Datagedreven prestatiemonitoring, foutdetectie en dynamische dashboards voor offshorewindmolenparken

Data-Driven Performance Monitoring, Fault Detection and Dynamic Dashboards for Offshore Wind Farms

Olivier Janssens

Promotoren: prof. dr. ir. S. Van Hoecke, prof. dr. ir. M. Loccufier
Proefschrift ingediend tot het behalen van de graad van
Doctor in de industriële wetenschappen: elektronica-ICT

UNIVERSITEIT
GENT

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. R. Van de Walle
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2016 - 2017

ISBN 978-94-6355-019-2
NUR 984
Wettelijk depot: D/2017/10.500/54

Examination Board

prof. dr. ir. Patrick De Baets (Chairman)
Faculty of Engineering and Architecture
Department of Electrical Energy, Metals, Mechanical Constructions and Systems.
Ghent University

prof. dr. ing. Kurt Stockman (Secretary)
Faculty of Engineering and Architecture
Department of Electrical Energy, Metals, Mechanical Constructions and Systems.
Ghent University

prof. dr. ir. Christof Devriendt
Faculty of Applied Sciences
Department of Mechanical Engineering
Vrije Universiteit Brussel

dr. ir. David Verstraeten
Yazzoom

prof. dr. ir. Aleksandra Pizurica
Faculty of Engineering and Architecture
Department of Telecommunications and information processing
Ghent University

prof. dr. ir. Tom Dhaene
Faculty of Engineering and Architecture
Department of Information technology
Ghent University-imec

dr. ir. Femke Ongenae
Faculty of Engineering and Architecture
Department of Information technology
Ghent University-imec

prof. dr. ir. Sofie Van Hoecke
Faculty of Engineering and Architecture
Department of Electronics and Information Systems
Ghent University-imec

prof. dr. ir. Mia Loccufier
Faculty of Engineering and Architecture
Department of Electrical Energy, Metals, Mechanical Constructions and Systems.
Ghent University

Acknowledgments

It is the hottest fire that forms the sternest steel.

Pierce Brown, Red Rising

A journey of a thousand miles begins with a single step; I took that step a little over 4 years ago. The person who guided me during this journey was Prof. Sofie Van Hoecke, a dedicated and hardworking but also patient researcher, educator and mentor. She is a rare person who is available almost around the clock for valuable advice. Because of her, I had the opportunity to work on several interesting projects and even guide students myself. I am very grateful to have had her as my promoter, and I wouldn't have wanted it any other way. After two years of research, my research topic changed direction and we required the help of someone with vast knowledge in mechanical engineering. Luckily, Prof. Mia Loccufier was available to become my second promoter. Mia possesses unwavering calmness and never ending curiosity backed by decades of experience. This combination of traits results in a very friendly person who is uncommonly good at arguing (as in "to give the reasons for your opinion, idea or belief"). Whenever I had long discussions or arguments with Mia, this joke came to mind: "Arguing with an engineer is a lot like wrestling in the mud with a piggy, after a couple of hours you realize he likes it." By having in-depth discussions about the research, plans of action sprung to life, and the research thrived. I am very thankful to Mia for acting as my promoter, and I can't imagine having succeeded without Mia's help.

Besides my promoters, many other people gave valuable feedback and advice. I would like to thank the members of my examination board: Prof. Patrick De Baets, Prof. Kurt Stockman, Prof. Christof Devriendt, Dr. David Verstraeten, Prof. Aleksandra Pizurica, Prof. Tom Dhaene and Dr. Femke Ongenae. Additionally, I had a great environment with great colleagues, ranging from my office colleagues consisting of Azarakhsh, Baptist, Florian, Fréderic, Jasper and Mi Jung, to the colleagues at the lunch table: Gerald, Glenn, Johan, Kristof, Laura, Martin, Niels, Steven, Tom, Ruben. I also cannot forget to thank the people with whom I talked sporadically, such as Bram, Benjamin, Dieter, Erik, Johan Beke, Ronny and Wesley. These people provided food for thought or appreciated distractions. Appreciated distractions were also provided by friends and family: Bart, Charlotte, David, John, Mathieu, Patricia, Silke and Tessa, for which I am grateful. Not included in this list are my parents, who especially deserve to be acknowledged for

more than just the years of my PhD research. Thank you for everything!

Finally, I would like to thank Steve, my husband; you're the bee's knees. I look forward to celebrating our cotton anniversary and, in the future, all the other anniversaries as well.

June 2017
Olivier Janssens

Table of Contents

List of Acronyms

A

API	Application Programming Interface
AU	Area Under

C

CFD	Computational Fluid Dynamics
CM	Condition Monitoring

B

BPFO	Ball Pass Frequency Of The Outer Raceway

D

DL	Deep Learning
DE	Differential Evolution
DFT	Discrete Fourier Transform

E

EWEA	European Wind Energy Association

ERT	Extremely Randomized Trees
EILB	Extremely Inadequately Lubricated Bearing
EV	Explained Variance

G

GC	Gini Coefficient

H

HP	Hard Particles
hdb	historical database
HB	Healthy Bearing

I

IM	Imbalance
IoT	Internet of Things
IRT	Infrared Thermal
IR	Infrared
IRCS	Infrared Cumulative Sum

K

kb	knowledge base
KNN	K-Nearest Neighbors

L

LOE	Line Of Equality

M

M_{20}	Moment Of Light
MILB	Mildly Inadequately Lubricated Bearing
MOB	Method Of Bins
MVC	Model View Controller

N

NN	Neural Network

O

O&M	Operations and Maintenance
ORF	Outer Raceway Fault

P

PCA	Principal Component Analysis
PL	Parameter Logistics

R

REB	Rolling Element Bearing
ReLU	Rectified Linear Unit
RDF	Resource Description Framework
RF	Random Forest
RFC	Random Forest Classifier
RMS	Root Mean Square
ROI	Region Of Interest
RPM	Rotations Per Minute
RoR	Ruby on Rails

S

SCADA	Supervisory Control And Data Acquisition

SD	Standard Deviation
SGBRT	Stochastic Gradient Boosted Regression Trees
SPC	Statistical Process Control
SVM	Support Vector Machine

U

URI	Uniform Resource Identifier

V

VIB	Vibration

Samenvatting

Tegen 2020 heeft de Europese Unie zich geëngageerd gebruik te maken van 20 % hernieuwbare energie. Deze doelstelling kan alleen maar gehaald worden als de individuele lidstaten hun capaciteit aan hernieuwbare energie uitbreiden door middel van bv. windenergie. Windenergie is namelijk een onuitputbare energiebron die zowel on- als off-shore geoogst kan worden, met een verwaarloosbare impact op het milieu. Landen zoals Groot-Brittannië, Nederland en België hebben het voordeel gebruik te kunnen maken van kustwateren voor het installeren van offshorewindturbines om zo gemakkelijker deze doelstelling te kunnen halen aangezien offshorewindturbineparken meer energie dan de onshorewindturbineparken produceren.

Vandaag de dag is de kost voor het oogsten van offshore windenergie nog aanzienlijk hoog. Dit komt voornamelijk door de exploitatie- en onderhoudskosten van deze offshore windturbines die dubbel zo hoog zijn als die van onshore windturbines. Om deze kosten te drukken, is er nood aan efficiënter onderhoud en een omschakeling van correctief naar predictief onderhoud. Predictief onderhoud vereist het continu inschatten van de gezondheidsstatus van de windturbine en zijn componenten door gebruik te maken van online/continue condition monitoring.

Continue condition monitoring van windturbines vormt echter een complexe taak, waarbij een grote kennis vereist is van machinebouw en elektrotechniek. Om de drempel naar continue monitoring te verlagen, worden datagedreven technieken voorgesteld in dit proefschrift die gebruik maken van de sensordata opgemeten in de windturbines en beperkte expertkennis. In dit proefschrift worden datagedreven technieken ontwikkeld voor het monitoren op drie niveaus: het windmolenpark, de individuele windturbine, en op niveau van de componenten in een turbine.

Hoofdstuk twee bespreekt datagedreven prestatiemonitoring van volledige windturbines. De gezondheidstoestand van een windturbine is direct gerelateerd met de energieproductie. Om problemen met de windturbine te kunnen vaststellen, moeten afwijkingen in deze energieproductie kunnen worden gedetecteerd. Dit wordt gedaan door een model te creëren gebruikmakende van machinaal leren en windturbine data gecapteerd tijdens normale operationele condities. Vervolgens kan het model worden gebruikt om de energieproductie te voorspellen en die waarde te vergelijken met de feitelijke energieproductie om afwijkingen te detecteren.

Bestaande technieken maken enkel gebruik van de windsnelheden om de energieproductie te voorspellen waardoor de resultaten niet optimaal zijn. Om deze

voorspellingsresultaten te verbeteren, wordt in dit hoofdstuk naast de windsnelheid eveneens de windrichting, draairichting van de turbine, de hoek van de wieken en de rotatiesnelheid van de wieken als input parameters gebruikt. Dit laat toe de fout op de voorspelling met 27.66 % te verminderen. Daarnaast wordt aangetoond dat deze datagedreven technieken inzicht geven in de verschillende significante variabelen en bijhorende interactie om de energieproductie te voorspellen. Dankzij de verbeteringen van deze voorspellingstechnieken, kunnen de afwijkingen van de energieproductie van de windturbines beter gedetecteerd worden.

Prestatiemonitoring geeft enkel informatie weer over de globale staat van de windturbine. Indien de windturbine de verwachte energieproductie niet haalt, moet de oorzaak van het probleem worden achterhaald om preventieve onderhoudsacties te ondernemen. De component die voor de meeste stilstand zorgt bij een windturbine is de versnellingsbak. Binnen de versnellingsbak zijn het dan weer de lagers die voor de meeste storingen zorgen. Daarom wordt in hoofdstuk drie onderzoek gedaan naar datagedreven technieken om fouten en schade in lagers te detecteren. Eén van de meest voorkomende oorzaken van lagerschade is ontoereikende smering. Dit kan zowel te veel als te weinig smering zijn, maar ook vervuilde smering. Huidige detectietechnieken die werken a.d.h.v. trillingen, hebben moeite om deze smeringsproblemen te detecteren. Online olieanalyse kan deze smeringsproblemen wel detecteren, maar is dan weer een heel dure techniek. Daarom wordt in hoofdstuk drie onderzoek gedaan naar het potentieel van warmtecameras als nieuwe sensor. Aangezien smering gerelateerde fouten met meer wrijving en dus hitte gepaard gaan, lijken warmtecameras de ideale sensor om smering gerelateerde problemen mee te detecteren en op te sporen. Foutdetectie op basis van warmtecameras staat momenteel nog in zijn kinderschoenen. Het monitoren op offshore windturbines zou bijgevolg te veel ongecontroleerde parameters introduceren om degelijke conclusies te kunnen trekken. Daarom werden de testen met warmtecameras in dit proefschrift op een gecontroleerde labo-opstelling uitgevoerd. Er werden twee grote datasets gecreëerd die zowel warmtebeelden als trillingen bevatten. De geeïntroduceerde schade en problemen in de opstelling waren: gezonde lagers, verminderende smering, extreem verminderde smering, vervuilde smering en schade aan de buitenring van het lager. In plaats van deze condities apart te beschouwen, werd elk van deze fouten gecreëerd in combinatie met verschillende gradaties van onbalans van de machine.

In hoofdstuk drie worden vervolgens datagedreven systemen voorgesteld om deze verschillende combinaties van condities automatisch te detecteren. Het eerste systeem gebruikt enkel warmtebeelden, het tweede systeem enkel trillingsmetingen en het derde systeem combineert beide sensorische signalen/metingen. Elk systeem extraheert hierbij vooraf gedefinieerde signaalkarakteristieken, die ontworpen zijn geweest aan de hand van feature engineering. De systemen die gebruikmaken van ofwel warmtebeelden ofwel trillingsdata vertonen elk hun sterke en zwakke kenmerken. Wanneer enkel warmtebeelden worden gebruikt, is het bijv. moeilijker om schade aan de buitenring van de lager te detecteren, wat dan weer accurater gedetecteerd wordt door het trillingsgebaseerd systeem. Het systeem op basis van trillingen heeft het daarentegen moeilijker om smering gerelateerde con-

dities te detecteren. Het combineren van deze twee types sensorische data in een multi-sensor datagedreven systeem, brengt het beste van beiden samen en overtreft elk van beide single sensor oplossingen op vlak van accuraatheid.

In hoofdstuk drie werden methodes gemaakt om specifieke signaalkarakteristieken uit sensordata te halen voor bepaalde condities en fouten. Deze aanpak heeft twee nadelen. De eerste is dat als de fout of condition niet (volledig) gekend is, er geen methodes kunnen gemaakt worden. Het tweede nadeel bestaat uit het feit dat als een nieuwe fout of conditie opeens zou moeten kunnen detecteerbaar zijn, nieuwe methodes aan het systeem toegevoegd moeten worden. Om deze problemen op te lossen wordt in hoofdstuk vier feature learning voorgesteld. Het grote voordeel hiervan is dat heel wat minder vakkennis vereist is waardoor fouten, waarvoor experten mogelijks nog geen signaalkarakteristieken ontworpen hebben, toch gedetecteerd kunnen worden. In hoofdstuk vier wordt feature learning gerealiseerd door convolutionele neurale netwerken te onderzoeken, creëren en vervolgens toe te passen op de datasets uit hoofdstuk drie. Ook hier worden terug twee systemen onderzocht met enkel warmtebeelden, resp. enkel trillingsdata, alsook een multi-sensor systeem die zowel warmtebeelden als trillingsdata gebruikt. Opnieuw kan het multi-sensorische systeem de condities accurater detecteren. Daarenboven detecteert het multi-sensor feature learning systeem de fouten beter dan wanneer het feature engineering systeem uit hoofdstuk drie wordt toegepast. Naast een verhoogde accuraatheid, leveren de feature learning systemen ook inzichten in de data wat bijzonder interessant is in het geval van de thermische beelden aangezien er vandaag de dag nog geen volledig fysisch gebaseerde modellen bestaan die de warmteprocessen accuraat beschrijven.

Wanneer een volledig windturbinepark wordt beschouwd, moeten de modellen uit de hoofdstukken twee tot vier getuned worden voor elke turbine in het windmolenpark, en zijn er bijgevolg veel modellen en bijhorende monitoringsdata nodig. Om in dat geval het overzicht over het volledige windturbinepark te bewaren, wordt in hoofdstuk vijf een dynamisch dashboard voorgesteld. Dynamische dashboards kunnen automatisch data van sensoren koppelen met visualisatie opties door gebruik te maken van Semantische Web technologieën en alles als web API aan te bieden. Door semantische beschrijvingen toe te voegen aan de web APIs, kan de reasoner in de dashboard engine automatisch sensorische data koppelen met mogelijke visualisatie opties, en bijgevolg zonder veel moeite worden gevisualiseerd. Aangezien de dashboard engine automatisch alle sensoren en visualisatie web APIs ontdekt, kunnen nieuwe functionaliteiten met weinig moeite worden toegevoegd aan het dashboard. De enige vereiste is dat de nieuwe sensoren voorzien worden van een web API en semantische beschrijving. Door ook de datagedreven modellen als soft sensor toe te voegen in het dashboard, is het eenvoudig om een overzicht te krijgen van een windturbinepark en de bijhorende online monitoring services, essentieel voor de ondersteuning van preventief onderhoud.

Dit proefschrift zet belangrijke stappen naar het ondersteunen van preventief onderhoud van offshore windturbines door aan te tonen dat datagedreven prestatiemonitoring en foutdetectie technieken kunnen verbeterd worden en dat dynamische dashboards makkelijk een overzicht van de data kan geven. De hogere

accuraatheid en het vroegtijdiger detecteren van problemen, maken predictief onderhoud mogelijk en kan uiteindelijk leiden tot een verlaging van de exploitatie- en onderhoudskosten van offshore windturbines, wat op zich een verlaging in prijs van hernieuwbare energie met zich mee kan brengen. Hierdoor wordt het nog interessanter te investeren in duurzame windenergie en zetten we verder stappen naar een groenere wereld.

Summary

The European Union has set a 20 % renewable energy target for 2020. To achieve this goal, individual member states are expanding their renewable energy capacity. An important energy source to achieve this goal is wind energy. Wind energy is a constantly replenishing source of energy and harvesting this energy has a negligible impact on the environment. Countries such as the United Kingdom, the Netherlands and Belgium have the advantage of having coastal waters to install offshore wind turbine farms as offshore wind energy can provide significantly more energy than onshore wind energy.

To harness the energy in the wind, offshore wind turbines are build and installed. However, offshore wind energy is currently still expensive, partially due to the required maintenance. For example, the maintenance costs for an offshore wind turbine is twice as high compared to the maintenance costs of an onshore wind turbine. To lower the operation and maintenance costs of offshore wind turbines, more effective maintenance strategies are required. A transition has to be made from corrective maintenance to predictive maintenance. Predictive maintenance can only be done by continuously estimating the state of the turbine and its components using a continuous/online condition monitoring system.

Continuous condition monitoring of a wind turbine is a complex task and requires substantial expertise knowledge on condition monitoring, mechanical engineering and electrical engineering. However, currently, not all possible faults and conditions are completely understood and defined. To overcome this hurdle, within this dissertation, data-driven techniques are proposed to make a first step towards condition monitoring of complex machines. Data-driven techniques use data captured from wind turbines in order to monitor their health. As solving a large complex problem is difficult as a whole, in this dissertation the monitoring task is split into three parts namely, component-level, machine-level and fleet-level.

In chapter two, the focus lies on performance monitoring of offshore wind turbines (machine-level). To estimate the overall condition of an offshore wind turbine, the power output has to be monitored. To do this, machine learning models are constructed and trained on data from offshore wind turbines during normal operational conditions. The goal of this phase is to have a model that can predict the expected behavior of the wind turbine. Subsequently, the predicted output can be compared to the actual generated output. As currently power prediction is mainly done by modeling the relationship between the measured wind speed

and the output power, the results are sub-optimal. Within this chapter, not only the wind speed is considered as input, but also the wind direction, wind turbine yaw, blade pitch and the rotor's rotations per minute to predict the output power of the wind turbine. Six algorithms are compared regarding their predictive performance. We conclude that by using these additional parameters together with stochastic gradient boosted regressions trees, power production prediction can be done, reducing the error of the prediction up to 27.66 %. Furthermore, we show that this data-driven technique enables to extract knowledge from the data, such as the importance of the variables to predict the output power and the interactions between the individual variables. In the end, improved power prediction can improve anomaly detection on the electrical signal of offshore wind turbines.

