this instant autoencoder works as a feature extractor and other deep learning method does the classification. Variations on auto encoder Restricted Boltzmann Machine, deep belief network and deep Boltzmann machine.

In machine health monitoring autoencoders are usually used for anomaly detection. As the encoder is trained to reproduce the input anomalies can be detected when the performance of the autoencoder worsens [58]. The reconstruction error of the autoencoder can be used as an anomaly score. This has the benefit that the autoencoder can be trained with data from the machine in healthy condition alone. This makes autoencoder easier to implement as it is usually healthy machine data that is simple to produce. Autoencoders have also been used for machine health diagnosis.

Usually, autoencoders are used to extract features from raw data and then the features are passed forward to a different neural network structure for classification. For example, Zhu et al.[59] used stacked autoencoders to extract features from hydraulic pump vibration data for unsupervised fault detection. They found the stacked autoencoder approach superior to traditional ML methods. Another study was conducted by Sun et al. who used autoencoders for bearing fault diagnosis [60]. In their study, they employed stacked autoencoders as an automatic feature extraction method. Extracted features were then forwarded to an MLP. Sun et al. found that compressed representation marginally increased the accuracy compared to a DNN was given raw vibration data, but the training time decreased significantly.

2.5.3 Convolutional Neural Network

CNN is a specialized feedforward deep neural network designed for processing images and time-series data that can be thought of as 2d and 1d grid structures [5]. CNNs are useful if the interest is to only know if a certain feature is present in the data and not where it is present. This property is called translation invariance, which makes CNN great for fault detection. CNNs were developed to overcome MLP shortcomings: a large number of trainable weights and the difficulties in recognising features if they appear in different points of the input. Convolutional layers address these problems by applying filters that detect features by computing activation values at evenly distributed locations. The activations form feature maps between layers. Because the trainable weights are shared in the filters and not between every neuron, the use of filters radically reduces the number of trainable weights. A network can be considered as CNN if it contains at least one convolutional layer. This section explores how a convolutional layer functions in the neural network

A typical convolution layer is comprised of three sub-layers: convolution, normalisation and pooling sublayer [5]. Because of the multiple sublayers, the convolution layer can be thought of as either one complex layer or three simple individual layers. Figure 22 presents the stages in the convolutional layer.

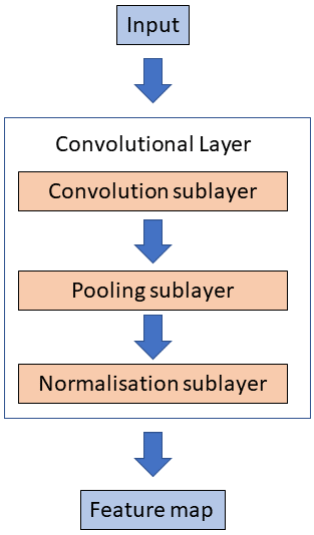

Figure 22: Convolution layer consists of three stages: convolution, pooling, and normalisation sublayers. In some sources, these stages are treated as individual layers in the network.

In the convolution sublayer, a filter computes crosscorrelations by being repeatedly applied to the input. The filter is a NxM with N kernels and M trainable weights. Figure 23 demonstrates the operation convolution sublayer. In this example, the filter has four weights and one kernel. One linear activation value is gained by taking a dot product of the kernel and the local area of the input which is dictated by the number of weights. This process is similar to the calculation of the weighted sum in the MLP. The filter is then applied repeatedly to a different local area of the input

by shifting the filter by the stride amount. In the example, in Figure 23 stride is one.

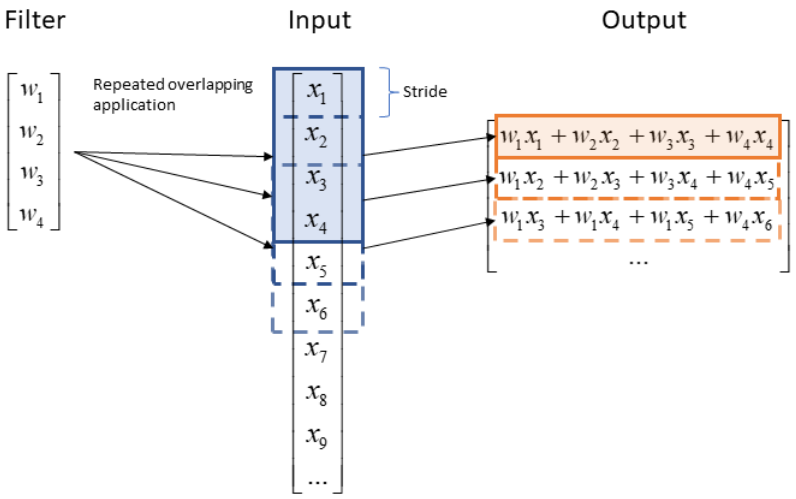

Figure 23: Convolution stage for 1D input. A filter of size 4 is repeatedly applied to the input. Between every application, the filter is shifted by the stride amount of 1. Output is a feature map that is forwarded into the activation stage.

Once the filter is applied to the whole input, the local area output values are forwarded to a nonlinear activation function. This is similar to the fully connected layers and uses the same activation functions, the usual one being Relu or its variant. The end of section 2.5.1 gives more information on activation functions and how they are used. The aim of this filter is to be trained to detect features and by sliding the filter over the whole input the features can be detected at any point. An additional benefit of filters is that it makes nodes share weights, which reduces the total number of trainable weights compared to fully connected layers. Usually, more than one filter is used per layer to detect different features. The output of the convolution layer is a feature map. In the feature map, detected features are imprinted to the location they were detected. This feature map is usually further processed with a pooling sublayer.

The next stage in the convolution layer is pooling. Even though the filter in the convolution stage can detect patterns anywhere in the input the resulting feature map is sensitive to translations of the detected pattern. Pooling helps to make the feature map more resilient to shifts [5]. Pooling downsamples the feature map, effectively creating a lower-resolution version. Because of the lower resolution, features have more leeway. In practice, the pooling operation is done by taking a patch of the feature map and condensing it into a single value. Two common pooling operations are average and max pooling. In average pooling, the output value is the mean of the patch and in max-pooling, the value is the largest value in the patch. Max pooling is the more common pooling operation as it has been shown to outperform average pooling [61]. Figure 24 presents a max pooling operation with a patch size of three.

The feature map can be further enhanced with batch normalization. Batch normalization is an additional step that reduces the effect of an internal covariate

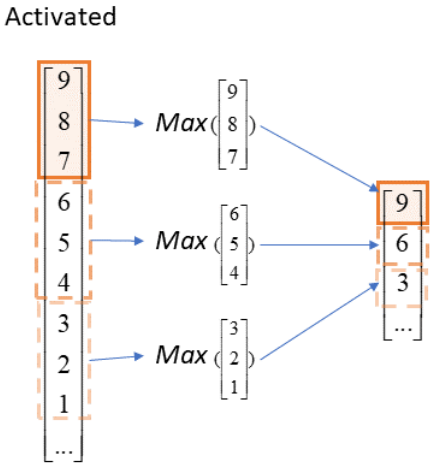

Figure 24: Pooling stage of the convolutional layer using max pooling with a patch size of 3. The result of the pooling stage is a lower-resolution version of the feature map.

shift. The internal covariate shift is the change in each layer's input distribution during training. Batch normalization can accelerate DNN training dramatically [62].

Usually, CNNs have multiple stacked convolutional layers. Filters in deeper layers can extract more abstract features. Usually, CNN architecture has multiple convolutional layers at the beginning and fully connected layers at the end. Fully connected layers are there to do the classification or regression based on the features extracted by the convolution layers.

As CNN are good for working with time series data they are commonly applied for machine fault diagnosis [7]. The following paragraphs introduce some applications of CNN for condition monitoring.

Zhang et al. applied CNN for rolling-element bearing fault diagnosis [63]. In their study, a CNN with wide first-layer filters (WDCNN) was used for intelligent fault diagnosis. Their model was trained with a dataset consisting of raw vibration data. The model employed five convolutional layers and two fully connected layers. Their model achieved 100% classification accuracy in bearing data set with ten different condition classes.

Mao et al. used CNN for gearbox condition monitoring using data fusion [64]. Fused data consisted of vibration data and thermal images. Their model trained with data fusion performed better than models trained with each data source separately.

Chen et al. used CNN for rolling bearing fault classification with data set collected from a self-priming centrifugal pump [65]. The pretrained model used for transfer learning was ResNet50, a 50-layer deep CNN for image classification. For the network, the fault signals were converted into time-frequency images by using a continuous wavelet transform. On the dataset collected from the centrifugal pump, their model achieved an accuracy of 99.98%.

In another study, Manikandan and Duraivelu applied CNN for vibration-based intelligent fault diagnosis for an industrial mono-block centrifugal pump. Their study

focused on broken impeller and seal failures. Similarly to the previous study, the vibration data was processed into 2D images. As a result, the CNN achieved an accuracy of 99.07%.

2.6 Training Neural Networks

The objective of training a supervised DNN is to make the models output labels match the known true labels. As the training progresses, the error between these labels gets smaller and the model is said to learn. In practice, learning is done by adjusting the models trainable weights so that the error, often called loss, decreases. In the training process, the trainable weights are repeatedly updated in small increments after which the loss is calculated. The training process is visualized in figure 25. This section will go over the steps shown in Figure 25 in detail. The process can be divided into five steps: initialization of the weights, feedforward pass, computing the loss, backwards pass, and Updating the weights. These steps are discussed in the following subsections.

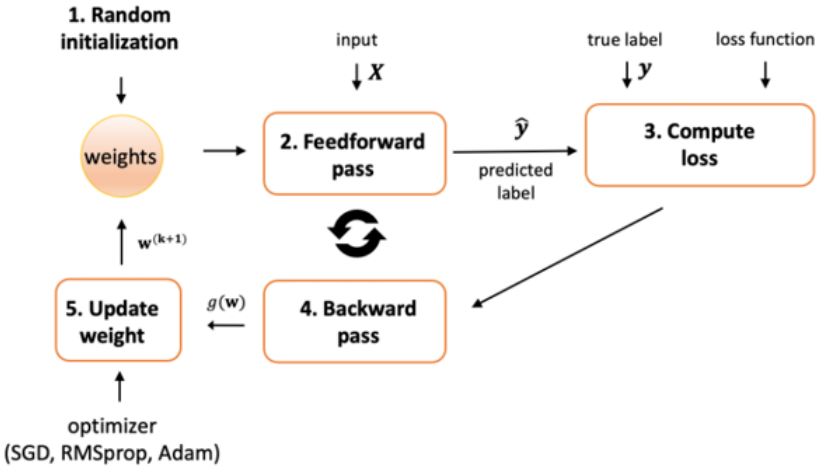

Figure 25: Training process of a deep learning model.