Performance monitoring only provides information about the overall health of the wind turbine. However, if the machine is not performing as expected, the cause of the problem has to be identified to enable a preemptive maintenance action. One of the components, that has the largest impact on the amount of downtime of a wind turbine, is the gearbox, and within the gearbox, the bearings are the major cause of failures. Hence, in chapter three, data-driven fault detection systems are researched and developed to detect rolling element bearing faults. One of the most frequently occurring root-causes of bearing failure is lubricant inadequacy. This can be both under or over lubrication, but also contaminated lubricant. However, using current state-of-the-art fault detection systems, such as vibration analysis, these types of faults are difficult to detect. Another option to detect these conditions is to use online oil analysis which requires a very large investment. Hence, in chapter three a relatively new type of sensory device is used, i.e. infrared thermal camera. Infrared thermal cameras are the ideal sensory device to detect faults related to lubricant. Sub-optimal lubricant conditions, such as under lubricant will increase the friction in the bearing resulting in additional heat to be generated which can be observed by the infrared thermal camera. Infrared thermal imaging-based fault detection is still in its infancy and applying it on a live offshore wind turbine, would be too uncontrolled to draw any conclusions. Hence, tests were done on a lab-scale setup. Within this dissertation, two large data sets containing both infrared thermal imaging recordings and vibration records were created. The faults/conditions considered are healthy bearings, under lubrication, extreme under lubrication, lubricant contamination and outer-raceway faults. Moreover, instead of considering each condition separately, all conditions are tested during different gradations of rotor imbalance.

In chapter three, single sensor systems are created to detect these faults. The first single sensor system solely uses infrared thermal imaging and the second solely uses vibration measurements. Each system extracts features from the respective data stream and subsequently provides the features to a machine learning algorithm, which in this case is a random forest classifier. Both single sensor systems exhibit a specific strength and weakness. The thermal infrared-based system on hand is less capable to detect outer-raceway faults, on the other hand, it can detect lubricant related conditions very well. In contrast, the vibration-based system performs less well when detecting lubricant related condition, but it performs

very well for the detection of outer-raceway faults. Hence, in the end, the best of both systems is combined resulting in a multi-sensor data-driven system which can outperform both single sensor systems.

To create the systems in chapter three, features had to be engineered for each specific sensor stream and machine condition. This approach has two downsides. First, feature extraction methods have to be engineered for every fault and condition. If the fault/condition is not completely understood or known, it is difficult to create features. Second, if the system suddenly should be able to detect a new fault/condition, possibly new feature extraction methods will have to be designed and added to the system. Hence, to remove these disadvantages, in chapter four, the focus is shifted from feature engineering to feature learning. More specifically, convolutional neural networks are developed and applied to detect the various condition using the different data streams. We show that these convolutional neural network-based systems –applied on raw data– can outperform the feature engineering-based systems. Furthermore, these systems enable insights to be gained into the data. This is especially interesting for the infrared thermal data as complete and accurate physical models are currently not available. Similar to chapter three, in chapter four the two single sensor systems are combined in one (feature learning) multi-sensor system outperforming the single sensor systems, but also outperforming the feature engineered system. By considering a new type of sensor and creating a data-driven multi-sensor feature learning system, fault detection can become more accurate and less dependent on expert knowledge hence reducing the entrance barrier to apply continuous monitoring.

Both chapter two and three enable to monitor detailed aspects of a single turbine. However, when considering an entire wind farm, many models are required to monitor all wind turbines. In order to get an overview of all the models and generated data of an entire wind farm, dynamic dashboards are introduced in chapter five. Dashboards enable users to visualize a plethora of data. A typical static dashboard requires significant work to build and considerable effort to connect the different sensor streams to the dashboard. Furthermore, when considering soft sensors, additional labor is required to make the soft sensor functional. Soft sensors are inferential estimators, generating values based on actual measurements. In this dissertation the machine learning systems proposed in chapter two, three and four can be considered as soft/virtual sensors as they process measurements into predictions.

Dynamic dashboards, opposed to static dashboard, require little manual configuration. Dynamic dashboards can automatically connect sensor streams to visualization options using semantic web technologies. In this chapter we encapsulate sensors as well as a visualization library in web APIs to web-enable them. Moreover, a semantic description is provided to the web APIs so that the reasoner, within the dynamic dashboard's broker, can match the sensor streams to possible visualization options. By doing so, both real sensor streams as well as virtual sensor streams can be visualized effortlessly. Finally, new sensors and visualization options can easily be added to the dashboard. They simply have to be encapsulated in a web API and have a semantic description. Hence, dynamic dashboards

enable a user to easily get an overview of an entire wind farm, which is essential for predictive maintenance.

In this dissertation, performance monitoring and fault detection for offshore wind turbines are improved. We show that data-driven techniques combined with dynamic dashboards lower the entrance barrier to condition monitoring. This will enable more faults to be detected earlier, enabling predictive maintenance and hence possibly lowering the operation and maintenance costs of offshore wind turbines. When the operation and maintenance costs are lowered of offshore wind turbines, the price of electrical energy generated by offshore wind can drop making investments in renewable (wind) energy even more attractive. By doing so, we continue to progress towards a greener world.

Chapter 1

Introduction

In this chapter, an introduction is provided where the importance of the research, presented in this dissertation, is highlighted. Subsequently, an overview of the research is presented together with the contributions in each chapter.

1.1 Situating the importance of renewable energy and offshore wind turbines

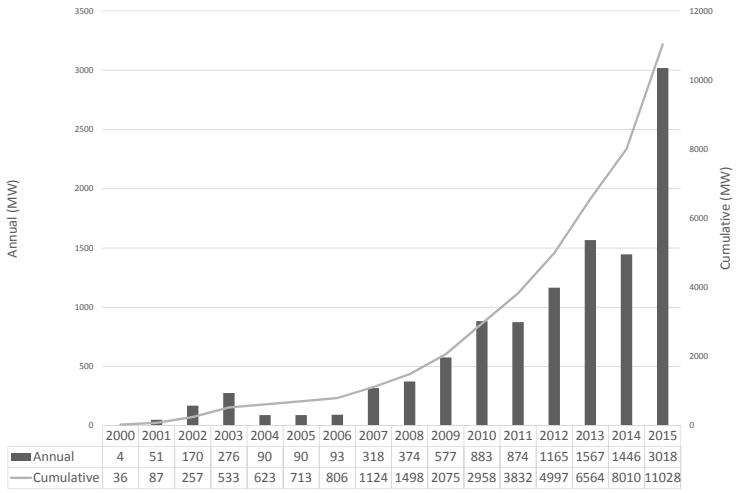
On a global scale, mankind heavily relies on non-renewable energy sources such as natural gas and fossil fuel. These resources are finite and will eventually dwindle, becoming too expensive or too environmentally damaging to extract. Even nuclear energy, which is still under debate if it can be considered renewable or nonrenewable, is damaging for the environment.

In contrast, renewable energy sources have a limited impact on the environment and are constantly replenished. Examples of renewable energy sources are solar, wind, biomass, geothermal, hydrogen, waves and hydropower. The significance of renewable energy has increased and will keep increasing over the coming decades as the European Union has set a 20 % renewable energy target for 2020 [6]. Due to the climate in Europe, one of the major renewable energy sources is wind energy. Hence, with Europe's 2020 goal in mind, the European Wind Energy Association (EWEA) proposed three possible growth scenarios for wind energy towards 2020. The three scenarios project that there will be an increase in installed wind turbine capacity of 41 %; 64 %; or 84.9 % respectively, compared to 2013 [6].

Currently, in Belgium there is an installed wind energy capacity of 2229 MW of which 712 MW is installed offshore and 1517 MW is installed onshore. Wind energy accounts for 6.7 % of the national electric production [7]. By 2020, the total land-based installed capacity should double to 3000 MW, and an additional 2271 MW is planned offshore, for a possible total of 5983 MW. This will help

achieve the target of 13 % of renewable energy for Belgium [7].

In Belgium, the installed onshore capacity should double in the coming years, but the installed capacity offshore, should triple. According to the World Energy Council, offshore wind is generally 8 m/s higher at European coastal waters compared to those onshore [8]. Hence, offshore wind energy will play a major role to achieve Belgium's target. The increase in installed offshore capacity is also noticeably rising in whole Europe as can be seen in Figure 1.1. This increase is also noticeable financially, as the total investments in offshore wind in 2015 were more than 18bn Euro, which is a record year in terms of total committed funds [9]. As can be seen, offshore wind energy is important and will become more important in the future.



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Annual	4	51	170	276	90	90	93	318	374	577	883	874	1165	1567	1446	3018
Cumulative	36	87	257	533	623	713	806	1124	1498	2075	2958	3832	4997	6564	8010	11028

Figure 1.1: Installed capacity of offshore wind turbines in Europe [8].

1.2 Operations and Maintenance

Currently, the cost of offshore wind energy is still high as Figure 1.2 indicates. Generally, this is due to the operations and maintenance (O&M) costs which account for up to 25 % of the levelised costs of energy [10][1]. When comparing the

[1]The per-kilowatthour cost of building and operating a wind turbine over an assumed financial life and duty cycle. Key inputs to calculating LCOE include capital costs, fixed and variable operations and

O&M costs between offshore and onshore wind turbines, in Figure 1.2, it can be seen that offshore O&M costs are twice as expensive. Offshore turbines are more difficult to access and require specialized vessels resulting in a prolonged downtime.

There are several maintenance strategies when it comes to wind turbines and machines in general. The first is corrective maintenance where maintenance is only carried out when a problem occurred and subsequently has been detected. This type of maintenance results in the highest costs due to the long downtime and the sudden need for maintenance. The second type of maintenance is preventive maintenance which requires maintenance to be done periodically to prevent failure. This type of maintenance can be less expensive as it is possible to group maintenance actions and prevent failure of critical components hence reducing the downtime. Finally, there is predictive maintenance which requires an online condition monitoring systems that continuously estimates the condition of the machine and possibly components. Due to such a condition monitoring system, maintenance can be scheduled more efficiently and effectively reducing the overall O&M costs.

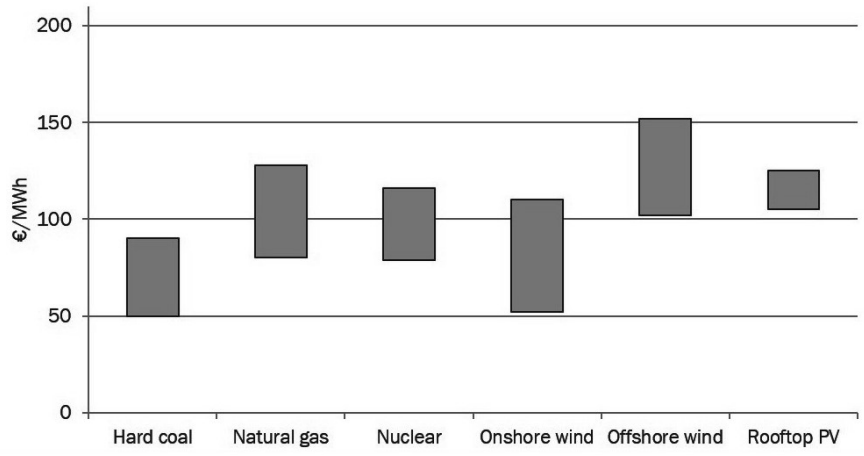


Figure 1.2: Levelised cost of electricity of major generation technologies in Europe [11].

However, due to the complexity of wind turbines in itself and the according wind farms, predictive maintenance is a challenging task. First steps towards condition monitoring and predictive maintenance are made within this dissertation at three major levels: machine-level (1.3), component-level (1.4) and fleet-level (1.5).

maintenance (O&M) costs, financing costs, and an assumed utilization rate

1.3 Machine-level: performance monitoring

In order to monitor the overall health of a wind turbine without requiring significant knowledge on the inner workings of a wind turbine, performance monitoring is done. Performance monitoring allows to quickly assess if a turbine is performing as it should and hence is a good first method to detect possible suboptimal conditions before considering the conditions of all the components. Only when sub-optimal performance is detected, the individual components have to be considered to found out what the cause of under-performance is (see Chapter 3 and 4).

To apply performance monitoring on wind turbines, the power production of the wind turbine has to be monitored for unexpected deviations. In practice, this can be done by comparing the expected output power to the actual produced output power. To determine what output power is expected, often only the wind speed is considered. However, in this dissertation multiple variables are considered to determine the expected output power. The research is conducted using data from three actual offshore wind turbines which are located in the North sea of the coast of Belgium.

1.4 Component-level: fault detection

A wind turbine's blades are connected to a hub. The blades themselves can rotate (pitch) due to bearings. As wind passes, the blades spin around making the hub and subsequently the main shaft rotate. To support the rotation, a main bearing is placed after the hub and another bearing near the gearbox. The gearbox converts the low-speed rotation of the main shaft in high speed rotation fast enough to drive the generator efficiently. This entire drive train sits in the wind turbine's nacelle which is turned into the wind using a yaw motor. Furthermore, the drive train also contains a braking system to shut down the turbine in certain circumstances.

For wind turbines, drive trains and more specifically gearboxes are the main cause of the most down time [12]. A gearbox comprises two major components: gears and bearings. Most often, the bearings are the cause of the failure. Bearing failures are most often (80 %) caused by improper lubrication. Improper lubrication can consist of lubrication starvation, wrong lubricant and lubrication contamination. Other bearing faults are due to inadequate bearing selection (10 %), improper mounting (5 %), indirect failures (4 %) and manufacturing errors (1 %) [13]. Lubrication inadequacy leads to fault escalation, which eventually results in faults that are detectable using existing fault detection tools. Detecting lubricant inadequacy can help prevent fault escalation and hence is important to achieve preventive maintenance. Therefore, in this dissertation the focus is on conditions and faults related to the lubricant bearings.

Most often when the underlying physics of faults are completely understood,

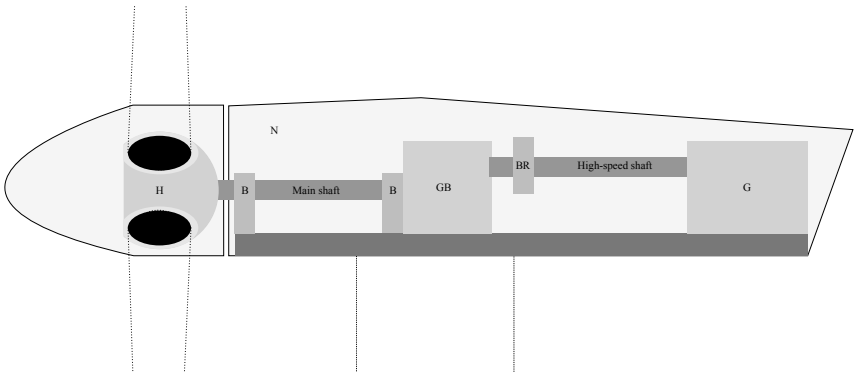


Figure 1.3: Wind turbine drive train. H = Hub; N = Nacelle; B = Bearing; BR = Brake; G = Generator; GB = Gearbox.

features from measurements can be engineered which are indicative of the respective faults or conditions. For bearings and also gear faults, most often vibration measurements are used [14]. Vibration measurements and the resulting physics-based features, are very useful to detect faults related to improper mounting or actual damages to bearing raceways [14]. However, localization of a fault is difficult due to the fact that vibrations propagate through the machine. Furthermore, vibration measurements are less suitable to detect lubricant related conditions.

Due to friction or churning of the lubricant in bearings, temperature distributions change. Hence, infrared thermal (IRT) imaging can help to identify bearing conditions and possibly locate them. Extracting useful features from IRT data based on physical models is difficult as modeling the thermodynamics of a bearing is challenging due to the complexity. One can for example resort to resistance methods or finite element analysis. Resistance methods require little computation time, however they lack accuracy. Finite element methods improve the accuracy, but are computationally cumbersome [15]. Hence, to use these methods to find appropriate features that will be observable using IRT imaging is difficult. In general, it can be stated that physics-based thermodynamic models are a powerful basis for fault detection, however, they require considerable knowledge, may lack the required accuracy or are possibly computationally burdensome.

Alternatively or additionally, data-driven approaches can be used. Data-driven approaches are designed to capture phenomena in the measurements without the requirement of having significant knowledge of the underlying physical interactions. Hence, these can possibly be applied to data for which current knowledge regarding fault detection is not complete and can therefore possibly help to understand the underlying physics. As is demonstrated in this dissertation on a lab-scale set-up, data-driven approaches have the potential to result in very well performing

systems that are also very fast.

In order to detect conditions and faults, features are mainly engineered by experts for specific conditions. Additional to, or as alternative to features derived from physical models, summary statistics such as the mean, standard deviation and kurtosis, –which are data-driven– are an option. However, even when adding summary statistics, feature engineering remains limited to the knowledge of the practitioner(s), which might result in suboptimal systems if the conditions and faults are not completely understood. Hence, in this dissertation also feature learning, additional to feature engineering, is researched for both vibration measurements and IRT data. Feature learning methods learn to create and extract features autonomously to be able to optimally distinguish between conditions. Such systems have been known to outperform systems that use engineered features [3].

When considering one modality at time, e.g. vibrations [16] or infrared imaging data [17], limited conditions and faults can be detected as certain conditions do not manifest themselves in certain modalities. Hence, multi-sensor systems are required. In this dissertation, IRT data and vibration measurements are combined to create a multi-sensor system. Both feature engineering and feature learning are investigated for this goal. In order to research these component-level approaches, a lab-scale set-up is used instead of a wind turbine's drivetrain. By using a lab-scale set-up, more controlled experiments can be done.

1.5 Fleet-level: dynamic dashboards

When the performance of a wind turbine and the conditions of its components can be monitored, a next step is to efficiently and effectively schedule maintenance actions. However, due to the fact that wind turbines are complex machines and contain many components, many signals have to be monitored. Even though data-driven based approaches help to monitor individual turbines and components, when considering an entire fleet, monitoring is still a daunting task because of the shear volume of the data that has to be inspected. To deal with the scale and complexity of an entire wind farm, intelligent dynamic dashboards are required.

Intelligent dynamic dashboards are researched and developed in this dissertation that not only can visualize raw real-time sensor data but also able to incorporate the extracted knowledge from both the performance monitoring systems and the component-level fault/condition detection systems. The core idea behind such a dashboard is to efficiently display data in such a way that it can enhance a user's insights into the data. To achieve this task, Semantic Web technologies are used to let the dashboard automatically combine sensors or knowledge extracted from raw data with visualizations. By intelligently combining the data and the visualizations, a user can monitor fleets more effectively and efficiently. Furthermore, data from new turbines or components can easily be added to the dashboard.

1.6 Main research contributions and outline

Within this dissertation, first steps are taken towards data-driven condition monitoring and preventive maintenance.

First, in Chapter 2, performance monitoring is improved using data-driven approaches. We illustrate that univariate performance monitoring can be improved by 26.1 % compared to the state-of-the-art approach using stochastic gradient boosted regression trees, which is a non-parametric data-driven machine learning method. Next, we show that by using additional input variables, performance monitoring can further improve (by 27.66 % compared to our univariate approach) as the models are capable of capturing the complex phenomena in the data.

Chapter 3 focuses on the detection of faults and conditions. Among the faults and conditions considered are lubrication inadequacy conditions, which are very difficult to detect using state-of-the-art approaches. The chapter starts with a feasibility study of the performance and usability of infrared thermal imaging for fault/condition detection in rotating machinery. In order to achieve this, two new features are proposed regarding the infrared thermal imaging domain. It is shown that it is indeed feasible to do automated fault and condition detection using infrared thermal imaging, as the proposed system can detect up to 88.25 % of the occurrences of the possible conditions and faults. Next, a vibration-based fault detection system is developed using existing expert features. This is done for two reasons. The first is to be able to compare the results to the infrared thermal imaging system, and the second is to create a multi-sensor system. The multi-sensor system combines two types of sensor data, i.e. infrared thermal imaging data and vibration measurements. It is shown that this multi-sensor system removes the weaknesses of the respective approaches and can detect conditions/faults which were previously hard to detect using state-of-the-art approaches (100 % detection rate).

Instead of designing the features for the modalities manually, Chapter 4 shows that features can effectively be learned from the data using convolutional neural networks. Convolutional neural networks are researched and designed in order to apply them on vibration measurements and infrared thermal imaging. Furthermore, in this chapter, it is illustrated that transfer learning for infrared thermal imaging-based fault detection can be done using convolution neural networks trained on natural images. Similar to Chapter 3, research is done into a multi-sensor system which outperforms single-sensor systems by 1.66 % compared to the infrared thermal imaging-based system and by 38.33 % compared to the vibration-based system. In the end, we also illustrate that from these convolution neural networks insights into the underlying physics can be extracted which can possibly help extend the knowledge about machine faults and how they occur in future research.

In Chapter 5, a dynamic dashboard is presented which can be used to monitor the vast number of signals originating from a wind farm. The dynamic dashboard can not only intelligently visualize raw sensor data, but also visualize the extracted knowledge from performance monitoring systems and fault/condition detection systems. The proposed dashboard engine uses Semantic Web technology to automatically connect sensor data to visualization widgets.

In Chapter 6, the dissertation is concluded with a brief discussion of the results and interesting future work.

Additional to the main chapters, four appendices are included in this dissertation. The first appendix, i.e. Appendix A, provides an overview of basic machine learning concepts and techniques required to understand this dissertation. This appendix can be consulted if the reader is not familiar with machine learning.

Appendix B, provides an overview of the impact of the different hyperparameter values for the machine learning models discussed in Chapter 2.

In Appendix C, an additional application is discussed on which infrared thermal imaging-based condition detection is applied. The systems proposed in Chapter 3 and 4, are evaluated for their general applicability on this additional application. It is shown that the methodologies proposed in these two chapters transfer well to a new use case.

In order to prove that imbalance gradations should be detectable using a thermal infrared camera, in Appendix D, the observable displacement by the camera is calculated for each imbalance gradation starting from the actual induced imbalance in the set-up.

Finally, in Appendix E, the properties of the new features proposed in Chapter 3 are proven and discussed.

1.7 Publications

The research results obtained during this PhD research were published in scientific journals and presented at international conferences. The following list provides an overview of the publications during this PhD research.

1.7.1 International journal publications

1. **Olivier Janssens**, Rik Van de Walle, Mia Loccufier and Sofie Van Hoecke. "Deep Learning for Infrared Thermal Image Based Machine Health Monitoring". *IEEE Transactions on Mechatronics*. Accepted for publication.

2. **Olivier Janssens**, Mia Loccufier, Rik Van de Walle, and Sofie Van Hoecke. "Data-Driven Imbalance and Hard Particle Detection in Rotating Machinery Using Infrared Thermal Imaging". *Infrared Physics & Technology*, 82: 28–39, 2017.

3. **Olivier Janssens**, Nymfa Noppe, Christof Devriendt, Rik Van de Walle, and Sofie Van Hoecke. "Data-driven multivariate power curve modeling of offshore wind turbines". *Engineering Applications of Artificial Intelligence*, 55:331–338, 2016.

4. **Olivier Janssens**, Viktor Slavkovikj, Bram Vervisch, Kurt Stockman, Mia Loccufier, Steven Verstockt, Rik Van de Walle, and Sofie Van Hoecke. "Convolutional neural network based fault detection for rotating machinery". *Journal of Sound and Vibration*, 377:331–345, 2016.

5. **Olivier Janssens**, Raiko Schulz, Viktor Slavkovikj, Kurt Stockman, Mia Loccufier, Rik Van de Walle, and Sofie Van Hoecke. "Thermal image based fault diagnosis for rotating machinery". *Infrared Physics & Technology*, 73:78–87, 2015.

1.7.2 Submitted international journal publications

1. **Olivier Janssens**, Mia Loccufier, Rik Van de Walle, and Sofie Van Hoecke. "ViTIS: Vibration and Thermal Imaging Based Multi-Sensor Fault Detection for Rotating Machinery". *ISA Transactions*. Under review.

2. **Olivier Janssens**, Mia Loccufier and Sofie Van Hoecke, "Dynamic Dashboards for Multi-Sensor Machine Monitoring Using Semantic Web Technologies". *Semantic Web Journal*. Under review.

3. Djairho Geuens, **Olivier Janssens** and Sofie Van Hoecke, "Non-intrusive Emotion Recognition using Computer Peripheral Input Analysis". *IEEE Transactions on Affective Computing*. Under review.

1.7.3 Book chapters

1. **Olivier Janssens**, Rik Van de Walle and Sofie Van Hoecke. "A learning based approach for real-time emotion classification of tweets". *in Applications of social media and social network analysis, Lecture Notes in Social Networks*, pages 125–142, 2015.

1.7.4 International conference publications

1. Gilles Vandewiele, Pieter Colpaert, Joachim Van Herwegen, **Olivier Janssens**, Ruben Verborgh, Erik Mannens, Femke Ongenae, and Filip De Turck. "Predicting train occupancies based on query logs and external data sources". *in Proc. of the 26th International World Wide Web Conference: 7th International Workshop on Location and the Web*, 2017.

2. Gilles Vandewiele, Kiani Lannoye, **Olivier Janssens**, Femke Ongenae, Filip De Turck, and Sofie Van Hoecke. "A genetic algorithm for interpretable model extraction from decision tree ensembles". *in Proc. of The Pacific-Asia Conference on Knowledge Discovery and Data Mining: Biologically Inspired Techniques for Data Mining*, 2017.

3. **Olivier Janssens**, Mathieu Rennuy, Steven Devos, Mia Loccufier, Rik Van de Walle, and Sofie Van Hoecke. "Towards intelligent lubrication control: Infrared thermal imaging for oil level prediction in bearings". *In Proc. IEEE Conference on Control Applications*, pages 1330–1335, 2016.

4. Gilles Vandewiele, **Olivier Janssens**, Femke Ongenae, Filip De Turck, and Sofie Van Hoecke. "GENESIM: genetic extraction of a single, interpretable model". *NIPS 2016: Workshop on Interpretable Machine Learning in Complex Systems*, 2016.

5. Ahmed Aldahdooh, Enrico Masala, **Olivier Janssens**, Glenn Van Wallendael, and Marcus Barkowsky. "Comparing simple video quality measures for loss-impaired video sequences on a large-scale database". *in Proc. Eighth International Conference on Quality of Multimedia Experience*, pages 1–6, 2016.

6. Sofie Van Hoecke, Koen Samyn, Gaetan Deglorie, **Olivier Janssens**, Peter Lambert, and Rik Van de Walle. "Enabling control of 3D visuals, scenarios and non-linear gameplay in serious game development through model-driven authoring". *in Lecture Notes of the Institute for Computer Sciences Social Informatics and Telecommunications Engineering*, pages 103–110, 2016.

7. **Olivier Janssens**, Lothar Verledens, Raiko Schulz, Veerle Ongenae, Kurt Stockman, Mia Loccufier, Rik Van de Walle and Sofie Van Hoecke. "Infrared and vibration based bearing fault detection using neural networks". *in Proc. 13th International Workshop on Advanced Infrared Technology and Applications*, pages 220–223, 2015.

8. Sofie Van Hoecke, **Olivier Janssens**, Raiko Schulz, Kurt Stockman, Mia Loccufier, and Rik Van de Walle. "Towards thermal imaging based condition monitoring in offshore wind turbines". *in Proc. of 13th International Workshop on Advanced Infrared Technology and Applications*, pages 291–295, 2015.

9. Sofie Van Hoecke, Cynric Huys, **Olivier Janssens**, Ruben Verborgh, and Rik Van de Walle. "Dynamic monitoring dashboards through composition of web and visualisation services". *in 2nd EAI International Conference*

on Software Defined Wireless Networks and Cognitive Technologies for IoT, pages 1–10, 2015.

10. **Olivier Janssens**, Steven Verstockt, Erik Mannens, Sofie Van Hoecke, and Rik Van de Walle. "Influence of weak labels for emotion recognition of tweets". *in Proc. Mining Intelligence and Knowledge Exploration*, pages 108–118, 2014.

11. **Olivier Janssens**, Koen Samyn, Rik Van de Walle, and Sofie Van Hoecke. "Educational virtual game scenario generation for serious games". *in Proc. Serious Games and Applications for Health*, pages 1-8, 2014.

12. Steven Verstockt, Viktor Slavkovikj, Pieterjan De Potter, **Olivier Janssens**, Jurgen Slowack, and Rik Van de Walle. "Automatic geographic enrichment by multi-modal bike sensing". *in Communications in Computer and Information Science. In Communications in Computer and Information Science*, 456, pages 369–384, 2014.

13. **Olivier Janssens**, Maarten Slembrouck, Steven Verstockt, Sofie Van Hoecke, and Rik Van de Walle. "Real-time emotion classification of tweets". *in Proc. IEEE/ACM International conference on advances in social networks analysis and mining*, pages 1430–1431, 2013.

14. **Olivier Janssens**, Jonas De Vylder, Jan Aelterman, Steven Verstockt, Wilfried Philips, Dominique Van Der Straeten, Sofie Van Hoecke, and Rik Van de Walle. "Leaf segmentation and parallel phenotyping for the analysis of gene networks in plants". *in Proc. 21st European signal processing conference*, 2013.

15. Steven Verstockt, Viktor Slavkovikj, **Olivier Janssens**, Pieterjan De Potter, Jurgen Slowack, and Rik Van de Walle. "Web-based enrichment of bike sensor data for automatic geoannotation". *in Proc. GEOCROWD*, 2013.

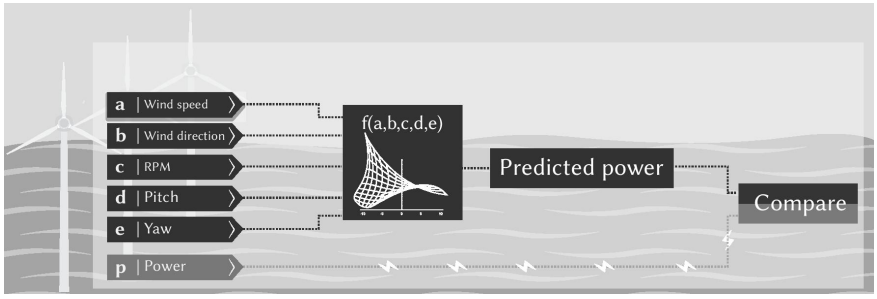
16. Steven Verstockt, **Olivier Janssens**, Sofie Van Hoecke ,and Rik Van de Walle. "Spatio-temporal video retrieval by animated sketching". *in Proc. International Conference om Computer Vision Theory and Applications*, pages 723–728, 2013.

17. **Olivier Janssens**, Steven Verstockt, Pieterjan De Potter, Piet Verhoeve, and Rik Van de Walle. "Ki-Touch: a Kinect-based virtual touchscreen". *in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems: Workshop on Color-Depth Camera Fusion in Robotics*, 2012.

Chapter 2

Machine level: Data-driven multi-sensor performance monitoring

Chapter highlights



This chapter focuses on improving performance monitoring of offshore wind turbines. We show that univariate power prediction can be improved up to 26.1 % compared to the state-of-the-art by applying stochastic gradient boosted regression trees. Moreover, we further improve this power prediction approach by incorporating additional input variables into the model, resulting in an additional improvement of 27.66 %.

2.1 Introduction

A wind turbine is engineered with the purpose of generating electrical energy using the energy in the wind. Hence, a wind turbine's most important property is its efficiency, i.e. how well it converts wind energy into electrical energy. This efficiency is formulated as a power curve that shows the relationship between the wind speed and the generated power. The follow up of a wind turbine's power curve is referred to as performance monitoring.

Performance monitoring requires three steps. The first is the acquisition of data from a wind turbine. The data consists of physical properties such as the wind speed and power output. Second, using this data, a power curve, i.e. model, is built which is used to predict the output power of the wind turbine. Third, the predictions given by the model are compared to the actual generated power output. When the wind turbine is healthy, the predicted values should be similar or nearly similar to the actual measurements. Hence, an operational limit can be placed on the difference between the predicted output and actual power output. If the operational limits are crossed, there is unexpected behavior, i.e. an anomaly, which can be further investigated.

To create such a power curve model, two approaches are possible, namely physics-based approaches, which stem from the underlying physics, and data-driven approaches, which only use collected measurement data. Within this chapter the focus lies on data-driven models that are designed to predict a wind turbine's output power as these data-driven methods are very powerful, do not require considerable expertise knowledge on wind turbine dynamics and they can provide very good results quickly.

In the next section (2.2), related literature is discussed. First, information is provided how physics-based models are created and used, and why they are less preferable compared to data-driven methods. Next, data-driven approaches are discussed together with the state-of-the-art approaches. Based on the review of the related literature two opportunities are identified. The first is that performance of state-of-the-art approaches only using wind speed as input variable have limited performance and hence have room for improvement. The second opportunity lies in using more than one variable as input. The contributions in this chapter therefore consist of improved data-driven univariate power prediction approaches as well as data-driven multivariate power prediction approaches. The multivariate approach uses wind direction, turbine yaw, blade pitch and rotations per minute of the rotor on top of the wind speed as variables. The proposed approaches are discussed in Section 2.3 and are first evaluated on a synthetic data set. Next, the approaches are evaluated on a data set collected from three actual offshore wind turbines which are located in the North Sea of the coast of Belgium. A description of the data sets is given in Section 2.4. Sections 2.5 and 2.6 present the achieved results

and illustrate that by using data-driven techniques insights into the data can be acquired by looking at how important the input variables are for the model to accurately predict the output power. Finally a discussion is given in Section 2.7 and a conclusion in Section 2.8.

2.2 Related literature

In this section background information together with related literature is provided on both physics-based approaches for performance monitoring and data-driven approaches

2.2.1 Physics-based machine-level performance monitoring

A "physics-based model" is defined as a set of mathematical functions wherein the relationship between various parameters is modeled explicitly by a domain expert.

The basis of a power curve model is the relationship between the wind speed and generated power, as can be seen in Equation 2.1, wherein A denotes the area the wind turbine's blades cover, v the wind speed, ρ the air density and C_p the rotor efficiency, which indicates how much of the energy in the wind is converted into electrical energy.

$$P_w = \frac{1}{2}\rho A v^3 C_p \tag{2.1}$$

It was determined that the rotor's efficiency could maximally be 59.3 % [18]. This is known as Betz' Law. However, in reality less power is extracted from the wind as a wind turbine is not operational at all wind speeds. At very low wind speeds insufficient torque is exerted by the wind on the turbine blades to make them rotate. When the wind speeds are greater than the *cut-in speed* the turbine will start to rotate and generate electrical power. This power generation will be according to Equation 2.1. However, at high wind speeds, the output power will reach the limit of the electrical generator. This limit is called the *rated output power* and is reached at the *rated output speed*. When wind speeds higher than the rated output speed occur, the wind turbine's controller will limit the power generation, for example by changing the angles of the blades, resulting in a constant level of output power. If the wind speed increases further, the wind turbine can suffer from damage. To prevent damage, a braking system is activated by the controller to bring the rotor to a standstill. The braking is activated at the *cut-out speed*. In Figure 2.1 a power curve is shown including the terms mentioned above.

State-of-the-art physical models are often designed to extend the model described in Equation 2.1. Extentions are made by creating additional subsystem

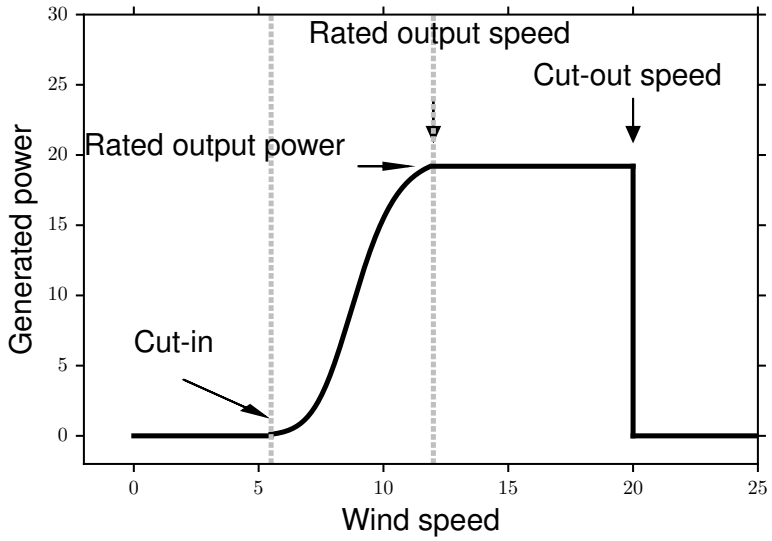


Figure 2.1: Example of a wind turbine power curve. The minimum speed at which a wind turbine produces power is known as the cut-in speed. The rated output power is the maximum output power and the cut-out speed is the maximum operational speed.

models to take energy losses into account due to sub-components' efficiency and external influences. For each of these losses, separate models need to be designed by domain experts. For example, a model for the rotor, generator, speed controller, pitch angle controller, protection system and electrical transmission can be created [19–22]. External influences, such as turbulent air and shading effects also have an influence on the wind turbine's performance. Attempts have been made to take these effects into account for power prediction [23–25]. However, these approaches are still very much sub-optimal as it is, for example, difficult to predict the power produced when considering a row of wind turbines [26].