2.6.1 Initialization of the Weights

At the very beginning of the training, the network weights are initialised. The initialization method can affect model learning.

One way to randomly initialize weights is to draw them from Gaussian distribution [66]. However, a DNN model can have difficulties in converging with weights that are initialized with fixed standard deviation [67]. The use of fixed standard deviation with a non-linear activation function can lead to high or small initial activation values, which in turn result in exploding and vanishing gradients.

The non-linearity of the activation function can be taken into account with the use of a Kaiming initialization method. The Kaiming initialization is a zero-mean

Gaussian distribution whose standard deviation for layer l with n inputs is $\sqrt{2/n_l}$. [67]

Another weight initialization method is transfer learning. In this method, weights are taken from a pre-trained model and retrained with the new dataset [50]. Transfer learning can be used if a model exists for a different but related problem. It has the potential to significantly improve sample efficiency and reduce training time.

2.6.2 Feedforward Pass

After weight initialization, the feedforward pass is performed. The network is given a sample or batch of samples of the training dataset to attain the model output $\hat{y}$. Equations (14) to (19) show a mathematical representation of feedforward pass in a network with two hidden layers, where X is the input vector, W is the trainable weights, z is the neuron values, f is the activation function and $\hat{y}$ is the output. From the equations, we can see how the information flows and is passed to the next layer in the network.

$$X \quad \text{Input} \tag{14}$$
$$z_2 = \mathbf{W}_1 X \quad \text{1st hidden layer neuron value} \tag{15}$$
$$a_2 = f_1(z_2) \quad \text{1st hidden layer activation value} \tag{16}$$
$$z_3 = \mathbf{W}_2 a_2 \quad \text{2nd hidden layer neuron value} \tag{17}$$
$$a_3 = f_2(z_3) \quad \text{2nd hidden layer activation value} \tag{18}$$
$$\hat{y} = \mathbf{W}_3 a_3 \quad \text{Output layer} \tag{19}$$

2.6.3 Compute Loss

The next step in the training is calculating the loss. In the context of supervised ML, loss means the difference between true y and model output label $\hat{y}$. Loss is calculated using a loss function, which is also known as an error or cost function. The loss function is selected depending on the nature of the problem. For regression problems, the most used loss function is the mean squared error (MSE), shown in Equation (20). MSE calculates the difference between each individual $\hat{y}$ and y, squares it, adds all the losses together and returns the mean value.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{20}$$

However, in the case of a classification problem the MSE loss function can not be used as the labels do not have comparable numeric values. In classification tasks, the output of the model is a set of probabilities of the input belonging to each class. This distribution can be used to determine how certain predictions the model is making in which high certainty for the correct class means low loss. Therefore, a loss function is needed that compares probability distributions. The most common loss function in classification tasks is a cross-entropy loss. Cross-entropy is presented in Equation

(21), where $\hat{y}_i$ is the true class probability distribution and y_i is the model output probability distribution for i:th datapoint. A perfect cross-entropy loss score is 0.

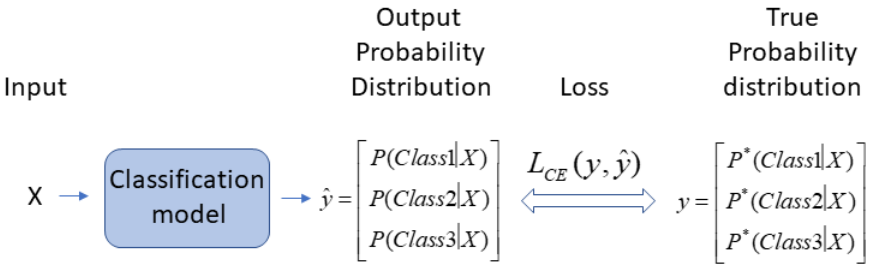

Figure 26: Crossentropy loss

$$L_{CE} = -\sum_{i=1}^{N} y_i \log_2(\hat{y}_i) \tag{21}$$

Loss is calculated for both training and validation datasets. Training loss is used to optimise the weights. Validation loss is used to monitor the training progress. Because validation loss is not used for optimisation, it indicates how the model generalizes to unseen data samples. Training and validation losses are often plotted as learning curves where the loss is on the y-axis and the training time is on the x-axis.

2.6.4 Backwards Pass

The next step in the training is a backwards pass. As the name implies, the backwards pass starts from the output and goes through the network towards the input. The input is training samples in a random order, which makes it stochastic optimisation. The goal of the backwards pass is to calculate gradients of the loss function with respect to each weight of a neural network [5]. Resulting gradients are then used to update the weights. Gradients are calculated using a back-propagation algorithm that uses a chain rule.

The chain rule is a derivative rule that makes it possible to find a derivative of a composite function. It is an essential part of the back-propagation algorithm. The chain rule is expressed in Equation (22) [68].

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dz} \tag{22}$$

As discussed in section 2.5.1, NN can be expressed as a function composition. Therefore, the chain rule can be applied to calculate gradients of loss function L. For example, gradient g for weights W_1 for NN shown in Equations (14) to (19) is:

$$g(W_1) = \frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial f_3} \cdot \frac{\partial f_3}{\partial f_2} \cdot \frac{\partial f_2}{\partial f_1} \cdot \frac{\partial f_1}{\partial W_1} \tag{23}$$

2.6.5 Updating Weights

Weights in the network are updated according to the gradient from the backwards pass. The objective of weight update is to locate a minimum of the loss function. The minimum is found using optimizers. The most basic optimizer is a stochastic gradient descent (SGD) from which more complex optimizers like adaptive gradient (AdaGrad), and root mean squared propagation (RMSprop), are derived. Currently, the optimiser regarded the most efficient is adaptive moment estimation (Adam) [5]. This section discusses SGD and Adam optimizers.

SGD is probably the most used optimizer in ML [5]. In SGD weights are updated in the direction of the negative gradient by the amount of the learning rate. SGD is presented in Equation (24), where w_t is the updated weight, w_{t-1} is the previous weight, η is the learning rate and L is the loss function [69].

$$w_t = w_{t-1} - \eta \nabla_w L \tag{24}$$

A key hyperparameter in SGD optimisation is a learning rate η. The learning rate determines how much the gradient changes the weights. The value of the learning rate has a great impact on model learning. Too low learning rate makes learning slow. Too large learning rate fails to converge to the minimum or even diverge from it. The ideal learning rate is small enough that the optimal value is reached and large enough that the optimal value is reached in a reasonable time. The effect of the learning rate value is presented in Figure 27. The downside of SGD is that it uses a fixed learning rate for every weight in the network.

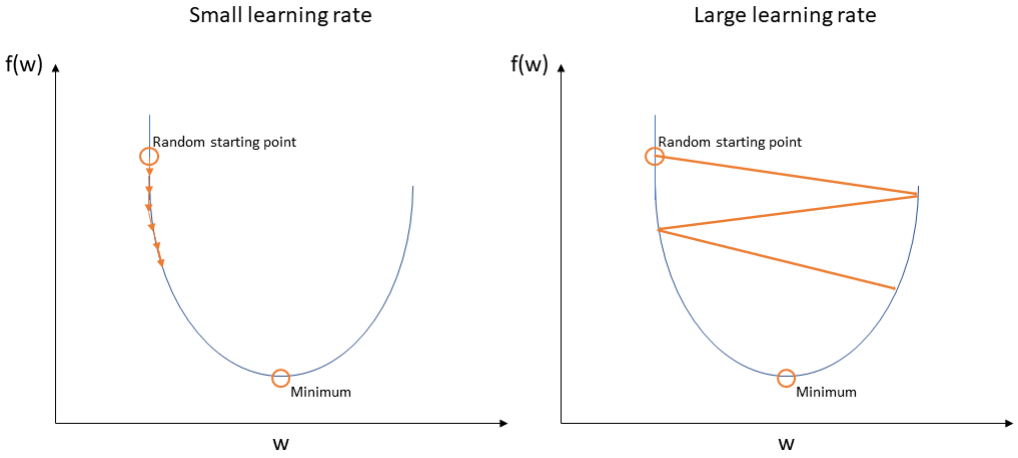

Figure 27: Effect of too small and too high learning rate in SGD. Too small learning rate takes a long time to converge. Too large learning rate does not converge at all and ends up oscillating around the minimum.

Adam optimizer is an adaptive learning rate algorithm. Whereas SGD has a fixed learning rate, Adam changes the learning rate depending on the value of the gradient. Adam uses a momentum m_t and scaling factor v_t to adjust the learning rate. Equations (25)-(29) demonstrate how the steps are updated using Adam optimizer [70].

Adam updates the momentum m_t and the scaling factor v_t based on the most recent gradient g_t. Momentum increases the learning rate if the gradient points consistently in a certain direction and decreases if the gradient's directions differ. Scaling factor scales gradient steps between mini-batches to achieve consistent gradient decent. Momentum is calculated according to Equation (25) and scaling factor according to Equation (26). Both equations include a hyperparameter β_1 or β_2 that express the importance between the current and the previous gradient.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{25}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{26}$$

Bias corrections $\hat{m}_t$ and $\hat{v}_t$ prevent the weight updates from becoming too large [70]. This can happen with small β_1 and β_2. Bias corrections for momentum are shown in Equation (27) and for scaling factor in Equation (28). Once these values are derived, the weights are updated according to Equation (29), where η is the to-be-adapted learning rate and ϵ is a small value. Hyperparameter ϵ is used to avoid divisions by zero.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{27}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{28}$$

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{29}$$

Once the weights have been updated the training process starts all over again from the feedforward pass. Each round the training goes through the training dataset is called an epoch. The number of epochs is a common way to express the training length. The training continues until it has gone through a certain number of epochs or the training is stopped with an early stopping.

The early stopping point can be determined with validation loss. If at a certain point the validation loss does not improve or starts to increase, training should be stopped. After this point, the model will overfit to the training data and its ability to generalize to unseen data worsens. Once the training has been completed, the resulting model needs to be evaluated with the test dataset.

2.7 Evaluating the Model

Model evaluation is a process of determining how well the machine learning model performs in the task it is designed to do. This chapter examines model evaluation only in supervised and classification tasks. Unsupervised models and regression problems demand different methods for model evaluation. The model is evaluated using test data that has not been used for training the model. This is done in order to gain an unbiased assessment of the model performance. Ideally, this would expose if the model has overfitted to the training data.