Another physics-based approach is the usage of computational fluid dynamics (CFD). CFD can be used to model wake effects in a wind farm [27, 28]. These models are most often used to study the wake effects, but when using operational data they can also be used to predict the power output of individual turbines which result in better performing models than when using analytical models as described above [29]. CFD also has been used to model wind turbines that suffer from icing, i.e. ice that forms on the blades, for which the results can be used to develop a possible ice detection system [30]. Limited studies have also been carried out to simulate certain faults such as damaged blades in wind turbines [31] and can then possibly be used to detect damaged blades of real wind turbines, which however, has not been done yet.

Physics-based approaches are white-box approaches as they enable an operator/practitioner to better understand how the performance of a wind turbine is influenced as these approaches explicitly model the interaction between the different variables. However, they can require considerable expert knowledge, and for CFD-based approaches, considerable computational power is required to be practically applicable.

2.2.2 Data-driven machine-level performance monitoring

A "data-driven model" is defined as a model wherein the relationships between the various parameters are not explicitly modeled by a domain expert, but rather constructed from the data using statistics or machine learning. As these models are not inherently interpretable, they are often called black-box approaches.

Monitoring the performance of a machine or process by only considering data has been done for a long time using statistical process control (SPC) [33]. SPC is done by defining what the normal operational conditions of a machine are and subsequently define control limits for its signals. Such a technique is easy to implement, and hence is often a first step in the performance monitoring process. By monitoring the variability of signals, a good estimate of how the machine is performing can be acquired. When labeled data is missing, i.e. data annotated with the machine's condition, SPC is the only approach that can be taken for data-driven performance monitoring as it does not require labeled data. If labeled data is available, a classification approach can be used enabling fault diagnosis. However, as many possible faults can occur, lots of annotated data should be available to detect the specific condition, which is often not feasible.

The concept of statistical process control is closely related to anomaly detection because data points outside the control limits are considered to be anomalies. However, generally speaking, statistical process control is done on one signal at a time (univariate) such as the generated power, hence it is impossible to take multivariate phenomena into account. To solve this problem, dimensionality reduction techniques can be used such as Principal Component Analysis (PCA) [34]. PCA is an unsupervised technique that reduces the dimensionality of many correlated signals. PCA is used to construct subspaces with a lower dimensionality than the original data. This is done offline and using data from normal operational conditions. When performing online monitoring the data is projected into these subspaces and using test statistics, such as the squared prediction error or T^2 statistic, abnormal operation conditions can be detected [35]. Many variations of this approach have been proposed. However, one of the main drawbacks of these approaches is that non-linear phenomena cannot be captured, such as in a power curve, as these are techniques that are based on linear combinations of the input variables.

Another anomaly detection approach is the regression-based approach. Regression is used to predict a continuous output. In the performance monitoring case, several input parameters are used and a value that is indicative of the machine's performance is predicted such as the generated power. As with the dimensionality reduction approach, only data from normal operational conditions is used to train the model. Subsequently, the model can be used to predict what the output value should be, given the input parameters. This predicted value can be compared to the actual measured value. Using anomaly detection techniques or SPC rules, one can then detect if the actual value deviates too much from the predicted value. The advantage of such an approach is that it can deal with multivariate problems and non-linearity if an appropriate algorithm is chosen.

In recent years, data-driven (regression-based) modeling methods have gained attention for performance monitoring of wind turbines. Often the research effort focuses on improving **univariate modeling** (i.e. model the relationship between the wind speed and the generated power). For example, Kusiak et al. [36] compare two parametric models with five non-parametric models on separate data sets. The two parametric models used are both logistic curves with four parameters controlling the shape of the curve which describes the relation between the wind speed and power output. These four-parameter logistic curves are either optimized using maximum likelihood or the method least-squares. The non-parametric models are a multi-layer perceptron, a random forest, a M5P decision tree, a boosting tree and a K-nearest neighbors (KNN) regression model. Of the non-parametric models, the KNN model achieves the best results. From the parametric models, the four parametric logistic curve optimized with least squares performs the best.

A recently published comprehensive review on power curve modeling by Lydia et al. [37] also presents a comparative literature study on parametric and non-parametric methods. Methods reviewed are, among others, linearized segmented model, polynomial curve fitting, both four and five parameter logistic curves fitted using several different optimization algorithms, neural networks (NN) and fuzzy methods. When comparing the two reviews it can be concluded that by fitting a five-parameter logistic (5PL) curve using differential evolution (DE) the best results are achieved. This was shown using synthetically generated data set and also a data set containing data from onshore wind turbines [38].

It was recently shown that other possible combinations of parameters, which are measured and stored by the supervisory control and data acquisition (SCADA) system, can be used for performance monitoring of a wind turbine. In [39] the rotor curve (mapping between rotor speed and wind speed) and blade pitch curve (mapping between the turbine's pitch and wind speed) are used for more precise anomaly detection. In the work of Schlechtingen et al. [40, 41] more than just

the wind speed is used as input of data-driven models (**multivariate modeling**). Methods such as cluster center fuzzy logic, NN, KNN and adaptive neuro-fuzzy interference system models are compared for performance monitoring [40, 41]. The results indicate that when augmenting the wind-speed based models by using the ambient temperature and wind direction as additional inputs, the variance in the generated power is better accounted for. Because of this, better prediction results are achieved and more accurate anomaly detection is possible.

The related literature and specifically the review by Lydia et al. [37] indicate that considerable research has been done regarding univariate power curve modeling. However, as illustrated in [40], multivariate power curve modeling can improve performance monitoring. There are several scenarios one can think of where additional parameters will be useful. For example, the anemometer might measure high wind speeds, which in a simple model will result in high output power. However, the turbine is in fact generating no energy because of too high wind speeds (larger than the cut-out speed). Using a simple model this would be flagged as an anomaly. However, by incorporating the rotation speed of the rotor into the model, the model will be able to handle this scenario and still predict well. Hence, the event will not be flagged as an anomaly.

In the next section information about different machine learning algorithms is provided which we compare for their prediction efficacy regarding performance monitoring. The goal is to see if the state-of-the-art approach can be surpassed.

2.3 Data-driven performance modeling

Within this section, six approaches for performance modeling are employed which are selected based on the related literature.

Five parameter logistic curve fitted by differential evolution (5PL-DE):
The first approach is the state-of-the-art approach discussed in Section 2.2.2. This approach is a parametric approach as it fits a five parameter logistic curve to wind speed and power production data using the differential evolution optimization algorithm [37, 38]. This approach only allows univariate modeling and not multivariate modeling, i.e. only the wind speed can be used as input variable of the model.

In optimization procedures for parametric models, the parameters of an explicit mathematical function are tuned, resulting in a function that fits the data as well as possible. For the power curve, several functions have been proposed and fitted to wind speed and power data, such as the logistic curve with $1, .., 5$ parameters. As Figure 2.1 illustrates, it is clear that a power curve is not symmetric. Therefore, a

five parametric logistic function is the most suitable choice since it is an asymmetric s shaped curve. The functional form of the 5PL can be seen in Equation 2.2, where $\boldsymbol{\theta} = (a, b, c, d, g)$, for which the range of parameters is restricted to: $c > 0$ and $g > 0$ and x is the input parameter, i.e. wind speed. The parameters and their related effects are listed in Table 2.1. The effects of varying the parameters of a 5PL function can be seen in Figure 2.2.

$$f(x, \boldsymbol{\theta}) = d + \frac{(a-d)}{(1 + (\frac{x}{c})^b)^g} \tag{2.2}$$

Parameters	**Usage**
a	Horizontal lower asymptote
b	Steepness of the curve
c	Point on the curve where the curvature changes direction
d	Horizontal upper asymptote
g	Controls the asymmetry

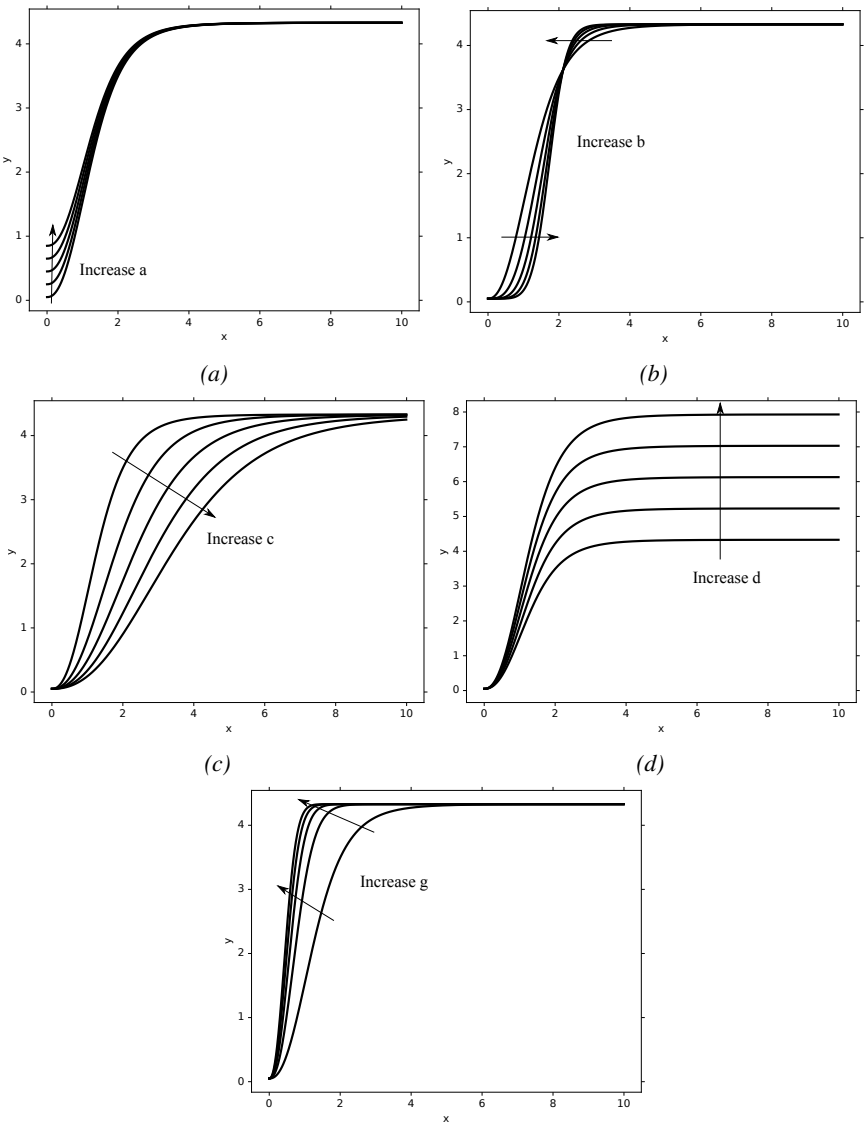
Table 2.1: Description of the parameters in the five-parameter logistic curve [42].

The objective function, optimized to fit the 5PL can be seen in Equation 2.3.. k is the amount of samples in the data set, **y** is the measured power, and $\boldsymbol{\theta}$ is the set of parameters which need to be estimated.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{k} [f(x_i, \boldsymbol{\theta}) - y_i]^2 \tag{2.3}$$

In order to find the best values for the 5 parameters of the logistic function, differential evolution (DE) is used to optimize Equation 2.3. Differential evolution tests and changes various values for a,b,c,d and g in an efficient and effective manner with the goal of making the logistic curve match the data as well as possible. Similar to Lydia et al. [37], we use the DE/best/1/bin scheme consisting of the following steps:

- **Step 1. Create population**: The population is a set of *individuals* (an individual is a vector which represents a set of parameters for Equation 2.2). The population, described by Equation 2.4 is a nxm matrix, where n is the amount of parameters per individual and m is the number of individuals in the population. The number of individuals is a parameters which has to be set by the practitioner. It should be noted that if this number is too large, the algorithm will take a long time to finish. If this number is too small, the algorithm will not be able to fit the curve well to the data.



(a) *(b)* *(c)* *(d)* *(e)*

Figure 2.2: Effects of varying the a, b, c, d and g parameters, respectively of a 5PL function.

$$\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_j, ..., \boldsymbol{\theta}_m] \tag{2.4}$$

With $\boldsymbol{\theta}_j$ as in Equation 2.5.

$$\boldsymbol{\theta}_j = [\theta_{1j}, \theta_{2j}, ..., \theta_{nj}]^{\mathrm{T}} \tag{2.5}$$

Note that n is equal to 5 for the 5PL.

- **Step 2. Evaluate the population**: In this step each individual in the population is evaluated according to the objective function described in Equation 2.6, wherein x_i represents the wind speed, and k the number of samples in the training set. The best individual from the population $\boldsymbol{\Theta}$ is the one for which the value of the objective function is the smallest. This individual is then defined as $\boldsymbol{\theta}_{best}$.

$$g(\boldsymbol{\theta}_j) = \sum_{i=1}^{k} [f(x_i, \boldsymbol{\theta}_j) - y_i)]^2 \tag{2.6}$$

- **Step 3. Mutation**: A set of donor vectors **V**, i.e a nxm matrix is created in this step according to Equation 2.7.

$$\mathbf{v}_j = \boldsymbol{\theta}_{best} + \gamma(\boldsymbol{\theta}_I - \boldsymbol{\theta}_{II}) \tag{2.7}$$

Where $\mathbf{v}_j$ is the $j-th$ donor vector. $\boldsymbol{\theta}_I$ and $\boldsymbol{\theta}_{II}$ are two mutually exclusive vectors chosen from the population at random. This difference is scaled by the scalar γ, which lies in the interval [0.4,1].

- **Step 4. Crossover**: Trial vectors (**U**) are generated based on the donor vectors and the population. The crossover is applied to each pair of individuals $\boldsymbol{\theta}_j$ and its donor vector $\mathbf{v}_j$, this results in a trial vector as described in Equation 2.8.

$$u_{ij} = \begin{cases} v_{ij} & \text{if } \beta \leq \epsilon \text{ or } j = j_{rand}; \\ \theta_{ij} & otherwise; \end{cases} \tag{2.8}$$

In this equation, β is a random number in the range of [0,1], ϵ is the crossover rate and is set manually, j_{rand} is a random number in the range of [1,m]. If the cross-over rate is set to a large number, more elements of the donor vectors will end up in the trial vectors.

- **Step 5. Selection**: Members for the new generation are created according to Equation 2.9.

$$\boldsymbol{\theta}_j = \begin{cases} \mathbf{u}_j & \text{if } g(\mathbf{u}_j) \leq g(\boldsymbol{\theta}_j); \\ \boldsymbol{\theta}_j & otherwise; \end{cases} \tag{2.9}$$

- **Step 6. Termination**: A fixed number (i.e. generations) of iterations is run through starting from step 2. In the end, the individual resulting in the smallest error given by the objective function is the optimum parameter set.

The hyperparameters (see Section A.3.1), such as the crossover rate, number of generation and the size of the population are optimized using grid-search (see Section A.3.5).

K-nearest neighbors regression (KNN):
The second approach is K-nearest neighbors regression, a non-parametric instance based regression algorithm [1], as it is the best non-parametric method regarding power curve modeling described in [36]. Furthermore, KNN can be used for both univariate modeling as multivariate modeling. Instead of learning the underlying pattern of the data, the algorithm will compare new samples to instances which are saved in the training phase. For more information on KNN we refer the reader to Section A.4.1 in the preliminaries chapter.

Method of bins (MOB):
The third approach is the method of bins, which is included because it is the standard approach set by the International Electrotechnical Commission [43] to model a power curve. The goal of this method is to create a power curve which can be compared to the manufacturer's power curve to see if the installed turbine is working as expected. The MOB is a method which only allows univariate modeling and not multivariate modeling. This means that only the wind speed can be used as input variable. The determination of a power curve is done by applying the MOB on the normalized data set. Every bin has to span 0.5 m/s for which the mean of the normalized wind speed and power output have to be calculated. Based on these bins and corresponding mean values it is then possible to predict the power output given the wind speed.

Random forest regression (RF):
As fourth approach, random forest regression is employed since it is an ensemble method, which are known to outperform individual models [44]. Random forests are easy to use and have few hyperparameters that have to be tuned. More information on random forest regression can be found in Section A.4.3.

Extremely randomized trees (ERT)

The fifth approach entails a variation on random forest regression called extremely randomized trees regression which is known to outperform random forests [45]. Compared to random forests it sometimes reduces the variance even more [45]. More information on extremely randomized trees can be seen in Section A.4.4.

Stochastic gradient boosted regression trees (SGBRT):

The last method is stochastic gradient boosted regression trees which is also an ensemble method based on regression trees which performs very well [46].

In the RF and ERT methods, the individual regression trees are independent of each other and can therefore be constructed at the same time, i.e. in parallel. Alternatively, in the SGBRT method the individual trees are built in a forward stage-wise procedure. The algorithm starts off by learning a single regression tree model on the training set. Gradient boosting improves this initial model by sequentially constructing additional models that are trained on the error of the previous model. More information about gradient boosted regression trees is provided in Section A.4.5.

A RF consists of many complex models, i.e. deep regression trees, which have high variance and low bias, resulting in an ensemble model with an unchanged bias but low variance. Bias measures how far off in general a model's predictions are from the correct value. Conversely, boosting uses small models which have low variance (i.e. underfitted), but high bias. By combining these models in this stage-wise procedure the bias is reduced, resulting in a well performing ensemble method. An example of the bias-variance trade-off for RFs and SGBRTs can be seen in Figure 2.3. In this figure a function is approximated by a RF (Figure 2.3a) and by SGBRT (Figure 2.3b). The red lines indicate the predictions of the individual trees and the dashed line the respective predictions of the ensemble models. As can be seen for the RF, individual models are complex and have a high variance. Nevertheless, when they are averaged, the variance is reduced (i.e. the black, dashed line's fluctuations are more narrow). For SGBRT, it can be seen that the individual trees are simple and have a high bias and low variance as they do not follow the function to approximate and do not fluctuate a lot. However, the final ensemble model has a low bias and a relatively low variance and therefore approximates the function well.

These tree based approaches are not only chosen because they are known to perform very well, but also because they can provide insights into the data. In machine learning, input features are almost never equally important, often only a few features will substantially influence the output. Using tree-based approaches the relative importance of features can be calculated. Calculating the importance of the features is done as follows: when traversing a tree, top to bottom, at every node the error reduction by that node is computed. This error reduction is subse-

quently multiplied by the amount of samples that are routed to the node. For every feature these quantities are added together and then normalized, giving the feature importance.

2.4 Data sets

Data from actual wind turbines is difficult to acquire. This is due to the fact that from the data knowledge can be gained about the wind turbine by reverse engineering. Hence, intellectual property can be stolen. This is why Lydia et al. [38] compared different modeling methods on both actual and synthetic data so that the results from the synthetic data can serve as a reference for other researchers. Therefore, the six algorithms described above are evaluated on a synthetic data set generated according to the procedure described in [38]. Unfortunately, this synthetically generated data set only contains two parameters, i.e. the wind speed and the generated power. Hence, to evaluate the algorithms for multivariate modeling, a data set was created containing data from three actual offshore wind turbines.

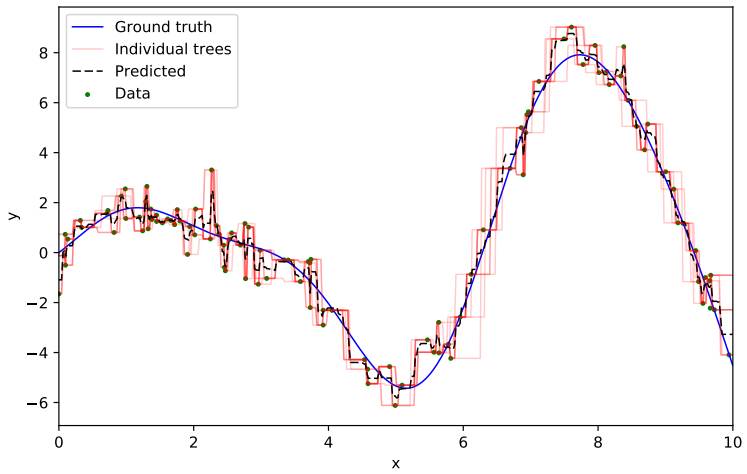
2.4.1 Synthetic data

A synthetic data set is created using Equation 2.10, where the output power is denoted as p_a, the wind speed as v, the rated power as p_r, the cut-in speed by v_c, the rated speed by v_r and the cut-out speed by v_s. c is referred to as the power curve coefficient and is calculated according to Equation 2.11 where ρ is the air density, R the radius of the rotor determining its swept area, and c_p is the rotor efficiency coefficient. As in [38], these parameters are set as follows: c = 0.03906 kg/m, p_r = 20 kW, v_c = 2 m/s, v_r = 8 m/s, v_s = 18 m/s and σ, the standard deviation for the added noise, is set to 0.5 kW.

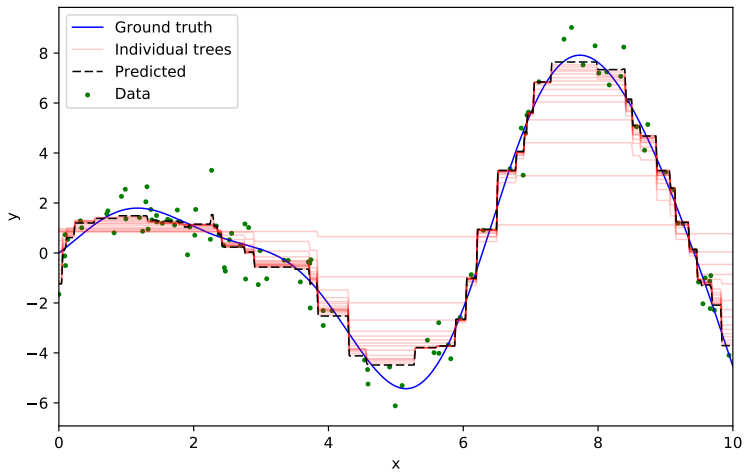
$$p_a(v,\sigma) = \begin{cases} 0, & v < v_c, v > v_s \\ cv^3 + N(0,\sigma^2) & v_c \leq v \leq v_r \\ p_r & v_r \leq v \leq v_s \end{cases} \tag{2.10}$$

$$c = \frac{1}{2} c_p \rho \pi R^2 \tag{2.11}$$

From article [38], it can not be deduced what the range of input wind speeds was, or if the inputs were generated in a specific way. In order to get a series of realistic wind speeds, the method described in [47] is used in this research to generate wind speeds. This method uses a one-step Markov chain model, introducing dependency between the wind speed at time step t and $t-1$. The wind speeds are



(a)



(b)

Figure 2.3: In (a) the predictions of individual regression trees of a random forest are plotted together with the mean value. In (b), the individual simple regression trees are plotted together with the eventual prediction.

generated in the range of [0,v_s]. Our generated data set is made available online for future comparisons[1].

2.4.2 Real world data sets

The real data used in this research come from three wind turbines located in the North Sea, 46 km off the Belgian Coast. The three data sets were gathered –using the SCADA system– in the same time period and are regarded as only containing normal operational conditions. Each data set contains three months worth of data. More specifically, for each one of the three wind turbines 13248 10-minute average values of the rotations per minute (RPM) of the rotor, wind direction, the wind turbine's yaw, the pitch of the blades, generated power and wind speed were collected. The wind speed is normalized following the IEC standard [43]. The normalization procedure uses the absolute air temperature and air pressure, therefore it can be stated that these parameters are indirectly accounted for in the models. As only SCADA data is used, no additional sensors have to be installed in the turbine, which makes performance monitoring an easy to use step in the condition monitoring process.

Each data set is split into a training set and test set of respectively 10800 and 2448 measurements. Data preprocessing is kept to a minimum so that only entries with missing values are removed and only measurements in the operational range of the wind turbine are kept i.e., [3 m/s; 25 m/s]. Additionally, for training and testing purposes, the individual variables are scaled between 0 and 1, but for error measurement calculations, they are rescaled back to their original ranges. This is done because training and testing on rescaled data improves the results slightly. It should be noted that this scaling is required for algorithms such as KNN regression.

2.4.3 Testing procedure

Since every modeling technique has several hyperparameters which needs to be determined, such as the number of trees in the random forest method or the learning rate in the SGBRT method, a grid-search procedure using 10-fold cross-validation on the training set is used. 10-fold cross-validation divides the original training set in ten subsets. One subset is used as validation set and the nine other subsets are combined and used to train the model with the specific hyperparameters. This is done ten times to make sure that every subset is used once for validation. It is very important to note that the test set is not used to determine the optimal hyperparameters to ensure generalization.

Afterwards, the optimal hyperparameters are used in the algorithms during the training phase while utilizing the entire training set in order to create the models.

[1] https://data.mendeley.com/datasets/gst3cdfnn5/1

Afterwards, the models are tested on the test set. In order to judge the performance, two metrics are calculated: mean absolute error (MAE) and the root mean squared error (RMSE).

2.5 Univariate performance modeling

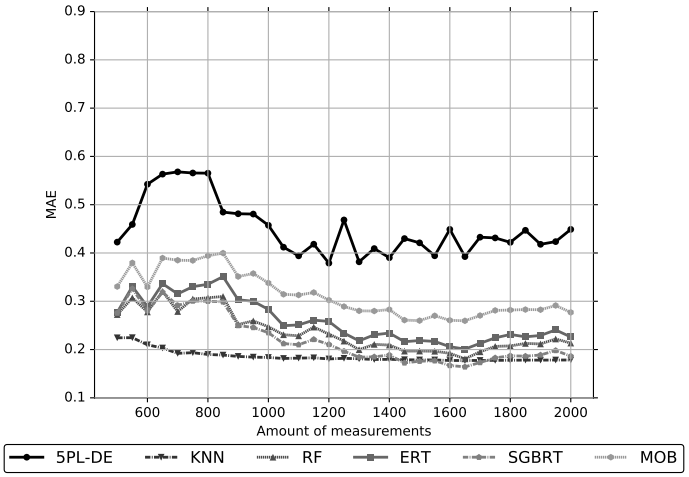
The state-of-the-art results for performance monitoring, as described in the literature review (Section 2.2.2), are achieved using an univariate method, i.e. solely the wind speed is used to predict the generated power. Therefore, within this section, the six algorithms only use the wind speed as input. First, a test is conducted using the synthetic data set and then using the data set from the actual offshore wind turbines.

2.5.1 Results for the synthetic data set

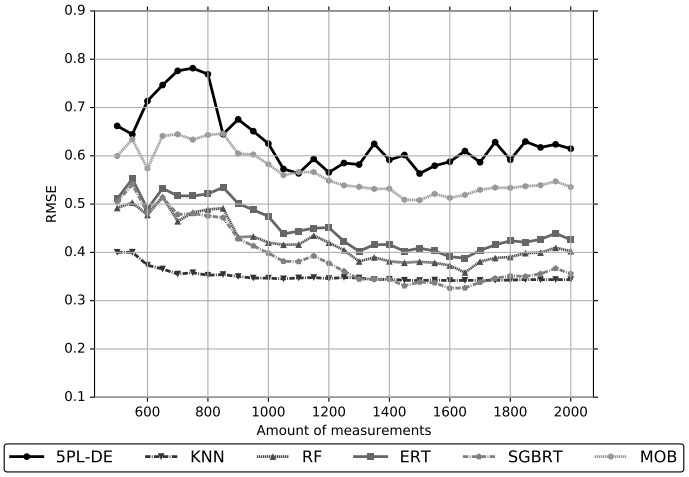
Lydia et al. [38] tested their methods using 1008 synthetically generated values. In order to make the comparison, in this research all six models are trained and tested on different sizes of data sets. The initial data set size is set to 500 values and step-wise incremented by 50 values until the data set consists of 2000 values. For the six models the MAE and RMSE are calculated and presented in Figure 2.4.

In [38], the scores for MAE and RMSE for the 5PL optimized by DE are respectively 0.4874 and 0.6408. The MAE and RMSE achieved here for 5PL optimized by DE for 1000 data points is 0.4573 and 0.6252. These values are similar to those reported in [38], hence these results confirm that the implemented method works correctly.

The graphs illustrate that by adding more data, most of the error rates get lower until a certain point. Generally, when compared with other methods, the 5PL fitted by DE will achieve relatively worse results. The best result in the end is achieved by the KNN method (MAE: 0.1783; RMSE: 0.3435). Nevertheless, the SGBRT method surpasses the KNN's results briefly when 1650 data points are used (MAE: 0.1641; RMSE: 0.3257).



(a) MAE of the different models on the synthetic data set.



(b) RMSE of the different models on the synthetic data set.

Figure 2.4: Six models evaluated on different sizes of the synthetic data set.

2.5.2 Results for the real world data set

Regarding the real world data set, the MAE and RMSE for the univariate power curve modeling can respectively be seen in Figure 2.5a and 2.5b. The results are presented in a stacked bar diagram such that the overall results of the algorithm, regardless of the wind turbine, can be observed as well as the results of the algorithms for the individual turbines.

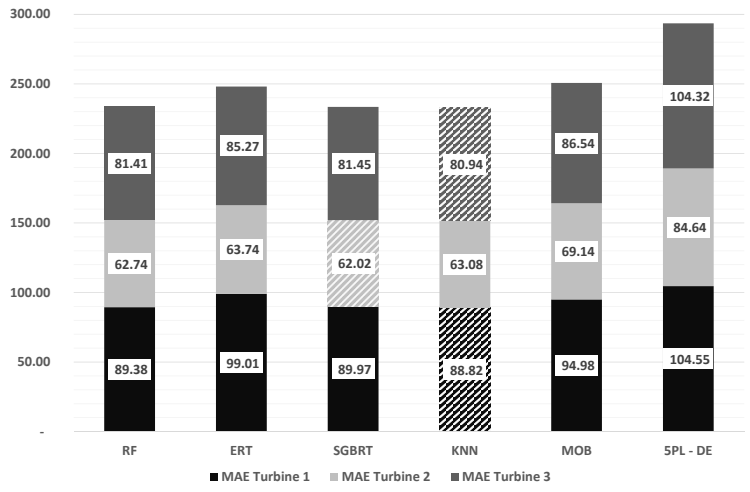
The results indicate the following:

- The non-parametric data-driven models, i.e. RF, ERT, SGBRT, KNN and MOB perform very similarly. The average MAE for these models is 239.69 with a standard deviation of 8.85. This low standard deviation indicates that the MAE of these five models is very similar.
- Contrary to what M. Lydia concludes [38], the 5PL curve optimized by DE performs noticeably worse. Compared to the average MAE of the other models, the 5PL's MAE is 55.81 larger. Considering that the standard deviation of the MAEs of the other models is 8.85, it can also be concluded that the 5PL performs significantly worse. This is possibly due to the fact that this approach is limited by its explicit parametric design.
- When considering the MAE for the individual wind turbines, it can be seen that for turbine one and three, KNN provides the best results (88.82 and 80.94 respectively) and for turbine two, SGBRT (62.02).
- The RMSE indicates that for turbine three the KNN method provides the best results (140.00) and for turbine one and two the SGBRT method (150.98 and 99.97 respectively). The results achieved by KNN are to be expected as the literature review in Section 2.2.2 indicates that KNN is an algorithm that performs well on this task. Nevertheless, as this is still a fairly easy task, other methods such as SGBRT are able to achieve similar or better results.

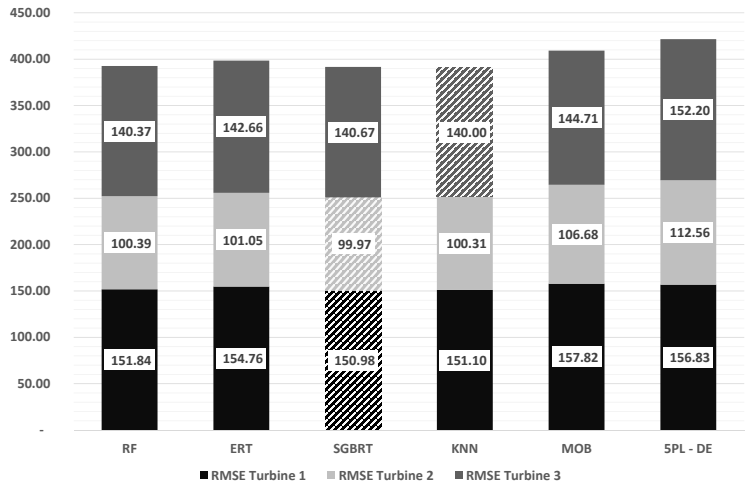
The hyperparameters for the models for the different experiments are listed in Table 2.2. From this table, it can be seen that the values of hyperparameters can be very diverse such as the number of trees. However, as is discussed in the next section, from experiments it can be determined that the number of trees has a small impact on the results.

As this section demonstrates, improvements compared to the state-of-the-art, i.e. 5PL optimized using DE, are possible. On average, the RF, ERT, SGBRT, KNN and MOB achieve an improvement of 18.89 % (σ = 2.99 %) compared to the 5PL approach when considering the MAE. When considering the RMSE, this improvement is 5.91 % (σ = 1.79 %).

Nevertheless, the experiments also illustrate that limited improvement can be achieved if only information from one variable (the wind speed) is incorporated



(a)



(b)

Figure 2.5: Stacked bar charts of the (a) MAE and (b) RMSE of the predictions for the power output of the three wind turbines by different univariate approaches. Shaded bars indicate the best score for a turbine.

Algorithm	Parameters	Value turbine 1	Value turbine 2	Value turbine 3
5PL-DE	Differencing scaling factor	1	1	1
	Crossover rate	0.5	0.5	0.5
	Population size	50	50	50
	Generations	100	100	100
KNN	K	200	100	100
RF	Number of trees	100	300	10
ERT	Number of trees	400	10	10
SGBRT	Loss	Least squares	least absolute deviation	Least squares
	Learning rate	0.01	0.01	0.1
	Number of trees	500	800	100
	Subsample ratio	0.3	0.1	0.1

Table 2.2: Hyperparameters for the univariate data-driven performance models for the three offshore wind turbines.

in the models to predict the generated power. In the next section additional variables, which possibly encapsulate additional useful information, are provided to the algorithms.

2.6 Multivariate power curve modeling

The results, when the wind speed, yaw, pitch, RPM and wind direction are incorporated into the models can be seen in Figure 2.6a and 2.6b. It should be noted that as the MOB and the 5PL-DE do not work with more than one input variable, they are not added to the comparison. From these results the following observations can be made:

- When incorporating these additional variables into the model, the average MAE is reduced by 14.61 % and the average RMSE by 12.65 %.
- Regarding the MAE, SGBRT gives the best results across all turbines (65.08; 58.92 and 70.54 respectively for turbine one, two, three)
- For the RMSE, SGBRT gives the smallest error for turbine one (103.26). ERT provides the best results for turbine two and three (89.51 and 115.57).
- The tree-based methods provide the best results and KNN performs much worse. The average MAE for the tree-based methods across the three turbines is 200.53 (σ = 5.20), while the KNN for the three turbines together is 248.91. This is a difference of 48.38. Considering that the standard deviation of the MAE of the tree-based methods is 5.20, it can be concluded that KNN performs significantly worse.
- When five input parameters are used by the KNN method it even performs worse in comparison to when only the wind speed is used as input. The MAE

increases by 6.90 % when the KNN approach uses the 5 input variables compared to when it only uses the wind speed as input. This is expected as it was also observed in the work of Schlechtingen et al. [40]. They remark that by increasing the dimensionality of the feature space, the amount of neighbors K should decrease to achieve a good score. As a consequence the model becomes more sensitive to outliers.

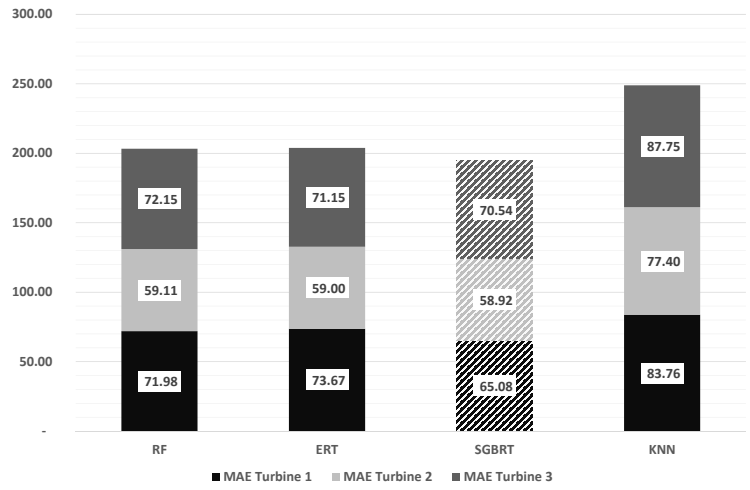
The hyperparameters for the models for the different experiments are listed in Table 2.3. The impact of changing the different hyperparameters for the gradient boosted regression trees model can be seen in Figures B.1; B.2; B.3 and B.4 for the number of trees, the subsample ratio, the learning rate and the maximum depth respectively. Optimizing the hyperpameters results in limited improvements. It should be noted that if the learning rate is very small, the predictions are much worse, however, this can be mitigated by increasing the number of trees. Finally, the maximum depth has a larger impact, but in general it seems that the depth has to be at least 3. For the random forest model and for the extremely randomized trees model, the impact of modifying the number of trees can be seen in Figure B.5 and Figure B.6 respectively. From these figures, it can be concluded that optimizing the number of trees has a limited impact on the results.