One of the most informative ways to evaluate test performance in multi-label classification is the confusion matrix [71, 72]. A confusion matrix summarises the results of model output in a matrix where the correct and incorrect classifications are divided into entries in the matrix. The name confusion refers to the fact that from it we can see where the model is confused when making predictions. The confusion matrix gives great insights into what kind of errors the model makes. Figure 28 represents the confusion matrix for a binary classification problem.

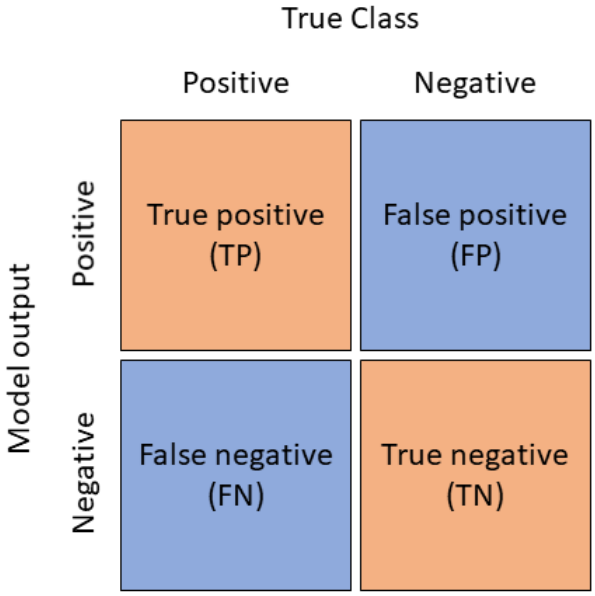

Figure 28: Confusion matrix for binary classification.

For binary classification one class is named positive and the other negative. On the side of the matrix are the predicted labels and on the top are the expected true labels. This makes the correct predictions in the main diagonal as seen in Figure 28. In binary classification, these are referred to as true positive and true negative. Other entries are incorrect predictions where the model has confused a class for another. In a binary classification, there are two kinds of errors that the model can make: it can predict a negative label as true, or a true label as negative. These are referred to as false positives and false negatives. In condition monitoring systems, false positives are false alarms and false negatives are missed faults.

Many performance metrics can be derived from the confusion matrix. This chapter explains four of the most common ones: accuracy, precision, recall, and F1-score [72]. Usually, the most common and understandable metric is accuracy. It is the ratio between true predictions and the total number of instances. Accuracy can be calculated:

$$\text{Accuracy} = \frac{\text{True positive} + \text{True negative}}{\text{Total}} \tag{30}$$

However, accuracy does not distinguish between false positives and false negatives. Taking these factors into account gives a comprehensive picture of the performance.

Also, accuracy might not give a realistic representation of the performance if the dataset is imbalanced [71]. This might be the case if data for one label is easier to produce in the data acquisition phase. In this case, high accuracy could be reached easily if most of the data is negative and the model only gives true negative predictions. This is why measures like precision and recall are commonly used in addition to accuracy.

The precision is how many of the positive predictions were truly positive [73]. Precision can be calculated with Equation 31.

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}} \tag{31}$$

The recall is how many of all of the positive samples were correctly predicted [73]. The recall is also sometimes called sensitivity. It can be calculated with Equation 32.

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \tag{32}$$

Precision and recall give different perspectives on the model's performance. However, these measures should not be used alone, as there is a possibility of getting excellent performance on one measure and poor on the other. Using F1-score, precision and recall can be combined into a single measure [74]. F1-score is a harmonic mean of precision and recall and can be calculated as follows:

$$\text{F}_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{33}$$

To get a perfect F1-score, the model has to have precision and recall of 1. However, measures do not have to be treated as equal. Depending on the application, one measure might be deemed more important than the other. For example, if false positives are more unwanted then false negatives precision should be preferred over recall. Weighting one measure over another can be done with F_β-score. F_β-score uses factor β that is chosen to reflect how many times recall is deemed more important than precision [75]. F_β-score can be calculated Equation 34.

$$\text{F}_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}} \tag{34}$$

All the above metrics assumed a binary classification task. In the case of multi-class classification, the model evaluation becomes more complicated as there are no universal positive or negative predictions. Multiple classes are often needed in machine health diagnosis as the model is usually used to diagnose multiple fault states. Multi-class classification can be reduced to a binary case by comparing one class to all of the other cases [73]. Therefore, precision, recall, and F1-score can be calculated individually for each class. Figure 29 presents an example of 3-class classification confusion matrix. In Figure 29 colours present the positives and negatives for class A. For the class A green table entry is the true positive, red is the true negative, blue is the false positive, and orange is the false negative. After calculating the F1-scores

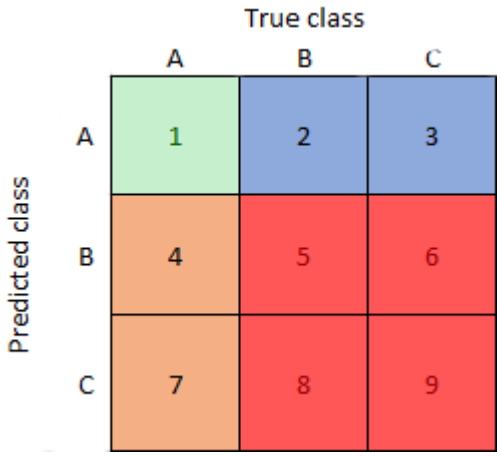

Figure 29: Multi-class confusion matrix. For the class A green table entry is the true positive, red is the true negative, blue is the false positive, and orange is the false negative.

for all the classes, they can be combined in various ways into a single metric. These are macro and micro F1.

Macro F1 is the unweighted mean of each classes individual F1 score [74]. In macro F1 all of the classes are treated equally and therefore it is a good measure for imbalanced datasets where all of the classes are as important. Macro F1 can be calculated with the following formula:

$$\text{Macro F1} = \frac{F_1^A + F_1^B + F_1^C}{3} \tag{35}$$

Micro F1 differentiates from the two aforementioned measures by not considering each class individually [74]. Micro F1-score gives a good and simple measure for model performance with balanced datasets. When calculating the micro F1-score total true positives, total false positives and total false negatives are considered. In the micro F1-score, total true positives are in the main diagonal of the confusion matrix, total false positives are the sum of all false predictions on each row, and false negatives are the sum of false predictions on each column.

3 Materials and Methods

This section of the thesis describes the methods used for the experimental part of the study. The first section introduces the experiment setup. The second section describes the test runs used to measure the torque data from VFD. The third section describes how the measurement data was preprocessed into a dataset in preparation for the model training. The fourth section introduces the trained CNN model architecture. The final fifth section shows how the model was trained and evaluated.

3.1 Experiment Arrangement

The experiments were conducted in a research centre of the Sulzer Pump factory located in Kotka. The aim of the experiment was to generate a large high-quality labelled dataset for supervised learning. The dataset for the DL application was acquired by measuring raw torque data from VFD while operating a centrifugal pump in known conditions. The operated media in the test loop was water. During the measurements, the temperature of the pumped water was around room temperature.

Figure 30 shows a simplified flow sheet of the experiment arrangement. Measurements were taken from VFD driven centrifugal pump in a closed-loop system. The main components of the setup were the centrifugal pump, VFD, control valves, water tank, and data logger. VFD was used to vary pump speed. Control valves were used to vary the head, flow rate, and NPSH.

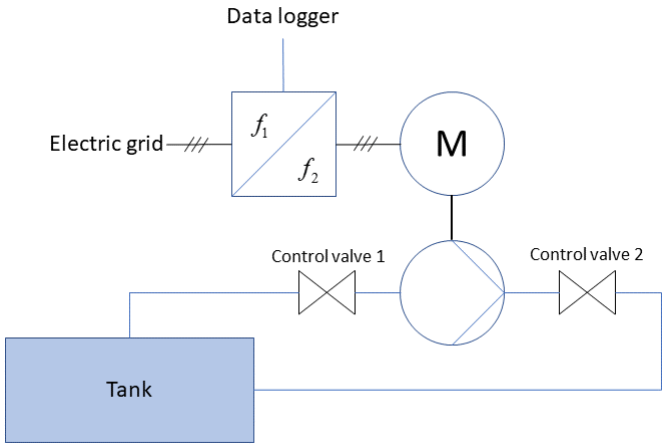

Figure 30: Simplified representation of the experiment arrangement.

The centrifugal pump used in experiments was Sulzer Ahlstar end suction process pump with a closed impeller. The pump was connected to the motor using a stiff coupler, which does not dampen the torque changes as a flexible one would. VFD used in the experiments was ABB wall-mounted single drive ACS880-01. Data was stored using a National Instruments Compact DAQ-3174 system with a NI-9231 input module.

3.2 Test Runs

With the experiment arrangement, raw torque data from VFD was measured with NPSH and QH test runs. NPSH test runs were used to gather data from the pump in normal, cavitation and abrasive metal-metal contact conditions. QH test runs were used to gain additional data on abrasive metal-metal contact. Between the measurements pump speed, head, and flow rate were changed to give variety to the dataset. These measurements were recorded with a sampling frequency of 8kHz and each measurement point was recorded for 5 seconds. This results in 40 000 time steps per measurement point.

In the NPSH test run, pump speed and flow rate were kept constant. NPSH was lowered gradually and the measurements were taken from different NPSH values. The NPSH value on each run was lowered until the pump head start to rapidly decrease. Because of the rapid decrease, there is limited time to take measurements from fully developed cavitation. This is why the number of measurement points for each NPSH run varies.

The NPSH-head curve was recorded in order to label the data correctly at a later stage. Figure 31 presents the planned NPSH test run locations on the pump curve at different rotational speeds. From NPSH test drives only two had contact situations which both had 2000 rpm rotational speed. More info on NPSH test runs can be found in Appendix A.

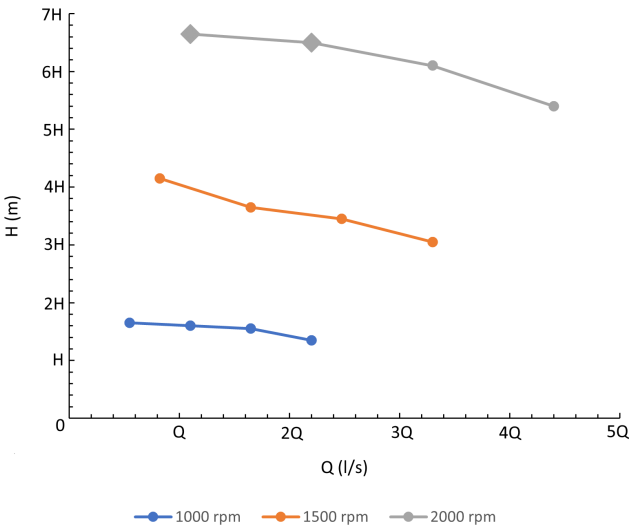

Figure 31: NPSH test run QH points. At each point, the NPSH test run was performed with multiple measurement points. From each point normal and cavitation data can be measured. Measurement points with the diamond marker are where contact occurs.