Algorithm	Parameters	Value turbine 1	Value turbine 2	Value turbine 3
KNN	K	5	5	5
RF	Number of trees	300	300	400
ERT	Number of trees	500	50	300
SGBRT	Loss	Least squares	Least absolute deviation	Least squares
	Learning rate	0.05	0.01	0.01
	Number of trees	300	1000	1000
	Subsample ratio	0.3	0.5	0.8

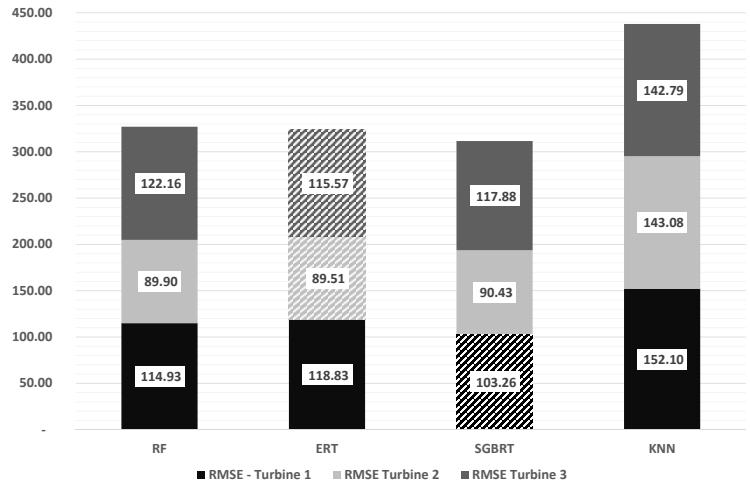
Table 2.3: Hyper-parameters for the multivariate data-driven performance models for the three offshore wind turbines.

2.6.1 Feature importance

Multivariate approaches have more than one input feature. However, the features are not equally important to predict the output power. As stated in Section 2.3, by using tree-based methods, importance of a feature can be calculated. Figure 2.7 presents the importance of the features to the SGBRT model. The results indicate that the performance of the SGBRT model depends on how much influence the additional features have on the generated power. The difference in MAE between the univariate case (Figure 2.5a) and multivariate case (Figure 2.6a) regarding the SGBRT model for turbine one, two and three is 24.89; 3.1 and 10.91. Per turbine, this is an improvement of 27.66 %; 5.00 % and 13.39 % respectively. When this



(a)



(b)

Figure 2.6: Stacked bar charts of the (a) MAE and (b) RMSE of the predictions for the power output of three wind turbines by multivariate approaches. Shaded bars indicate the best score for a turbine.

improvement is compared to the feature importance scores in Figure 2.7 a logical trend can be observed. A model, for which the wind speed is very important, achieves little improvement. Vice versa, when the other parameters are very important to the model, there is a greater improvement. For turbine one, the wind speed is of 40.07% importance according to the SGBRT model. This model has a large improvement of 27.66 %. For turbine three, the wind speed is of 53.65 % importance according to the SGBRT model. This model has a good improvement of 13.39 %. Finally, for turbine two the wind speed is of 62.28 % importance according to the SGBRT model. This model has a modest improvement of 5 %.

Regardless of the univariate or multivariate input, the models for turbine two receive the best and hence lowest MAE and RMSE. Additionally, the improvement achieved by using more variables as input, is small for this turbine. These facts indicate that turbine two is less influenced by environmental and operational aspects during the measured time period compared to turbines one and three, since most of the information is considered to be present in the wind speed. Fortunately, if the environmental and operational aspects influence the performance of the turbine more, the multivariate models are able to capture this as the other models improve considerably when more variables are used. This also reaffirms the fact that by using data-driven, non-parametric approaches a model can learn how to use additional information without explicitly modeling the physical relationship between the parameters.

2.6.2 Partial dependence

A common method to understand the relationship between a feature and a prediction is to calculate and visualize partial dependence [48]. This is done by observing how changes in the input affect the prediction. Partial dependence for one feature is calculated following Equation 2.12. N is the number of samples in the data set, $pred$ is the already trained prediction model (i.e. SGBRT) that takes a sample ($\mathbf{x}_i$) as input. In this sum, the j-th feature is fixed and set to v. The original input data is kept fixed. Only the value of one of the features is varied and predicted to investigate the influence of said feature on the output. An example of this procedure can be seen in Table 2.4. In the table there are five input variables (i.e. the wind speed, wind direction, yaw, pitch and RPM) and the output power which is predicted by the trained model (i.e. predicted output power). In this example we want to know what the influence is of the RPM on the predicted output power. Hence, the RPM of all samples is set to a fixed value. Subsequently, the output power is predicted. Afterwards, the predictions are averaged. This means that, in the example, we now know what the average output power will be if the RPM is set to 5, while marginalizing the other features. Next, the RPM is set to another value for every sample to see the new value's effect on the output. This procedure is done for a

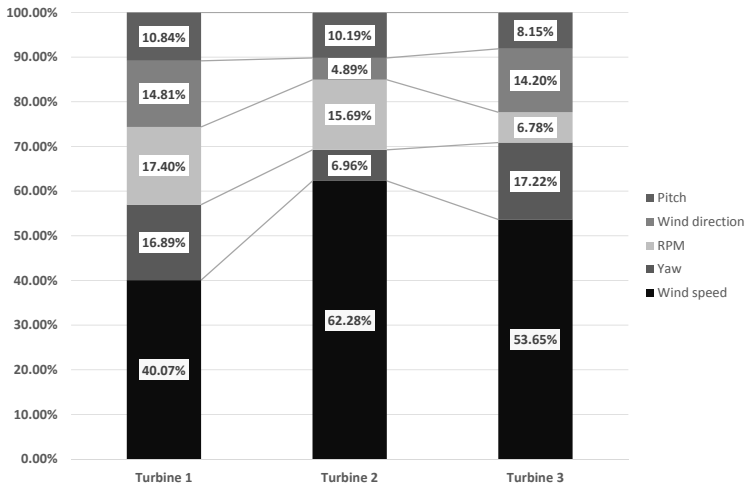


Figure 2.7: The importance of the different variables for the different turbines according to SGBRT.

wind speed (m/s)	**wind direction**(°)	**Yaw** (°)	**Pitch**(°)	**RPM**	**Predicted output power** (W)
5	10	5	20	5	500
6	11	6	21	5	510
5	11	7	20	5	505
4	12	7	19	5	490
3	11	8	18	5	480
4	10	7	19	5	485
5	9	8	20	5	470
6	8	6	21	5	465

Table 2.4: Example how the partial dependence is calculated.

range of RPM values to get a full view of the effect of the RPM on the prediction.

$$pdp_f(v) = \frac{1}{N}\sum_{i}^{N} pred(\mathbf{x}_i) \text{ with } x_{if} = v \tag{2.12}$$

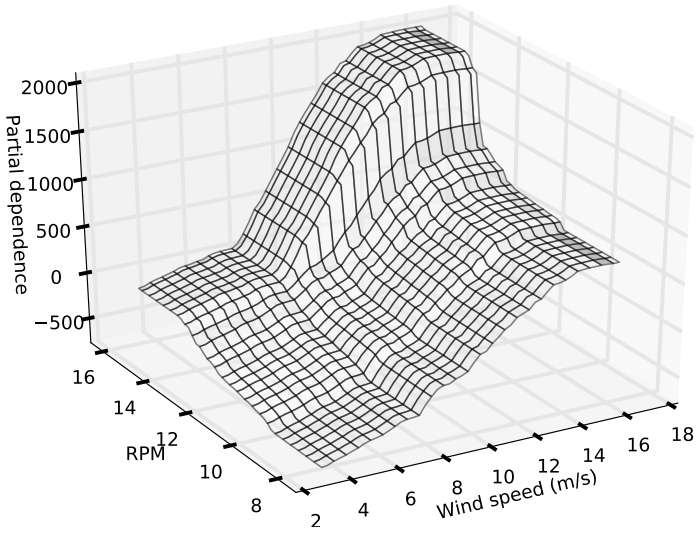
From the feature importance we observed that the RPM and the wind speed are important to the model to predict the output power. Hence, the partial dependence of the prediction on the RPM and wind speed is calculated. The partial dependence plot is given in Figure 2.8. In this plot it can be seen that there is a strong interaction between the RPM and the wind speed. When the wind speed rises, the output power will also rise (partial dependence). However, the RPM will magnify this

effect. For example, when the RPM increases when the wind speed is already at its maximum, the power output will still rise. As can be seen, the model has learned the non-linear interaction between the RPM, wind speed and the output power without requiring it to be modeled explicitly.

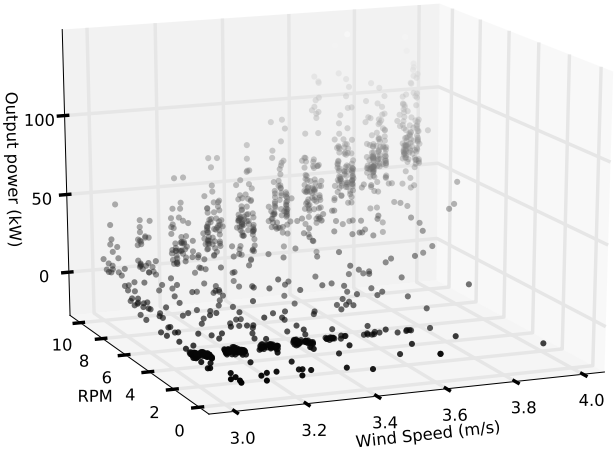
The interaction mentioned above can be seen in the data as Figure 2.8a illustrates. A small range of the wind speed is plotted versus the RPM and the measured output power. In this plot it can be observed that when the wind speed is maximal, the increase in RPM will result in an increase in the output power. The interaction that can be observed in the data in this scatter plot can also be seen in the partial dependence plot. Hence, it can be concluded that the data-driven model has learned the phenomena in the data.

Another phenomenon can be seen when looking at the partial dependence on the pitch and the wind speed (Figure 2.9). In this figure it can be seen that the wind speed is a good indicator for the output power as long as the pitch is above 0. When the pitch is lower than 0, the output power will be reduced, especially if the wind speed is high. This is a result of the wind turbine's pitch control system. This system will adjust the blade pitch into negative angles when there is a strong wind to keep the rotation speed within limits.

Similar partial dependency plots can be made for the yaw and the wind direction. However, these do not show any clear interpretable interaction, as there is no wind direction that influences the power prediction differently than the other wind directions. Hence, they are not included here.



(a)



(b)

Figure 2.8: Partial dependence plot (a) of the RPM and wind speed. Scatter plot (b) of the RPM, wind speed and measured output power, illustrating the interaction of these variables.

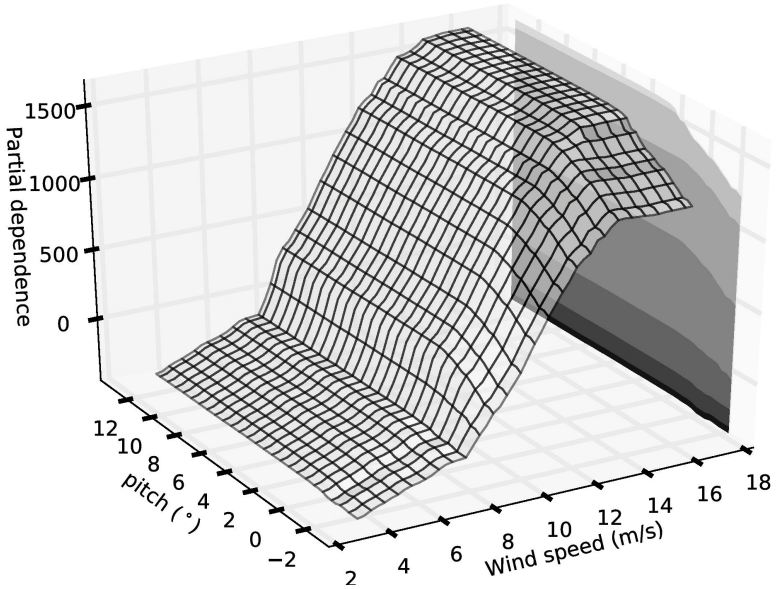
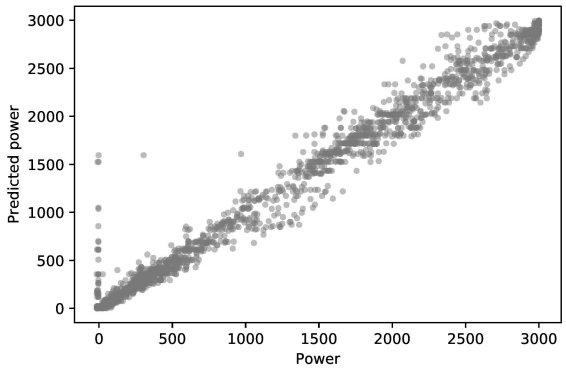


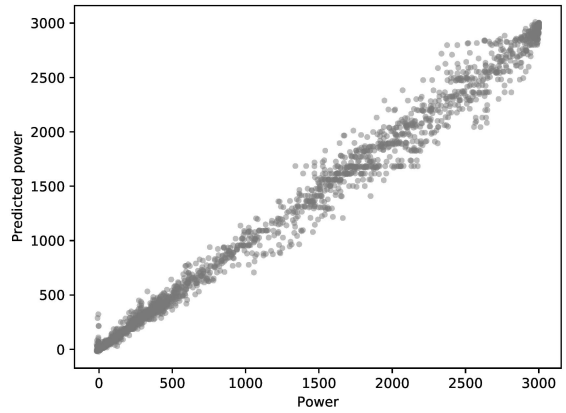
Figure 2.9: Partial dependence plot of the pitch and wind speed illustrating the interaction of these variables. The figure show the 3D surface of the partial dependency, but also a contour plot to highlight the effect of the pitch. Here too, the partial dependence means the effect on the output power.

2.6.3 Generated power versus the predicted power

The general improvement by multivariate power curve modeling is also visible when plotting the actual generated power versus the predicted power Figure 2.10. These plots show that for the multivariate case the points are closer to the $y = x$ line, where the generated power is equal to the predicted power, compared to the univariate case.



(a)



(b)

Figure 2.10: Scatter plots of the actual power output versus the predicted output power by the SGBRT model when only using the wind speed as input variable (a) and using five input variables (b).

2.7 Discussion: data-driven versus physics-based

In this section we reflect on the trade-off between data-driven and physics-based approaches.

It is shown that data-driven approaches can improve predicting the output power of wind turbines. These data-driven approaches have several advantages. First, as illustrated in this chapter, data-driven methods have the advantage of being able to capture the phenomena in the data. Second, data-driven techniques

are easy to apply, resulting in an approach that can be implemented in a short amount of time. Third, the techniques applied in this chapter do not require a lot of computational resources. For example, the experiments were run on a standard laptop with an Intel-core i5 processor which has a clock speed of 1.7 Ghz 8 GB of RAM memory. To train the different methods on the data, less than 0.5 seconds is required per method. Fourth, data-driven performance monitoring, as presented in this chapter, only requires data form the supervisory control and data acquisition system, which is a standard component in offshore wind turbines. Hence, no additional sensors have to be installed, keeping the cost low. Additionally, limited expert or domain knowledge is required. Finally, it should also be noted that data-driven methods will create models adapted to the turbine from which the data originated. This means that such a model will only perform optimally for that turbine. Hence, should a model be used for another turbine, it will perform less well.

Physics-based approaches heavily rely on expertise knowledge. Furthermore, models are often created for a certain type of turbine instead of per installed turbine. This is because turbine dependent phenomena, such as degradation, turbulent air at the turbine's location, shading effects or icing are not yet incorporated in physical models, are imprecise and unreliable, or are difficult to incorporate. Even if phenomena such as wake effects are modeled using computational fluid dynamics and operational data, considerable computational power and expertise is required [29]. Because of these disadvantages, physics-based approaches are more difficult to use in practice and slower to implement. Physics-based approaches do have a major advantage and that is that they are white-box systems, meaning that they are human interpretable, which is not the case for data-driven methods. Powerful data-driven methods are often black-box systems, which are not human interpretable. However, some techniques, such as those used in this chapter, can provide some insights into the data, but these are rather limited compared to physics-based approaches.

Both data-driven as well as physics-based approaches have their respective advantages and disadvantages. Hence, for future research, it would be interesting to investigate how to combine them. For example, additional domain expertise could be used to further improve the data-driven approaches and make them more interpretable.

2.8 Conclusion

In this chapter performance monitoring for offshore wind turbines is improved. We optimize and compare existing machine learning models that have not been used within this context before to achieve this.

First, power curve modeling was done on a synthetic data set generated ac-

cording to the procedure described in [38] and adjusted with the method proposed in [47]. The resulting synthetic data set used in this research, is made available online. The results we achieved using this data set indicate that non-parametric data-driven methods outperform the state-of-the-art approach.

Second, we modeled the power curve of three offshore wind turbines using six algorithms providing only the wind speed as input. The results indicate that the state-of-the-art approach is not able to outperform the non-parametric methods on our data sets. These results confirm to the first opportunity highlighted in section 2.1 that univariate power curve modeling can be improved by using non-parametric data-driven methods.

Third, when only using one input variable, a limited amount of information is made available to the chosen algorithms. Hence, experiments were done using multiple input variables which are readily available from the SCADA system. Five variables are used as input i.e. wind speed, rotations per minute of the rotor, yaw, wind direction and pitch. Our experiments show that by providing these additional variables to the models, an overall improvement can be seen for our data sets, except for the k-nearest neighbors model. The largest improvement is achieved using the data from turbine one, resulting in an improvement of 27.66% compared to the univariate solution. This confirms to the second opportunity identified in section 2.1 that data-driven power prediction can be improved by incorporating multiple variables in the model.

The results in this chapter show that data-driven approaches work very well, but are however limited interpretable. Nevertheless, they can be constructed in a short amount of time using data that is readily available through the SCADA systems of turbines. The data-driven models will capture the phenomena in the data if present and therefore performance monitoring can be improved.