In addition to NPSH test runs, QH test runs were performed to add more variety to abrasive metal-metal contact data. These tests were done using the same setup as in NPSH tests with the only difference being a different wear ring. The wear ring in

QH test runs had a 30% smaller radial clearance. The tighter wear ring radial cap was to make contact easier to produce.

In QH test runs, the pump was run in a single operating point with specified head and flow rate parameters. Radial force and shaft deflection was calculated with these specific parameters in order to verify the contact. Figure 32 shows the planned measurement points of the QH test runs. In the Figure, measurement points with a diamond marker are where contact occurs. More info on QH test runs can be found in Appendix B.

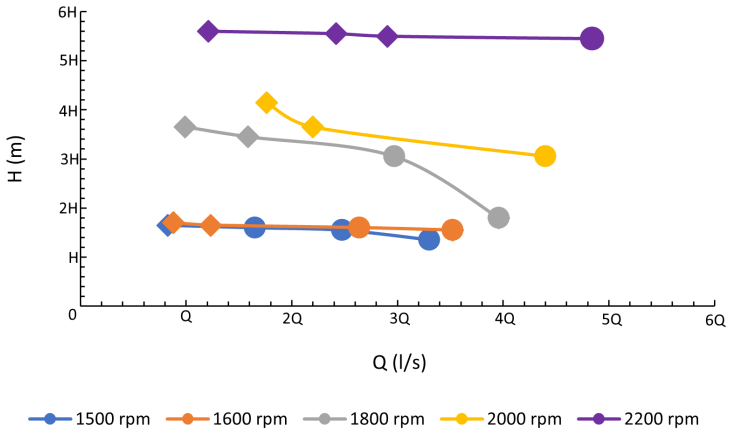

Figure 32: Contact test drive measurement points. Measurement points with diamond marker are where contact occurs.

3.3 Measurement Data Preprocessing

Measured raw data was preprocessed into a clean labelled dataset. Preprocessing includes cleaning, labelling, window splitting, augmenting and splitting measured data into multiple datasets.

Data was labelled into classes 0, 1, and 2 that correspond to the classes normal, cavitation, and abrasive metal contact. Unclear and edge case samples were left out. Cavitation and normal classes were determined by the recorded NPSH-H curves. Cavitation was determined as a 3% decrease in the pump head. Figure 33 demonstrates how the labelling was conducted for cavitation. Abrasive metal contact was determined mathematically by calculating the axle bend and comparing it to the sealing ring radial gap. Again, unclear samples were left out.

The labelled measurement data was divided into three datasets: training validation and testing sets. Of the measurement data 50% was allocated to training, 25% to validation and 25% to testing datasets. For each dataset, the label ratio was kept the same. Additionally, samples were shuffled in order to get samples of different operating parameters for each dataset. Training and validation datasets were then augmented to artificially add more samples. The testing dataset was not augmented to give a realistic representation of the model performance.

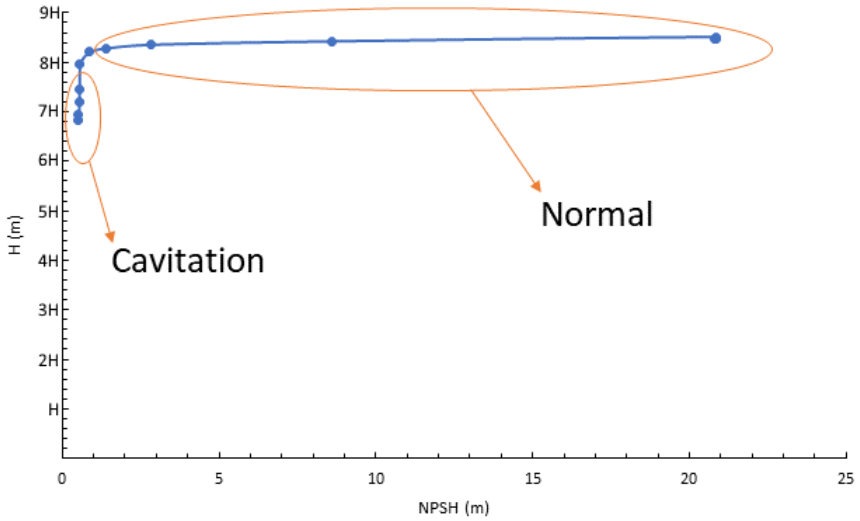

Figure 33: Method used for measurement labelling.

Training and validation were augmented using window slicing and flipping. In window slicing, samples were divided into 2048 timesteps with an overlap of 2000. This results in 59 424 samples in testing and 29 920 samples in the validation dataset. Datasets were kept separate in the window slicing, meaning that the datasets did not share slices from the same original samples. Additionally, a flipping augmentation method was also implemented. However, as the effect of flipping was not known model was trained with and without flipping. Flipping effectively doubles the number of samples in the dataset. Figure 34 summarizes the dataset sizes and what augmentation methods were used for each dataset.

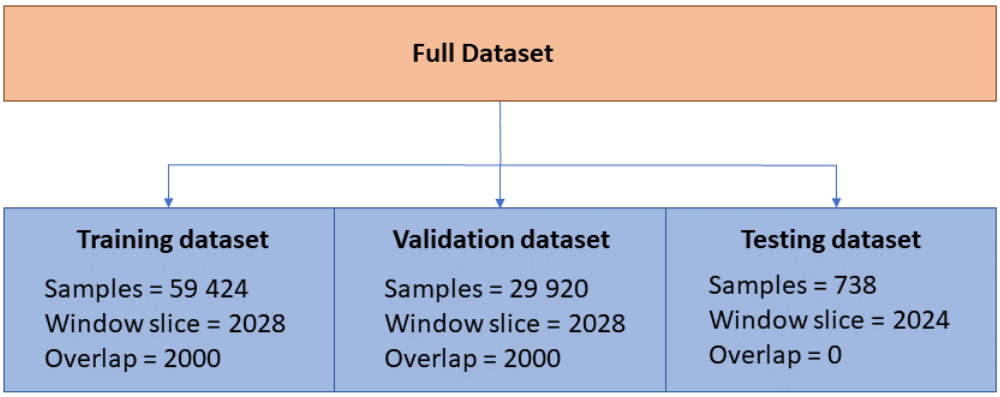

Figure 34: Dataset split of the measurement data.

3.4 Model Architecture

CNN architecture was chosen for the DL model. CNN model makes an excellent choice as they are designed to find features from time series data. The model

architecture is a slightly modified version of the WDCNN which was introduced in section 2.5.3. This model was chosen as the basis for this thesis application because of its success in a similar problem domain. In their study, there were ten condition classes, which is why the output layer was modified to output a vector with three elements [63]. A graphical depiction of the CNN model is presented in Figure 35.

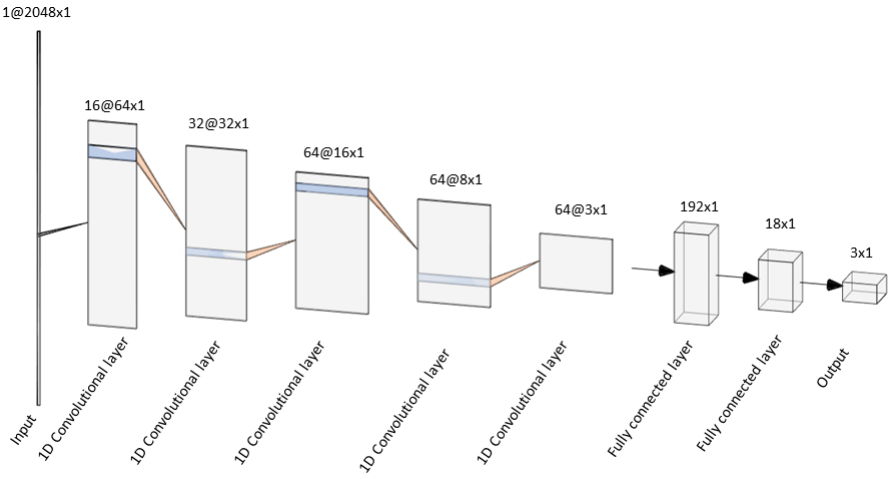

Figure 35: CNN architecture.

As can be seen from Figure 35, the CNN consists of five convolutional layers and two fully connected feedforward layers. The model takes a 1d vector of length 2048 as an input and outputs a 1D vector of length 3. The architecture has a total of 37 291 trainable parameters. Table 1 presents the layers of the CNN model architecture in detail.

Table 1: Details of the CNN model used in the experiment.

Layer	Filter Size	Channels in	Filters	Stride	Activation	Parameters
Conv1	64x1	1	16	16	ReLu	1 056
Conv2	3x1	16	32	1	ReLu	1 600
Conv3	3x1	32	64	1	ReLu	6 272
Conv4	3x1	64	64	1	ReLu	12 416
Conv5	3x1	64	64	1	ReLu	12 416
FCL	1x1	192	18	-	ReLu	3 474
FCL	1x1	18	3	-	Softmax	57

CNN layers at the beginning of the architecture extract features from the input. Each CNN layer has convolution, batch normalization, max pooling, and ReLu activation stages. The size of the first convolutional filter is 64×1, and the rest layers have a filter size of 3×1. After each convolutional layer, batch normalization is used to enhance the performance.

Fully connected layers do the classification based on the preceding convolutional layers' feature maps. A softmax activation at the end gives the output vector elements as a probability distribution. Output vectors' largest value is the predicted class.

3.5 Training and Evaluating the Deep Learning Model

The CNN model was trained using the training and validation dataset. Training used Adam optimiser. A validation dataset was used to monitor the training process. Training used regulation methods to mitigate overfitting and grid search for hyperparameter tuning.

To avoid overfitting, an early stopping regulation method was implemented. Early stopping stops the training if the performance does not improve after a certain amount of epochs. In this case, training was stopped if there was no improvement in ten epochs. Model performance was monitored with the validation dataset cross-entropy loss. Other regulation methods were considered but were not implemented in the final training. These included weight constraining and dropout.