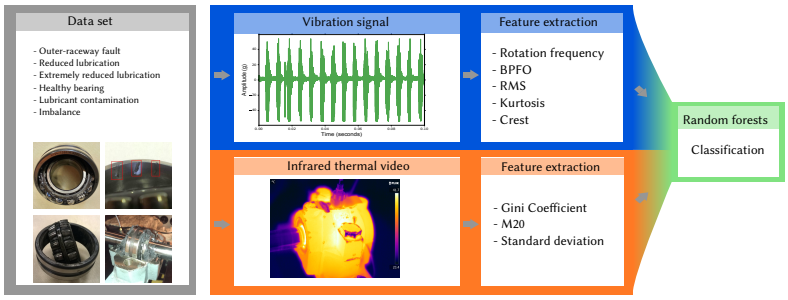
The work presented in this chapter is published in *Engineering Applications of Artificial Intelligence*:

- Olivier Janssens, Nymfa Noppe, Christof Devriendt, Rik Van de Walle, Sofie Van Hoecke, *Data-driven Multivariate Power Curve Modeling of Offshore Wind Turbines*, vol. 55, pp. 331-338, 2016

Chapter 3

Component level: Multi-sensor fault detection using feature engineering

Chapter highlights



In this chapter, we focus on detecting conditions/faults related to bearings using feature engineering. We consider faults such as lubrication inadequacies, which are currently difficult to detect. Therefore, a feasibility study is done on data-driven fault detection using infrared thermal imaging. We show that up to 88.25 % of the fault occurrences can be detected using our approach. To achieve this, two new image processing features are introduced to the field of infrared thermal imaging. For comparison, we also construct a vibration-based system. In the end, both systems are combined creating a novel multi-sensor fault detection system. We show that this system surpasses the performance of the individual systems.

3.1 Introduction

As discussed in Chapter 2, performance monitoring is a good technique to estimate the overall health of a wind turbine without having to know all the details of the machine. Performance monitoring helps to enable preventive maintenance because when under-performance is detected maintenance can be scheduled before a wind turbine breaks down. However, it will not enable an operator to pinpoint the exact problem in the machine and there is no guarantee that the maintenance cost will be reduced. To make maintenance labor more efficient, details such as the component, type of fault, severity, and location need to be known. For example, performance monitoring will show that a wind turbine is performing sub-optimal, however, if such a condition monitoring system could detect that a bearing in the gearbox suffers from damage, and what type of fault is present, the urgency can be estimated more precisely and the correct maintenance preparations and time scheduling can be done before going to the offshore wind turbine.

As has been mentioned in the Chapter 1, the components that are prone to failure in a wind turbine are bearings. Hence, in this chapter the focus lies on condition/fault detection related to rolling element bearings (REB). To monitor individual components of a machine, such as bearings, additional sensors have to be installed (in, on or near the machine). Sensors/data used for bearing fault detection are for example ultrasound, proximity probes, thermocouples and accelerometers. Accelerometers, which measure vibrations, can be used to detect a large part of the impending bearing failures [49]. Generally, the time between the detection of a potential fault and functional failure for vibration analysis is considered to be long, in the range of weeks to months. Vibration analysis has therefore been the method of choice when it comes to fault detection for rotating machinery, gears, and bearings in general.

Vibration-based fault detection for individual components is mainly done using mature, well-understood model-driven techniques [50]. Based on the geometry of rolling element bearings, it is possible to calculate, and therefore determine, certain vibration frequencies that are useful to identify faults using signal processing techniques [14]. Similarly, based on the physical properties of a rotating machine, certain characteristics linked to conditions can be calculated and hence be identified using signal processing. For example, a rotating machine can suffer from imbalance. Imbalance is caused by the fact that a rotating body (e.g. rotor) has an unbalanced rotating mass. During rotation, excess centrifugal force is exerted by the heavier side of the rotor, resulting in a small displacement. The acceleration related to the displacement occurs at the rate of the rotational speed. Hence, in the frequency spectrum of an acceleration signal –measured by an accelerometer– a high amplitude can be observed at the rotational frequency of the machine. Similarly, damages to the bearing's raceway will cause a peak at the bearing's fault

frequencies [14, 51].

Besides indicative frequencies, certain time-based statistical features have been identified for bearing fault detection. Such an approach can be considered data-driven feature engineering, as no knowledge of the underlying physics is used to create the features. Examples of useful statistics are for example kurtosis and crest factor [52, 53]. Also, the root-mean-square (RMS), another time-based feature, has been shown to be indicative of the amount of separation between the rolling elements and the raceways due to the lubricant in a linear bearing [54].

However, vibration analysis has its limits. Contrary to raceway damages, the detection of lubricant starvation remains difficult because a lubrication related fault will not manifest itself as a new cyclic frequency [55]. Today, state-of-the-art techniques for predictive health monitoring of rolling element bearings do not work for monitoring lubrication related conditions such as over-lubrication or under-lubrication [53].

Another type of lubrication inadequacy is hard particle contamination. These particles can consist of steel fragments, sand or any other type of residue and can damage the bearing and reduce the operational lifetime of the machine significantly. Hence, most of the largest bearing manufacturers state that lubricant contamination is the major root cause of bearing failure before they reach their rated life [56].

To detect the lubricant related problems in offshore wind turbines, regular offline lubricant analysis is done. Nevertheless, this is a very intrusive and hands-on condition monitoring method as it requires a technician to manually sample the lubricant, bring it on shore and subsequently analyze it. By analyzing the lubricant it is possible to categorize the contamination level according to the ISO 4406:99 or ISO 11171, if oil lubricant is used. As lubricant analysis is an offline procedure, it is possible for a fault to escalate and damage the bearing permanently. Hence, lubricant analysis can be complemented with an online particle counting sensor that measures how contaminated the lubricant is in real-time enabling fault escalation prevention. Nevertheless, these sensors can only be installed closed-loop oil lubrication systems. Also, online particle counting is not possible when grease lubrication is used. Furthermore, online particle counters are very expensive to install [49, 57] and cannot be used to detect other types of faults, i.e. raceway faults or imbalance.

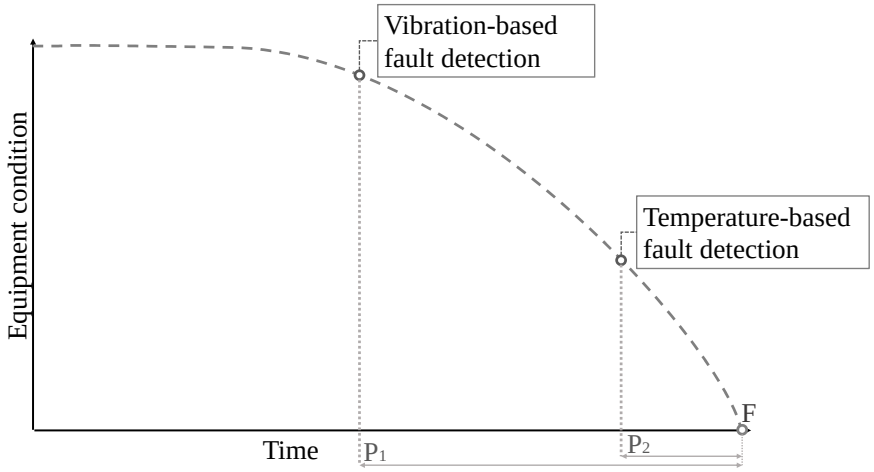
It is also possible to employ temperature-based condition/fault detection using thermocouple sensors. Thermocouples are relatively cheap compared to other sensors and allow for temperatures to be monitored inside machines. However, using temperature-based fault detection, there is a much shorter time span between the detection of a potential fault in a machine and the functional failure [49] as can be seen in the P-F curve in Figure 3.1a. The time span is often in the range of hours to days. Hence, temperature-based detection is often only useful after initial

fault detection using vibration analysis. However, in recent years infrared thermal (IRT) imaging has gained noticeable attention as it allows for visual temperature monitoring and fault localization. Because of the advantage of having spatial information of the heat in components, IRT imaging has been applied in several domains for specific tasks such as inspection of cracks, isolation, subsurface moisture, corrosion, gas flow, air flow, and welding processes [58]. Furthermore, by applying image processing and machine learning, IRT imaging can be used for autonomous/online fault detection. For example, IRT-based fault detection has been applied for the detection of conditions in rotating machinery such as imbalance, misalignment, coupling looseness and bearing damages. These faults are often considered only to be detectable using vibration-based techniques. As, IRT-based automated fault detection is still rather new and not based on models of the underlying physics. Current approaches are purely data driven resulting in non-human interpretable features.

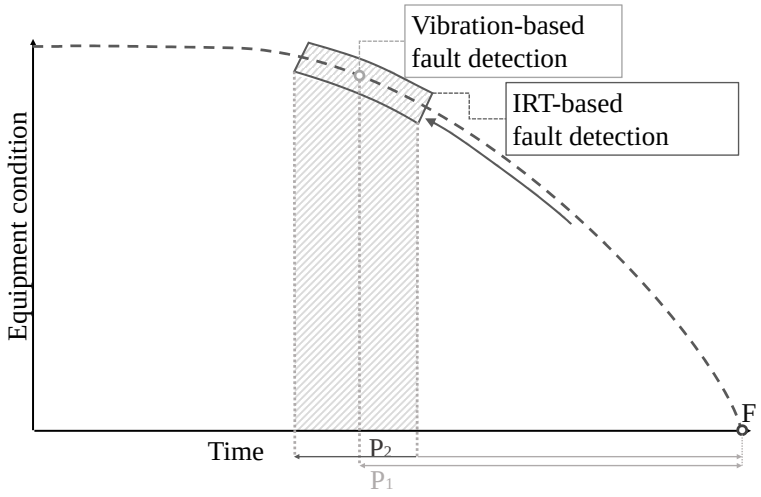
In this chapter, the feasibility of IRT-based fault/condition detection on component level is researched. IRT-based fault detection is still in its infancy, and the application on rotating machinery is limited. A wind turbine is too complex to be used in a feasibility study. Therefore, a lab-scale test-setup has been built. The feasibility study is done for several bearing conditions and machine conditions. The conditions consist, among others, of faults that occur early in the fault escalation process and which are very expensive to detect, not detectable in real-time or not detectable at all with current techniques such as vibration analysis. Such conditions are over-lubrication, under-lubrication and contaminated lubrication, which haven't been researched before using infrared thermal imaging. Such faults can escalate and propagate leading to damages. Thus, being able to detect those, the time between the detection of a potential fault and the functional failure is increased as is illustrated in Figure 3.1b.

We illustrate that IRT-based fault/condition detection, using our proposed engineered features, has certain limits and hence, a multi-sensor system is proposed which uses IRT video and vibration measurements. This system uses simple expert features from the mature model-based vibration analysis domain, which have been proven to work well, and data-driven statistical features extracted from the IRT data. This data is processed and combined and subsequently given to a machine learning algorithm. The algorithm is therefore capable of providing new knowledge, i.e. the condition of the component to an operator. By providing additional knowledge on component level, better maintenance actions become possible.

To put the multi-sensor based system's results into perspective, in this chapter first a single-sensor vibration-based system is proposed. Subsequently, a single-sensor IRT-based system is discussed and finally the strengths of the two single-sensor systems are combined in a multi-sensor system.



(a)



(b)

Figure 3.1: Potential failure and functional failure curves, illustrating (a) the time-span difference between vibration-based and temperature-based fault detection, respectively (b) vibration-based fault detection and infrared thermal imaging-based fault detection. The region indicates that using infrared thermal imaging ,the time-span can possibly be increased if lubricant related faults are present.

3.2 Related literature

In this chapter, the focus lies on condition/fault detection for rotating machines. Hence, in this section first the related literature is discussed regarding vibration-based fault detection in rotating machinery. Next, a literature overview is given of infrared thermal imaging based fault detection. Finally, related literature is discussed regarding multi-sensor approaches.

3.2.1 Vibration-based fault detection

As has been discussed in the introduction, using vibration measurements to detect the condition of a component in a machine, such as a rolling element bearing, is an established technique. Based on the physics of the faults and the dynamics of the system, several properties in the frequency spectrum have been related to specific conditions and damages. Often, features such as the amplitude at fundamental frequencies are used to identify certain machine faults. Furthermore, statistical features have also been used to detect bearing faults [52,53]. To be able to detect a fault automatically, several machine learning algorithms have been used in the past such as K-nearest neighbors classifiers, naive Bayes classifiers, decision trees and multi-layer perceptron classifiers [59–61]. Using these classifiers, several types of faults can accurately be detected such as inner-raceway faults, outer-raceway faults and rolling element faults.

However, some faults are more difficult to detect reliably such as lubricant starvation [55], which can be caused due to grease dry-out. Lubrication has many functions, such as friction control, wear control, contamination control, temperature control and corrosion control. Lack of lubricant is often the root-cause of many bearing failures [49]. If lubricant starvation is not detected in time, other additional faults may be induced, making it more difficult to identify every individual fault.

As discussed by Kankar et al. [61] and Monte et al. [62], when several faults are present in a rotating system at the same time, the detection of the faults is more difficult. Hence, more advanced detection techniques are required. One of the possible options is a multi-sensor system.

3.2.2 Infrared thermal imaging based fault detection

Infrared thermal imaging for fault detection in rotating machinery gained noticeable attention in the last years [63–69]. The focus of previous research has been on the detection of conditions such as shaft misalignment, bearing looseness, load imbalance and bearing faults. To detect these conditions, data-driven approaches have been applied using image processing and machine learning. Such approaches often follow the same pipeline as is displayed in Figure 3.2. Often, the first step of

the image processing pipeline entails the extraction of the region of interest (ROI). This is done manually or via an algorithm such as Otsu thresholding together with k-means clustering [63] or watershed-based algorithms [64]. The second step can consist of enhancing the image or ROI [66]. From this (enhanced) ROI, statistical features are derived such as the standard deviation, mean, skewness, kurtosis, variance, entropy, energy, central moments, maximum and minimum [63,65,66] of the components of the discrete wavelet decomposition of the thermal image [67, 68]. In the penultimate step, undiscriminating features are sometimes removed or combined to create better features. Algorithms used for this step include principle component analysis [64], independent component analysis [65], discriminant analysis [66] or relief algorithm [67,68]. The resulting features are subsequently used to determine the condition of the rotating machine using a classification algorithm. The classification algorithms which have been employed are support vector machine [65, 69], relevance vector machine [66], self-organising map [63] and linear discriminant analysis [67, 68]. Most of these approaches result in a system that can accurately detect the latent machine condition with an accuracy of 74 % up to 100 % [63–69].

Figure 3.2: General steps in an image processing/machine learning pipeline.

Several remarks should be noted regarding the related literature:

1. Often the classification algorithms are trained and tested on IRT data of the same bearing and recording. Therefore, a generalized solution is not guaranteed, i.e. it cannot be confirmed that the model will work using data from new bearings.

2. The conditions and faults under investigation are those that can easily be detected using vibration analysis, which is a robust and widely used fault detection technique, only illustrating that infrared thermal imaging is a more expensive but valuable alternative.

3. The detection of different amounts of imbalance is still challenging in combination with other faults [50].

In the approach proposed in this chapter, several test-runs of different bearings are done to create disjunct training and test sets ensuring a more generalized solution. Furthermore, among the faults and conditions considered are some that are difficult to identify or hardly detectable using today's detection techniques. Finally, multiple conditions and faults are considered simultaneously.

3.2.3 Multi-sensor systems

Regarding multi-sensor approaches, little research is available. However, it has been shown that multi-sensor systems outperform single-sensor based systems [70–72]. For example, to detect a cracked rotor, rotor rub or coupling misalignment, improved fault detection can be achieved when vibration data is supplemented by temperature measurements [70]. The advantage of a multi-sensor approach is recognized in other disciplines as well, such as fire detection and health-care [71, 72].

3.3 Set-up

In order to investigate the feasibility of infrared thermal-based, but also multi-sensor-based fault/condition detection system, a lab-scale set-up was used to create two data sets containing different conditions. The conditions in data set one can be seen in Table 3.1, and those for data set two in Table 3.2. The set-up used to create the various conditions is a rotating set-up which is depicted in Figure 3.3. The set-up has a motor which makes a shaft rotate. The shaft is supported by two rolling element bearings. Only the condition of the bearing that is removed the furthest from the motor (i.e. right-hand side) is changed in between runs. Hence, only the housing of the right-hand side bearing is monitored by the infrared thermal camera. The position of the infrared thermal camera can also be seen in Figure 3.3. The set-up is located in a darkened room to limit the noise in the thermal images due to external sources of infrared light. As mentioned in the introduction, also the vibrations of the set-up are recorded. Two accelerometers are mounted on this bearing housing to measure the acceleration in the x-direction and y-direction. Finally, also thermocouples are placed inside the darkened room to measure the ambient temperature. More details about the set-up, infrared thermal camera[1] and accelerometers can be seen in Table 3.3 and 3.4.

	No imbalance	**Imbalance: 13 g or 17.3 N**
Healthy REB (HB)	Condition 1	Condition 2
Outer-raceway fault (ORF)	Condition 3	Condition 4
Mildly inadequately lubricated bearing	Condition 5	Condition 6
Extremely inadequately lubricated bearing	Condition 7	Condition 8

Table 3.1: Summary of the 8 conditions in data set 1.

The various conditions are created artificially. To imitate outer-raceway faults (ORF) in the bearings, three small shallow grooves were added mechanically on the bearings' outer-raceway as can be seen in Figure 3.5c. When mounting the

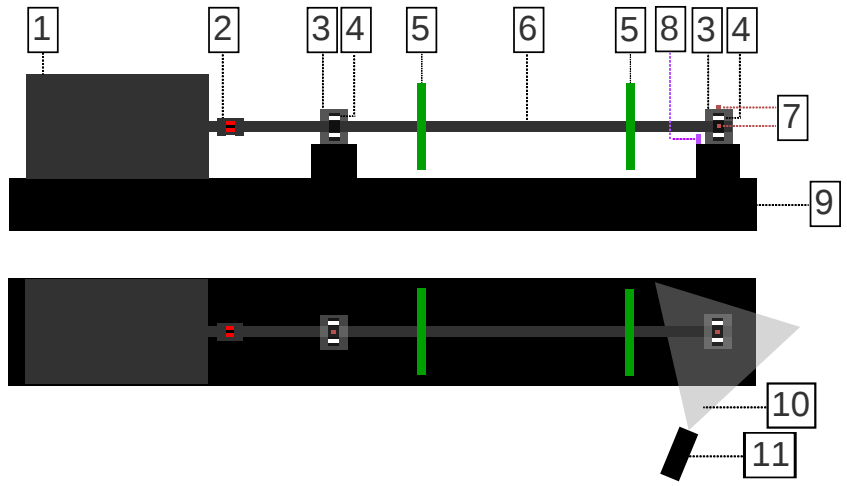
[1] This camera was bought with the support of Flanders Make

Condition	Bearing	Imbalance
1	Healthy bearing	Balanced
2	Healthy bearing	4.1 g or 5.5 N
3	Healthy bearing	9.3 g or 12.4 N
4	Healthy bearing	13 g or 17.3 N
5	Outer-raceway fault	Balanced
6	Outer-raceway fault	4.1 g or 5.5 N
7	Outer-raceway fault	9.3 g or 12.4 N
8	Outer-raceway fault	13 g or 17.3 N
9	Hard particle contamination	Balanced
10	Hard particle contamination	4.1 g or 5.5 N
11	Hard particle contamination	9.3 g or 12.4 N
12	Hard particle contamination	13 g or 17.3 N

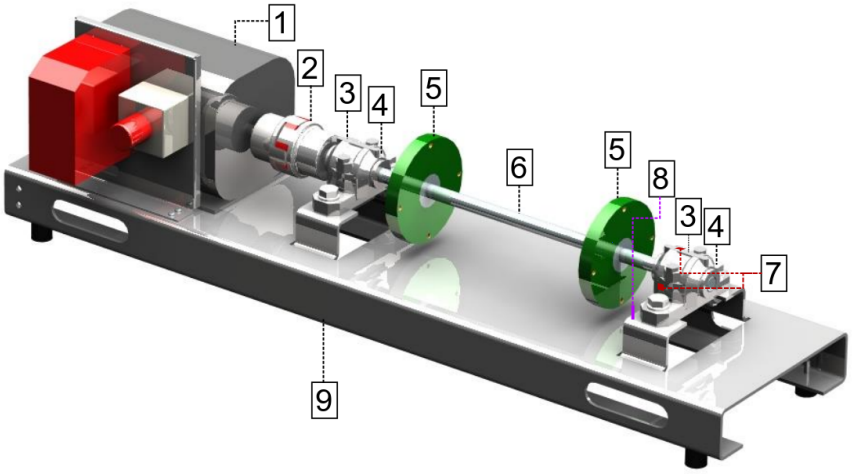
Table 3.2: Enumeration of the 12 induced conditions in data set 2. Every condition consists of a bearing condition –which are healthy bearing (HB), outer-raceway fault (ORF) or hard particle contamination (HP)– but also an imbalance gradation, which are balanced, 4.1 g; 9.3 g or 13 g of added weight to the rotating rotor.