The model has a large number of hyperparameters outside of the trainable parameters. The value of these parameters can significantly affect the learning outcome. To keep the model training time reasonable, only two hyperparameters were tuned. Tuned hyperparameters were learning rate and batch size.

In order to obtain the optimal combination of hyperparameters the model was trained multiple times using a method called grid search. In grid search, the model is trained multiple times using a different combination of hyperparameters. Furthermore, a model was trained ten times on the same hyperparameter combination. This is important because the random nature of the training can yield different results with the same parameters. For the grid search values 0.05, 0,005 and 0.0005 were used for learning rate and 8, 16, 32, and 128 for batch sizes. This results in 12 possible hyperparameter combinations. The combination of hyperparameters resulting in the best performance will be chosen as the final model.

The final model is evaluated using the test dataset. From these results, the model is evaluated using a confusion matrix. The confusion matrix can be used to calculate metrics like accuracy, precision, recall, and f1-score.

4 Results

This chapter presents the results of the experiments. The first section presents the data set obtained from the measurements. The second section presents the model training and grid search results. The third section evaluates the best model's detailed performance on the test dataset.

4.1 Measurement Results

A total of 219 measurement points were recorded with a sampling frequency of 8kHz for 5 seconds. These measurement points on the NPSH test drives are presented in Figures 36 to 38.

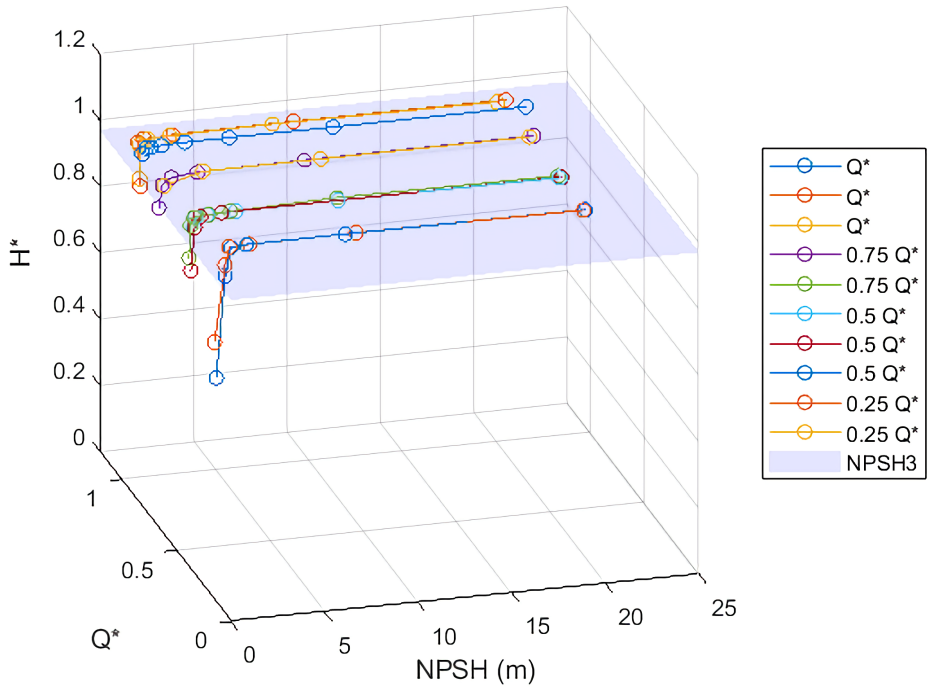

Figure 36: NPSH test drive measurement points 1000 rpm. The blue plane represents the NPSH3 value, after which the sample is labelled as cavitation. Axis values H* and Q* are given in relation to the best efficiency point value.

After discarding the unclear and edge case measurement points, a total of 161 measurement points were included in the training phase. The number of measurement points and label distribution is presented in Table 2.

As can be seen from the Table 2 the dataset is unbalanced. Label normal amounts to 50%, cavitation 36.8%, and contact 13.7% of the full dataset. This unbalance is compensated in the model training by assigning weight values for the loss function. Additionally, the accuracy of the model is also weighted using these values.

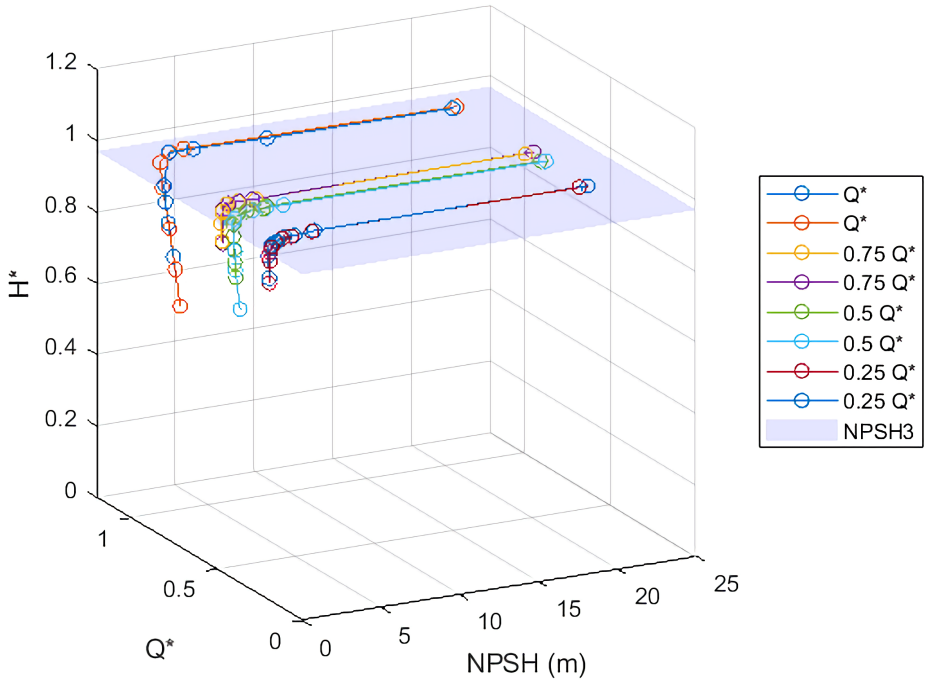

Figure 37: NPSH test drive measurement points 1500 rpm. The blue plane represents the NPSH3 value, after which the sample is labelled as cavitation. Axis values H* and Q* are given in relation to the best efficiency point value.

Table 2: Label distribution

Label	**Measurement points (no.)**	**Distribution (%)**
Normal	80	49.5
Cavitation	59	36.8
Contact	22	13.7

These raw data samples were divided into train, validation, and test datasets and augmented with window division. Table 3 presents the number of augmented samples in each dataset.

Examples of the raw torque signal data are shown in Figures 39 and 40. Figure 39 shows the time series samples with labels normal and cavitation. These samples were measured when the pump was driven with a rotational speed of 1000 rpm, head of 27 m and flow rate of 22 l/s. NPSH for a normal sample was 20.83 m and 0.86 m for cavitation. From this Figure 39, it can be seen that the cavitation time series data has slightly higher values and does not dip as low as the normal time series.

Figure 40 compares torque time series signals of normal and abrasive metal-metal contact. In this sample, the pump was operated with a rotation speed of 2200 rpm, head of 112 m, and flow rate of 12 l/s. In this sample, contact was severe with a

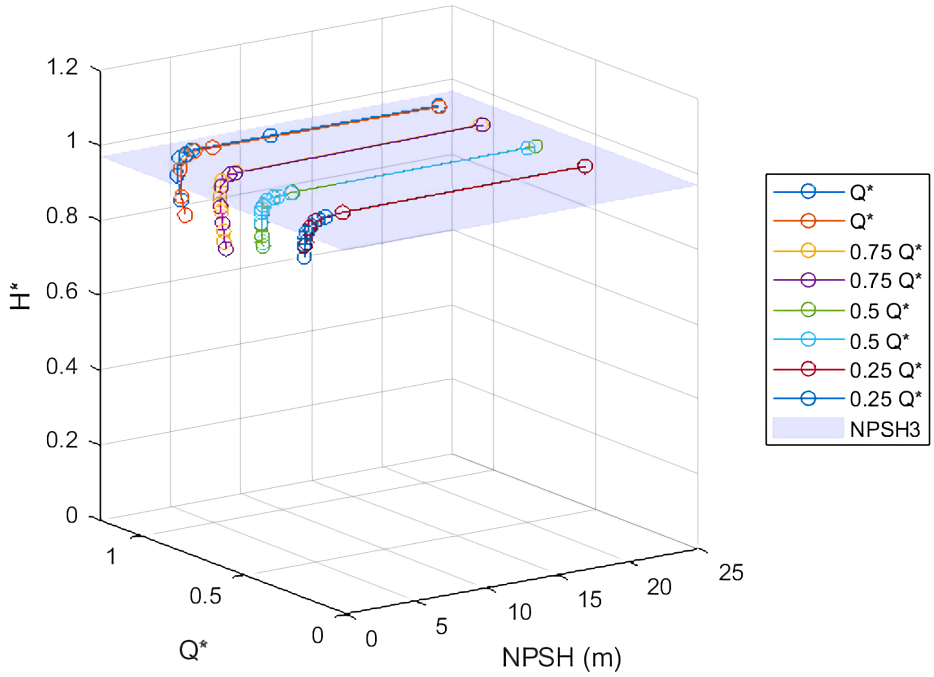

Figure 38: NPSH test drive measurement points 2000 rpm. The blue plane represents the NPSH3 value, after which the sample is labelled as cavitation. Axis values H* and Q* are given in relation to the best efficiency point value.

Table 3: Data augmentation results

	Dataset		
	Train	**Validation**	**Test**
Samples	59 424	29 920	738
Normal	29 504	14 960	360
Cavitation	21 692	11 220	270
Contact	8 228	3 740	108

calculated average shaft deflection of 0.89 mm while impeller and wear ring clearance was 0.33 mm. Normal condition torque signal had the same rotational speed as the contact sample with a pump head of 109 m and flow rate of 48 l/s. In the contact torque signal, we can see more fluctuation compared to the normal torque signal with the same rotational speed. A clear difference between the samples is promising for the DL model.

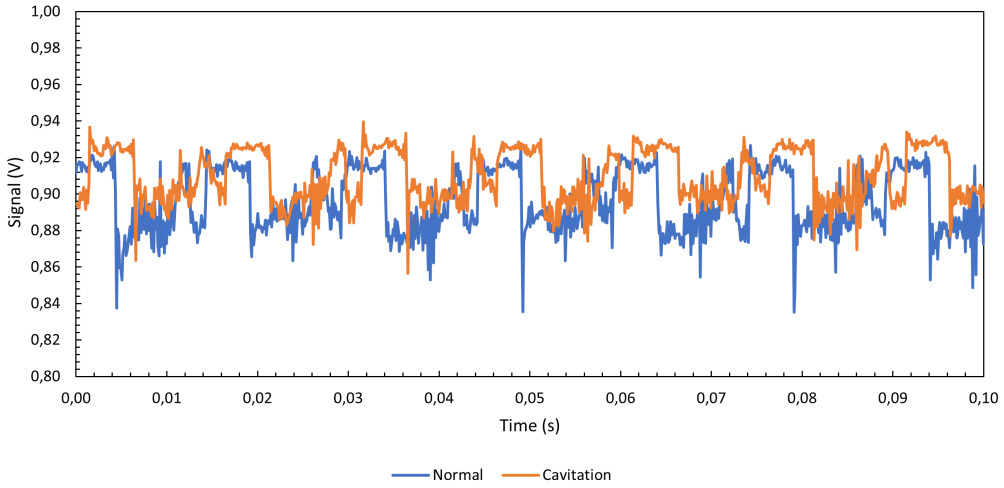

Figure 39: Example of normal vs cavitation torque signal.

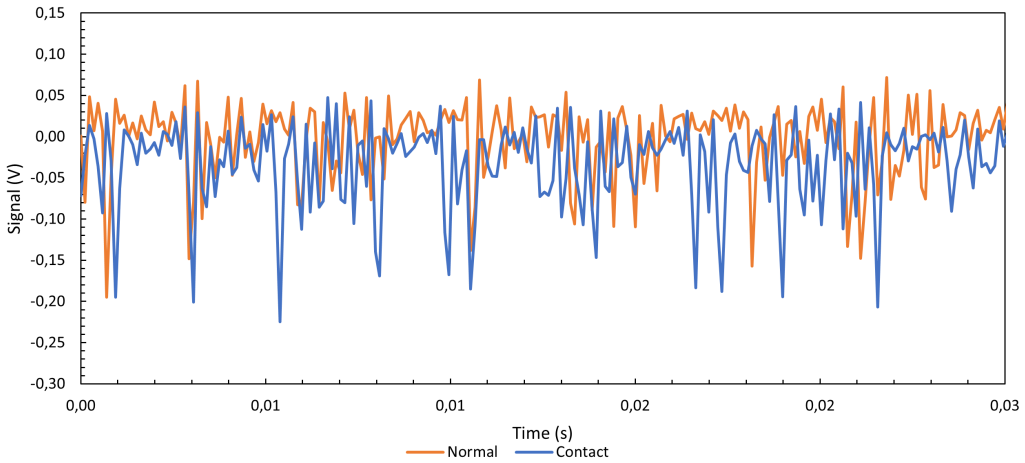

Figure 40: Example of normal vs. contact torque signal.

4.2 Model Training Results

This section presents the results of the model training. These include the hyperparameter tuning with a grid search method and final model selection.

Learning rate and batch size hyperparameters were tuned using a grid search. For each hyperparameter combination, ten models were trained. Table 4 presents the mean weighted accuracy of these models. Across all the hyperparameters model achieved weighted accuracy ranges from 55% to 65%.

Table 5 shows the best result for each hyperparameter combination. Cells of the tables are coloured as a heat map where green cells have the best results, red the worst, and yellow the average.

The best performing model was achieved with a learning rate of 1.00E-05 and batch size of 8 as can be seen from Table 5. From both the grid search tables can be seen, that a small learning rate and batch size resulted in better learning. The

Table 4: Grid search for the average weighted accuracy for each hyperparameter combination.

avg		Learning rate		
		1.00E-05	1.00E-04	1.00E-03
Batch size	8	0.652	0.565	0.572
	16	0.653	0.609	0.558
	32	0.637	0.612	0.603
	128	0.606	0.598	0.567

Table 5: Grid search for the best weighted accuracy for each hyperparameter combination.

max		Learning rate		
		1.00E-05	1.00E-04	1.00E-03
Batch size	8	0.784	0.655	0.656
	16	0.708	0.737	0.656
	32	0.738	0.714	0.651
	128	0.721	0.728	0.694

learning rate seems to be a more crucial hyperparameter than the batch size.

4.3 Model Results

The best performing model from the grid search was evaluated further. As stated in the previous section, the best model achieved weighted accuracy of 78.4%. This section presents a more detailed view of the model performance on the test dataset.

Figure 41 presents the learning curves of the chosen model. The Figure 41 shows that the model trained for 32 epochs before early stopping. At the end of the training, cross-entropy loss for the training dataset was 0.85 and 0.89 for the validation dataset. In both cases, the loss did not decrease much.

Table 6 shows the model results with a confusion matrix. The confusion matrix contains the model output on the test dataset and demonstrates what kind of errors the model makes on unseen data samples.

Precision and recall performance metrics can be calculated from the confusion matrix. Table 7 shows these metrics for each class.

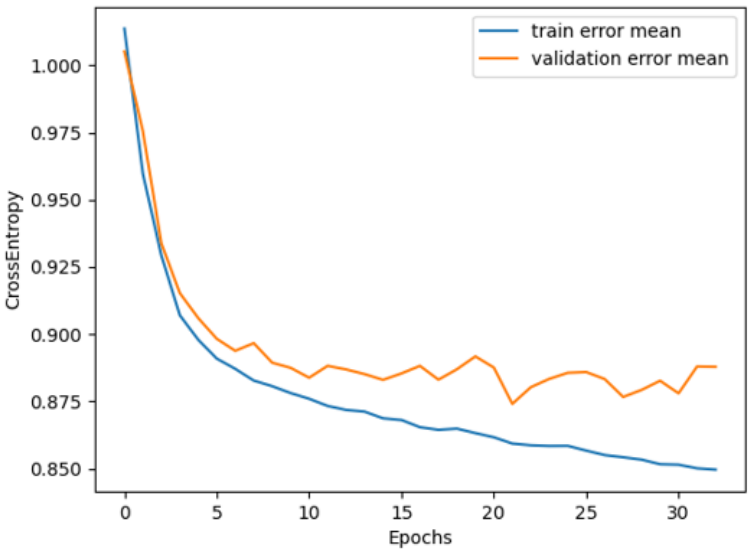

Figure 41: Training and validation learning curves of the best model.

Table 6: Confusion matrix.

	True class		
Model output	Normal	Cavitation	Contact
Normal	189	135	36
Cavitation	47	223	0
Contact	0	0	108

Table 7: Performance metrics.

	Precision (%)	Recall (%)
Normal	52.500	80.085
Cavitation	82.593	62.291
Contact	100.00	75.000

5 Discussion

This section analyses the previously presented results. This section is divided into four parts. The first part is the overview of the results and how they could be improved. The second part discusses the possible scientific impact these results have. The third part is about the possible practical impact these results can have. The final part introduces further research topics that could be conducted on the basis of this thesis.

5.1 Overview

The goal of the experiments was to train a three-class classification CNN model. The model was acquired by training it with a relatively large dataset of raw VFD torque data and choosing the best-performing model by grid search. The final CNN model achieved a weighted classification accuracy of 78.4%. This result was explored further with a confusion matrix and precision and recall performance metrics. Overall, the results are promising, but there is room for improvement.

The most encouraging result is the model's performance on contact classification. As can be seen from performance metrics in Table 7, the model achieved 100% precision on contact samples. Also, the model was able to distinguish contact from cavitation samples. As can be seen from the confusion matrix in Table 6, the model did not mistake any cavitation samples as a contact. Therefore, there is the possibility to extract more information from the raw data than just higher fluctuation in the torque signal. This makes it possible to diagnose more than one fault state from raw VFD data.

More research and testing need to be done to improve some aspects of the model's performance. Models' cross-entropy loss leaves much to be desired. Also, model classification performance between normal and cavitation conditions needs improvement.

Learning curves in Figure 41 show that models cross entropy loss does not decrease much before the model validation loss starts to increase. The model starts to overfit quickly to the training data and its ability to generalize worsens.

Another observation that can be made from the learning curves is that cross-entropy loss does not decrease much. The validation cross-entropy loss for the model is as high as 0.89. However, the model performs well regardless of the high cross-entropy loss. This disparity could be explained by the uncertainty of the model's output probabilities. For example, it might be the case that when the model output is wrong, it is very certain of the wrong classification. In addition, the model may not give confident outputs for the correct classification. This way even a correct output could contribute to the high loss value.

Lower cross-entropy loss could be achieved by implementing other regulation techniques in addition to early stopping. Possible regularization methods include weight constraining and dropout. Weight constraining gives limits for the size of the weight because large weights are usually a sign of overfitting. Dropout on the other hand deactivates neurons at random which would force the model to be more adaptable. However, these methods introduce a large number of hyperparameters that would need to be tuned.

Models' largest mistakes were with classifying cavitation and contact samples as normal as can be seen from the confusion matrix in Table 28. This results in normal precision of 52.5%. The second worse performance metric was cavitation recall of 62.3%, which is due to a large number of cavitation samples being misclassified.

It seems like the loss function optimisation gets stuck in local minima. The model learns to classify contact conditions well but does not learn to distinguish cavitation from normal samples. This theory is enforced by the average weighted accuracies of

the grid search, shown in Table 4. Average weighted accuracies are between 66%, which means that the model classifies contact conditions well and other conditions the model gets right half of the time.

The model´s poor performance in classifying cavitation samples from normal ones may indicate that the difference between normal and cavitation samples is not clear. It is possible that the labelling method introduced in Figure 33 does not divide the normal and contact samples harshly enough. As can be seen from Figure 7, cavitation can start before the pump head is affected. Therefore, it might be possible that some samples labelled as normal already show signs of cavitation which confuses the model learning.

5.2 Scientific Impact

This section explores what kind of influence the findings can have on science. There are two main aspects the results offer. The first one is the fault diagnostics on VFD raw data. The second one is the used DL model.

This thesis is one of few studies conducted on the use of raw VFD data for pump condition monitoring. As discussed in section 2.2, most of the previous studies have focused only on RMS levels. By monitoring only the RMS value, the method could only diagnose one condition. This method introduced in this thesis goes further by determining the pump condition from patterns of the VFD data. As the results show, this way it is possible to diagnose a larger variety of pump conditions.

Another scientific impact this thesis offers is the use DL model. Results show how a CNN model introduced in Table 1 performs on raw VFD torque data. Even though the model results need improvement, they can offer groundwork for further studies.

5.3 Practical Impact

In addition to scientific impact, this thesis has a broad practical impact. This section explores the practical impact that the results have now and the possible future impact.

In the current state, the model can offer a smart and sensorless condition monitoring method for a centrifugal pump and add value to VFD. As previously discussed, the results show that the model can detect accurately pump abrasive metal-metal contact conditions. By implementing this model, the automation level of pump maintenance can be enhanced with little cost. The model could be used to monitor pump operation and alert users if a contact situation occurs. Taken further, pump operation could be changed automatically to avoid contact.

There are a couple of ways to bring the model to use. One is having the model installed as a part of the VFD software. This is a simple solution and would require collaboration with the VFD manufacturer. Another method is to develop a data-gathering device that could be installed into VFD. An advantage of this approach is that it could work with different manufacturers' products. The device itself could run the model locally or remotely. As the model is quite small it can operate with

relatively light computing power. Alternatively, the device could forward the data to a server which runs the model remotely.

If developed further, the results can have a far greater practical impact. The results demonstrated the possibility of diagnosing multiple pump failures from the VFD data. If the model's reliability can be increased enough this method could offer a cost-effective condition monitoring system. Developing the model's application further, it could be used as a part of a full predictive maintenance system.

5.4 Future Research

The results offer multiple further research possibilities. Future research can be conducted on the VFD dataset or the DL model.

VFD raw data offers multiple research possibilities. This thesis did not analyse the VFD data in depth. It would be valuable to investigate what exactly the DL model finds in the data. Besides further analysis, the measured dataset could also be used for different applications, Dl or otherwise.

Another research topic is the effect of coupler stiffness on the VFD data. In this thesis measurements, a stiff coupler was used. A flexible coupler could dampen the torque data so that fault diagnostics become harder or impossible.

The DL model is another source of further research. As previously stated model's performance needs to be improved. This offers a further research possibility to use different or more advanced approaches. These methods could include model regulation mentioned in section 5.1, or changing the DNN architecture or type.

The model could also be researched further on how it performs when data is produced from different-sized radial centrifugal pumps. Data could also be derived from other centrifugal pumps like vertical or axial pumps. In this case, the model could be used for transfer learning to make it generalize to other data sources.

An obvious, research subject would be adding more pump fault conditions for fault diagnostics. An interesting research question would be is the model able to differentiate other mechanical faults from the contact. This also could give a clearer picture of what VFD-based condition monitoring is capable of.

6 Conclusion

The aim of this thesis was to develop a supervised ML method for condition monitoring and fault diagnosis of centrifugal pump based on raw VFD data. The use of VFD as a source of data can offer a cost-effective and sensorless condition monitoring method.

This thesis is one of the few studies conducted on the use of VFD data for pump fault diagnosis. Previous studies on the subject rely on measuring RMS values and comparing them to the known normal RMS values. The approach explored in this thesis enables to extract and diagnose more fault conditions than earlier methods.

The studied fault states were limited to two centrifugal pump fault states: cavitation and abrasive metal-metal contact. Cavitation is a phenomenon where vapour

bubbles form and collapse in the liquid due to reduced and subsequently increased pressure. Abrasive metal-metal contact is caused by uneven pressure in the pump volute case. These two fault conditions were chosen as they are a temporary phenomenon and therefore hard to spot with traditional condition monitoring methods.

The goal of the measurements was to obtain a large and quality dataset of VFD torque data under different pump conditions. Measurements were done with NSPH and QH test drives. In the test setup, fault conditions were produced by the use of control valves. The centrifugal pump was operated on different rotational speeds, heads, and flow rates to gain variety in the data.

The resulting data set had 161 measurement points. Dataset was not balanced as 50% samples had the label normal, 36.8% cavitation and 13.7% contact. The dataset unbalance was a result of samples of one class being easier to produce than others.

For the training and validation, the full dataset was divided into three datasets: train, validation, and test. Training and validation datasets were augmented with a window slicing method, in order to artificially inflate the dataset sizes. After augmentation, the dataset sizes were 59 424 samples for training, 29 920 for validation, and 738 for the testing dataset.

The measured dataset was used to train a DNN. The problem DNN was used to solve was a classification task with three classes: normal, cavitation, and abrasive metal-metal contact. Because data from VFD was in a form of time series a CNN type of DL model was chosen.

The trained CNN model had five convolutional and two fully connected layers. The model took time series data as an input and outputs a probability distribution as a vector with three values. Each of the values in the output vector corresponds to the probability that the input belongs to that class.

In the model training, learning rate and batch size hyperparameters were tuned using grid search. Trainable parameters were optimised using cross-entropy loss and Adam optimiser. During training, the cross-entropy loss did not decrease much. The best-performing models' train loss was 0.85 and the validation loss was 0.89. Regardless of the high error, the best model achieved weighted accuracy of 78.4%.

Further analysis of the results shows that the model performed exceptionally well on abrasive metal-metal contact data. The model achieved a precision of 100% on contact samples of the testing dataset. Additionally, the model was able to distinguish contact from cavitation samples, showing that fault diagnosis is possible. However, the model's performance in classifying normal and cavitation samples needs improvement. Precision on normal samples was 52.5% and recall of cavitation samples was 62.3%.

Overall, the results of this thesis are promising. With improvement, this kind of condition monitoring and fault detection system could be implemented into VFD software or as an embedded device.

References

[1] Liisa Heikinheimo, Jaakko Luovanto, and Vilhartti Hanhilahti. Valtioneuvosto myönsi käyttöluvan Olkiluoto 3 -ydinvoimalaitosyksikölle, 2019.

[2] Otso Huttunen, Harri Pietarinen, and Anni Lassila. Olkiluoto 3:n käynnistys myöhästyy – On mahdotonta arvioida ongelmien laajuutta, sanoo professori, 10 2022.

[3] Keith Mobley. *An Introduction to Predictive Maintenance*. Elsevier, 2 edition, 2002.

[4] Raymond S. Beebe. *Predictive maintenance of pumps using condition monitoring*. Elsevier Advanced Technology, 2004.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[6] Ishita Chakraborty, Daniel J Kluk, and Stress Engineering Services. Condition Based Monitoring of Pumps Using Machine Learning. *Offshore Technology Conference*, 2020.

[7] Samir Khan and Takehisa Yairi. A review on the application of deep learning in system health management, 7 2018.

[8] Vikas Singh, Purushottam Gangsar, Rajkumar Porwal, and A. Atulkar. Artificial intelligence application in fault diagnostics of rotating industrial machines: a state-of-the-art review, 2021.

[9] Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. Artificial intelligence for fault diagnosis of rotating machinery: A review, 8 2018.

[10] Monique F Kilkenny and Kerin M Robinson. Data quality: "Garbage in – garbage out". *The HIM journal.*, 47(3):103–105, 2018.

[11] BPMA. Variable Speed Driven Pumps Best Practice Guide. Technical report, British Pump Manufacturers Association, Birmingham, 2019.

[12] Tero Ahonen. *Monitoring of centrifugal pump operation by a frequency converter*. Lappeenranta University of Technology, 2011.

[13] IEEE Staff Corporate Author. Variable frequency drive as a source of condition monitoring data. In *2008 International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, 2008 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), pages 185–183, [Place of publication not identified], 2008. I E E E.

[14] Ilkka Kauranen, Mikko Mustakallio, and Virpi Palmgren. Tutkimusraportin kirjoittamisen opas opinnäytetyön tekijöille. Technical report, Teknillinen Korkeakoulu, Espoo, 2007.

[15] Allan Wirzenius. *Keskipakopumput*. Kustannusyhtymä, Tampere, 1969.

[16] Randall Whitesides. Basic Pump Parameters and the Affinity Laws. Technical report, PDH Online, Fairfax, 2012.

[17] Taufik Taufik, James Ruckdaschel, Dale Dolan, and Makbul Anwari. Modeling and analysis of a static VAR compensated mixed load system. In *Proceedings of the IASTED International Conference on Modelling and Simulation*, pages 99–105. Acta Press, 2010.

[18] Jorma. Järviö. *Kunnossapito*. Kunnossapidon julkaisusarja ; n:o 10. KP-Media, Helsinki, 4 edition, 2007.

[19] M Orskisz, M Wnek, K Kryzka, and P Joerg. *Variable Frequency Drive as a Source of Condition Monitoring Data*. International Symposium on Power Electronics, Electrical Drives, Automation and Motion, 2008.

[20] PSK Standards Association. PSK 5707 Vibration Measurement in Condition Monitoring. Diagnosis. Technical report, PSK Standard Association, 2019.

[21] ISO. Rotodynamic pumps-Hydraulic performance acceptance tests-Grades 1, 2 and 3, 2012.

[22] Bruno Schiavello and C Visser, Frank. Pump cavitation- Various NSPHR Criteria, NPSHA Margins, And Impeller Life Expectansy. *Proceedings Of The Twenty-Fifth International Pump Users Symposium*, 2009.

[23] B Zeqiri, PN Gelat, M Hodnett, and ND Lee. A novel sensor for monitoring acoustic cavitation. Part I: Concept, theory, and prototype development. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control a publication of the IEEE Ultrasonics, Ferroelectrics, and Frequency Control Society.*, 50(10):1342–1350, 2003.

[24] T. Ahonen, J. Tamminen, J. Ahola, and J. Kestilä. Novel method for detecting cavitation in centrifugal pump with frequency converter. *Insight: Non-Destructive Testing and Condition Monitoring*, 53(8), 8 2011.

[25] FLUID HANDLING INTERNATIONAL. Reducing cavitation risks with variable speed drives. *FLUID HANDLING INTERNATIONAL*, 2022.

[26] ABB. ACS880 Cavitation detection control program Supplement. Technical report, ABB, 2020.

[27] K. Prabith and I. R.Praveen Krishna. The numerical modeling of rotor–stator rubbing in rotating machinery: a comprehensive review, 7 2020.

[28] Larry. Bachus. *Know and understand centrifugal pumps*. Elsevier, Oxford, 2003.

[29] Johann Friedrich Gülich. *Centrifugal Pumps*. Springer, Villeneuve, 4 edition, 2020.

[30] Kari Santaoja. *Rasitusopin käsikirja*. Taras, Espoo, 4., uudistettu ja... edition, 2018.

[31] J Wiedenburg, Ernesto, Andre Ramme, E Mathesson, Annette Jouanne, and K Wallace, Alan. Modern online testing of induction motors for predictive maintenance and monitoring. *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS*, 2002.

[32] Bin Liu, Zheyuan Ding, Junfeng Wu, and Li Yao. Ontology-based fault diagnosis: A decade in review. In *ACM International Conference Proceeding Series*, pages 112–116. Association for Computing Machinery, 1 2019.

[33] Toufik Berredjem and Mohamed Benidir. Bearing faults diagnosis using fuzzy expert system relying on an Improved Range Overlaps and Similarity method. *Expert Systems with Applications*, 108:134–142, 10 2018.

[34] Chao Cheng, Jiuhe Wang, Hongtian Chen, Zhiwen Chen, Hao Luo, and Pu Xie. A review of intelligent fault diagnosis for high-speed trains: Qualitative approaches, 1 2021.

[35] Peng Yingzhou and Wang Huai. Application of Digital Twin Concept in Condition Monitoring for DC-DC Converter. Technical report, IEEE, 2019.

[36] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X. Gao. Deep learning and its applications to machine health monitoring, 1 2019.

[37] Alexander Jung. *Machine Learning The Basics*. Springer, Espoo, 1 edition, 2022.

[38] Basna Mohammed Salih Hasan and Adnan Mohsin Abdulazeez. A Review of Principal Component Analysis Algorithm for Dimensionality Reduction. *Journal of Soft Computing and Data Mining*, 02(01), 4 2021.

[39] Elena Quatrini, Silvia Colabianchi, Francesco Costantino, and Massimo Tronci. Clustering Application for Condition-Based Maintenance in Time-Varying Processes: A Review Using Latent Dirichlet Allocation, 1 2022.

[40] Lizheng Pan, Dashuai Zhu, Shigang She, Aiguo Song, Xianchuan Shi, and Suolin Duan. Gear fault diagnosis method based on wavelet-packet independent component analysis and support vector machine with kernel function fusion. *Advances in Mechanical Engineering*, 10(11), 11 2018.

[41] John Cristian Borges Gamboa. Deep Learning for Time-Series Analysis. Technical report, University of Kaiserslautern, Kaiserslautern, 1 2017.

[42] Christopher M Bishop. Neural Networks for Pattern Recognition. Technical report, CLARENDON PRESS, OXFORD, 1995.

[43] Taiwo Oyedare and Jung-Min Park. Estimating the Required Training Dataset Size for Transmitter Classification Using Deep Learning. *IEEE*, 2019.

[44] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *IEEE*, 2017.

[45] Aurelien Geron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O´Reilly, Sebastopol, 2 edition, 2019.

[46] Alberto Fernández, Salvador García, Francisco Herrera, and Nitesh V Chawla. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*, 61:863–905, 2018.

[47] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-Scale Convolutional Neural Networks for Time Series Classification. *ArXiv*, 3 2016.

[48] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time Series Data Augmentation for Deep Learning: A Survey. *ArXiv*, 2 2020.

[49] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data Augmentation for Time Series Classification using Convolutional Neural Networks. Technical report, HAL Open Science, Riva Del Garda, 2016.

[50] Tailai Wen and Roy Keyes. Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning. *ArXiv*, 5 2019.

[51] Russel Reed and Robert Marks. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, 2 1999.

[52] Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on MNIST classification task. *ArXiv*, 4 2018.

[53] Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. Artificial intelligence for fault diagnosis of rotating machinery: A review, 8 2018.

[54] Pier Francesco Orrù, Andrea Zoccheddu, Lorenzo Sassu, Carmine Mattia, Riccardo Cozza, and Simone Arena. Machine learning approach using MLP and SVM algorithms for the fault prediction of a centrifugal pump in the oil and gas industry. *Sustainability (Switzerland)*, 12(11), 6 2020.

[55] Maamar Al Tobi, Geraint Bevan, Peter Wallace, David Harrison, and Kenneth Eloghene Okedu. Faults diagnosis of a centrifugal pump using multilayer perceptron genetic algorithm back propagation and support vector machine with discrete wavelet transform-based feature extraction. *Computational Intelligence*, 37(1):21–46, 2 2021.

[56] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *ArXiv*, 3 2020.

[57] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th international conference on Machine learning*. Proceedings of the 25th international conference on Machine learning, 2008.

[58] Jinwon An and Sungzoon Cho. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. Technical report, Special Lecture on IE, 2015.

[59] Huijie Zhu, Rui Ting, Xinqing Wang, You Zhou, and Husheng Fang. Fault diagnosis of hydraulic pump based on stacked autoencoders. In *2015 IEEE 12th International Conference on Electronic Measurement and Instruments, ICEMI 2015*, volume 1, pages 58–62. Institute of Electrical and Electronics Engineers Inc., 6 2016.

[60] Jiedi Sun, Changhong Yan, and Jiangtao Wen. Intelligent bearing fault diagnosis method combining compressed data acquisition and deep learning. *IEEE Transactions on Instrumentation and Measurement*, 67(1):185–195, 1 2018.

[61] Dominik Scherer, Andreas Müller, and Sven Behnke. LNCS 6354 - Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. Technical report, Springer, 2010.

[62] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Technical report, ArXiv, 2015.

[63] Wei Zhang, Gaoliang Peng, Chuanhao Li, Yuanhang Chen, and Zhujun Zhang. A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. *Sensors (Switzerland)*, 17(2), 2 2017.

[64] Gang Mao, Zhongzheng Zhang, Bin Qiao, and Yongbo Li. Fusion Domain-Adaptation CNN Driven by Images and Vibration Signals for Fault Diagnosis of Gearbox Cross-Working Conditions. *Entropy*, 24(1), 1 2022.

[65] Zhihao Chen, Jian Cen, and Jianbin Xiong. Rolling Bearing Fault Diagnosis Using Time-Frequency Analysis and Deep Transfer Convolutional Neural Network. *IEEE Access*, 8:150248–150261, 2020.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 6 2017.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *ArXiv*, 2 2015.

[68] Ken Binmore and Joan Davies. *Calculus: Concepts and Methods*. Cambridge University Press, illustrated, reprint edition, 2001.

[69] Kochenderfer Mykel and Wheeler Tim. Algorithms for Optimization. *MIT Press*, 2019.

[70] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ArXiv*, 12 2014.

[71] Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de las Heras. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231, 7 2019.

[72] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 6 2006.

[73] Lei Tang, Suju Rajan, and Vijay Narayanan. *Large Scale Multi-Label Classification via MetaLabeler*. ACM, 2009.

[74] Rong-En Fan and Chih-Jen Lin. A Study on Threshold Selection for Multi-label Classification. Technical report, National Taiwan University, 2007.

[75] C J Van Rijsbergen. FOUNDATION OF EVALUATION. *Journal of Documentation*, 30(4):365–373, 1974.

A NPSH Test Run

Measurement no.	n [rpm]	Test type	Q/Qref*	H/Href*	Fr/Fr_max**	Calculated deflection***	Contact
1	1000	NPSH	1.00	1.00	0.06	0.06	No
2	1000	NPSH	0.75	1.15	0.21	0.20	No
3	1000	NPSH	0.50	1.19	0.29	0.29	No
4	1000	NPSH	0.25	1.22	0.39	0.39	No
5	1000	NPSH	1.00	1.00	0.06	0.06	No
6	1000	NPSH	0.75	1.15	0.21	0.20	No
7	1000	NPSH	0.50	1.19	0.29	0.29	No
8	1000	NPSH	0.25	1.22	0.39	0.39	No
9	1500	NPSH	1.00	1.00	0.14	0.14	No
10	1500	NPSH	0.75	1.13	0.46	0.46	No
11	1500	NPSH	0.50	1.20	0.66	0.65	No
12	1500	NPSH	0.25	1.21	0.89	0.88	No
13	1500	NPSH	1.00	1.00	0.14	0.14	No
14	1500	NPSH	0.75	1.13	0.46	0.46	No
15	1500	NPSH	0.50	1.20	0.66	0.65	No
16	1500	NPSH	0.25	1.21	0.89	0.88	No
17	2000	NPSH	1.00	1.00	0.25	0.24	No
18	2000	NPSH	0.75	1.13	0.82	0.81	No
19	2000	NPSH	0.50	1.20	1.17	1.16	Yes
20	2000	NPSH	0.25	1.23	1.58	1.56	Yes
21	2000	NPSH	1.00	1.00	0.25	0.24	No
22	2000	NPSH	0.75	1.13	0.82	0.81	No
23	2000	NPSH	0.50	1.20	1.17	1.16	Yes
24	2000	NPSH	0.25	1.23	1.58	1.56	Yes

* Values are given as a relation to the best efficiency point value for that rotational speed.

** Fr_max is the maximum force before contact.

*** Values for the deflection are a ratio between calculated deflection and clearance.

B QH Test Run

Measurement no.	n [rpm]	Test type	Q/Qref*	H/Href*	Fr/Fr_max**	Calculated deflection***	Contact
1	1500	Q-H	1.00	1.00	0.2	0.20	No
2	1500	Q-H	0.75	1.15	0.7	0.65	No
3	1500	Q-H	0.50	1.19	0.9	0.93	No
4	1500	Q-H	0.25	1.22	1.3	1.25	Yes
5	1500	Q-H	1.00	1.00	0.2	0.20	No
6	1600	Q-H	1.00	1.00	0.2	0.22	No
7	1600	Q-H	0.75	1.03	0.8	0.74	No
8	1600	Q-H	0.35	1.06	1.3	1.28	Yes
9	1600	Q-H	0.25	1.10	1.4	1.42	Yes
10	1600	Q-H	1.00	1.00	0.2	0.22	No
11	1800	Q-H	1.00	1.00	0.3	0.28	No
12	1800	Q-H	0.75	1.69	1.0	0.94	No
13	1800	Q-H	0.40	1.92	1.5	1.51	Yes
14	1800	Q-H	0.25	2.03	1.8	1.80	Yes
15	1800	Q-H	1.00	1.00	0.3	0.28	No
16	2000	Q-H	1.00	1.00	0.4	0.35	No
17	2000	Q-H	0.50	1.20	1.7	1.65	Yes
18	2000	Q-H	0.40	0.14	1.9	1.86	Yes
19	2000	Q-H	1.00	1.00	0.4	0.35	No
20	2200	Q-H	1.00	1.00	0.4	0.42	No
21	2200	Q-H	0.60	1.01	1.8	1.75	Yes
22	2200	Q-H	0.50	1.02	2.0	1.99	Yes
23	2200	Q-H	0.25	1.03	2.7	2.69	Yes
24	2200	Q-H	1.00	1.00	0.4	0.42	No

* Values are given as a relation to the best efficiency point value for that rotational speed.

** Fr_max is the maximum force before contact.

*** Values for the deflection are a ratio between calculated deflection and clearance.