Property	Value
Accelerometer type	IEPA
Accelerometer product type	4534-B
Accelerometer Brand	Brüel & Kjaer
Bearing code	FAG 22205-E1-K
Bearing type	Spherical roller bearing with tapered bore and adapter sleeve
Housing code	SNV052-F-L
Housing type	Closed plummer block
Grease	Molykote BR 2 plus
Rotation speed	25 Hz or 1500 rotations per minute
Eigenfrequency of the set-up determined using an impact test	17.8 Hz
Vibration sample frequency	51,200 Hz
Motor type	Single phase asynchronous induction motor
Motor power	1.1 kW
Weight mass disk	2.317 kg
Length shaft	68.7 cm
Diameter shaft	2 cm
Weight shaft	2.317 kg
Material shaft & disks	Steel

Table 3.3: Set-up details.



(a)



(b)

Figure 3.3: (a) Side-view and top-view of the set-up. (b) 3D image of the set-up. The labels are: 1. servo-motor; 2. coupling; 3. bearing housing; 4. bearing; 5. disk; 6. shaft; 7. accelerometer 8. thermocouple; 9. metal plate; 10 field of view; 11. IRT camera.

Property	Value
Thermal camera	FLIR SC655
Capture speed	6.25 frames per second (fps)
Resolution	640 x 480 pixels
Distance: camera - housing	38 cm
Emissivity	0.9
Lens	Macro lens
Spectral range	7.5 - 13 μm

Table 3.4: Thermal camera details.

bearings in the housing, the outer-raceway fault was placed at the 10 o'clock position (i.e. close to the top of the housing at the side which the IRT camera faces) for data set one and at the 6 o'clock position (i.e. loaded zone) for data set two. In Figure 3.4, these locations can be seen.

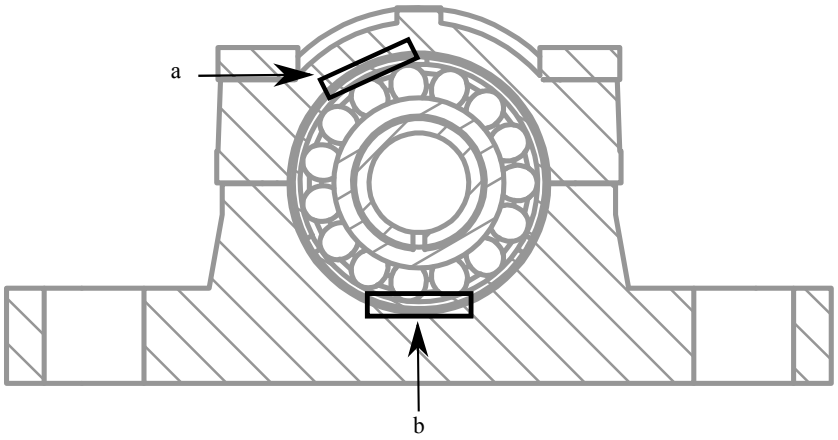
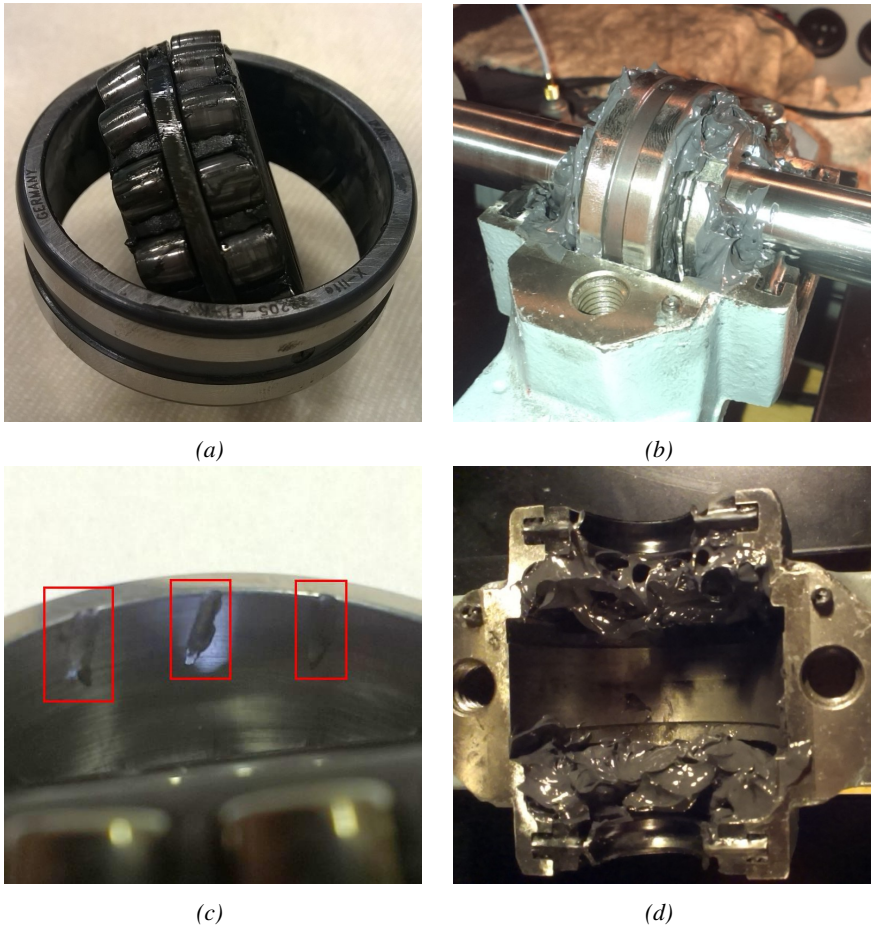


Figure 3.4: Section view of the bearing housing with the a bearing inside the housing. Region (a), is where the three shallow grooves were for data set one. Region (b) is the area the three shallow grooves were for data set two.

Lubricant grease is added to every bearing. The required amount of grease is 2.5 g and is determined using Equation 3.1, as recommended by the bearing manufacturer. In Equation 3.1 D is the outer diameter of the bearing and B the inner diameter [73]. For the used bearings D = 52 mm and B = 18 mm.

$$m = D * B * 0.0027 \text{ g} \tag{3.1}$$

Both the healthy bearings (HB) and those with an outer-raceway fault are placed in a housing which contains a grease reservoir (Figure 3.5d and 3.5b). The



(a) *(b)*

(c) *(d)*

Figure 3.5: Examples of different bearing conditions and the grease reservoir. (a) Mildly inadequately lubricated bearing. (b) Healthy Bearing. (c) Outer-raceway fault. (d) Grease reservoir.

grease reservoir contains 20 g of grease to fill the housing's cavities to the recommended 60 % [74]. For the bearings with reduced lubricant in data set one, i.e. mildly inadequately lubricated bearing (MILB) and extremely inadequately lubricated bearing (EILB), no grease reservoir is present. For the MILBs, the grease on each individual bearing is superficially removed (Figure 3.5a). Similarly, for the EILBs the grease in the bearings is decreased more. In Figure 3.6, the grease reduction can be observed. For the hard particle faults in data set two; 0.02 g of iron particles are mixed in the lubricant of the bearings.

To complete the data sets, all the different bearing conditions are also tested during imbalance, this is done by adding bolts to the right-hand side rotor at a

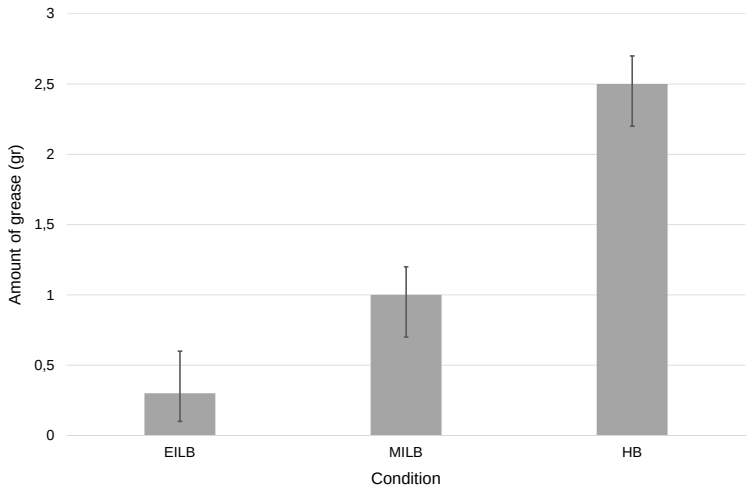


Figure 3.6: Amount of grease per condition.

radius of 5.4 cm. The weight of the bolts can be seen in Table 3.1 and 3.2.

3.4 Data-set

To construct data sets that are large enough to validate the proposed methods, every condition in both data sets is created for five different bearings. By using multiple bearings, variability is introduced in the data set due to manufacturing, mounting and grease distribution resulting in better generalized machine learning models when trained on this data set, contrary to the state-of-the-art. Each bearing is run for one hour, while the infrared thermal camera records the heating process. For data set one, in total, 5 bearings · 8 conditions · 1 hour = 40 hours of IRT data is recorded. For data set two; 5 bearings · 12 conditions · 1 hour = 60 hours are recorded. It is observed that the bearings' temperatures usually do not further increase after a duration of 50 minutes, i.e. they reach steady state [75]. Hence, only the last 10 minutes of each recorded hour are exported to AVI files for further processing and analysis. To reduce the size of the video files and maintain optimal image quality, lossless compression is applied using the H264 standard [76]. The resulting videos consist of monochrome frames for which the gray values correspond to temperatures in the range of 10 - 60°C for data set one; and 10 - 70°C for data set two. An example of a colorized image can be seen in Figure 3.7. Addi-

tional to the IRT data, vibration data is also logged using the two accelerometers. To synchronize the two modalities –similar to the infrared thermal data– only the last 10 minutes of vibration data is kept for further analysis.

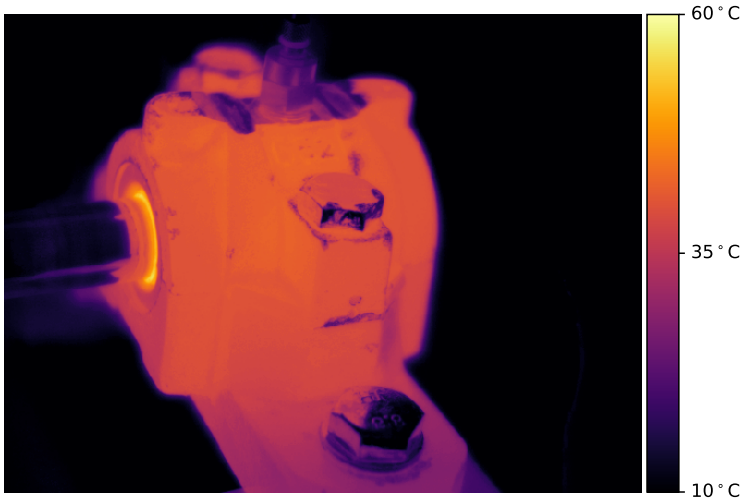


Figure 3.7: One frame of an IRT video of a healthy bearing.

In the next section a single-sensor fault/condition detection system is presented which solely uses vibration data.

3.5 Methodology: single-sensor vibration-based fault detection

Each measurement is assigned two labels: one for the machine condition and one for the bearing condition. Hence, we regard the condition detection task as a combination of two classification problems. Every 10-minute vibration recording is classified by the first classifier according to the amount of imbalance, and by the second classifier according to the bearing condition. This solution is depicted in the architectures in Figure 3.8a for data set one and in Figure 3.8b for data set two. The raw vibration data is used in two pipelines, each with their own respective feature extraction step, classification model and labels. By using an architecture with two pipelines, the combination of the two labels generated for each sample give

the final condition. For example, using the vibration data, pipeline one detects that the bearing is inadequately lubricated. At the same time, pipeline two detects that there is imbalance. Hence, by combining the two classification results the final condition is "Inadequately lubricated bearing during machine imbalance".

An alternative approach is a single pipeline system. However, a single pipeline systems has to be able to detect imbalance and the bearing condition at the same time. This results in more classes to distinguish compared to two separate pipelines which is a more difficult classification task resulting in worse results. For data set one there are in fact 8 classes (condition/fault combinations), which are more classes than the six classes of the two pipeline architectures, i.e. 2 classes (balance/imbalance) together with 4 classes (MILB, EILB, HB and ORF).

Another alternative consists of using k-1 binary classifiers (k being the amount of classes in the data set). Every classifier would have to indicate if a certain condition/fault is present or not. This solution has one problem and that is that this approach would result in class skew (i.e. not uniformly distributed class samples). This can result in reduced detection results. Hence, we choose to use a two-pipeline approach.

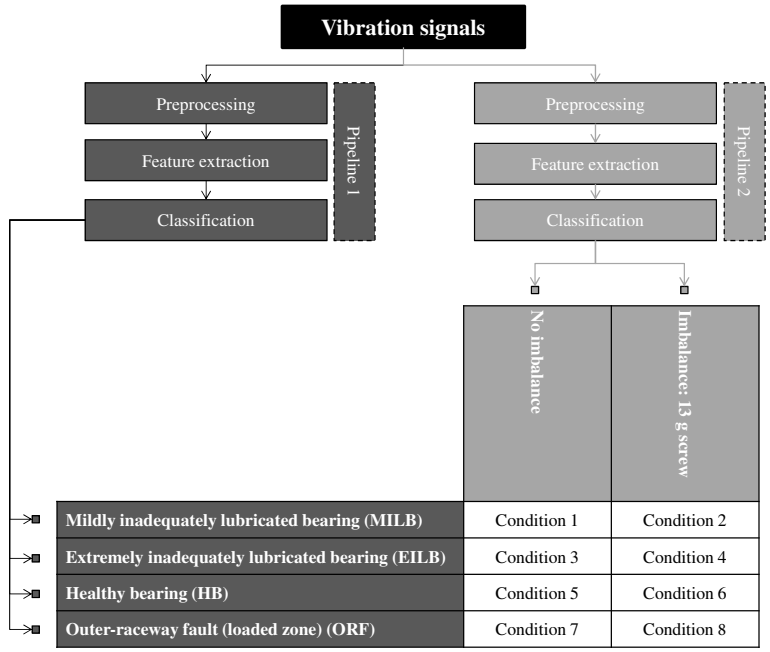
A high-level overview of how the vibration signals are processed next is given in Figure 3.9. Note that this procedure is done for every signal per pipeline.

3.5.1 Pipeline one

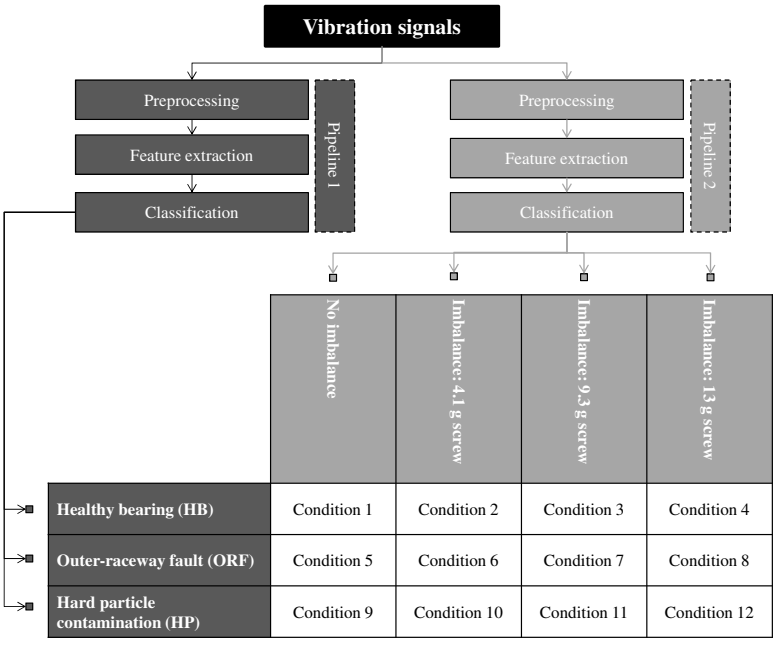
For data set one, pipeline one has to be able to detect if the bearing is healthy, inadequately lubricated, extremely inadequately lubricated or contains an outer-raceway fault. For data set two, pipeline one has to determine if there are hard particles in the lubricant, if the bearing is healthy or contains an outer-raceway fault (in the loaded zone). As Figures 3.8a and 3.8b, illustrate, the process start with preprocessing.

3.5.1.1 Preprocessing

Instead of considering each vibration signal as a single sample, windowing is used to extract several samples per signal (see Figure 3.9). This means that if the system can determine the condition of the machine after processing one window instead of one entire measurement. Furthermore, windowing results in a larger data set. A window contains one minute of vibration data, and overlaps by 50 % with its neighbouring window. This means that from every 10-minute vibration data recording, 19 windows are extracted, each containing 60 seconds $\cdot$ 51200 Hz $=$ $3,072,000$ samples. As there are two accelerometers mounted on the bearing's housing, there are in fact twice as many samples. The window length is experimentally determined, and provides the most optimal results. It should be noted that the window length, overlap and number of windows do not have a great influence



(a)



(b)

Figure 3.8: High level architecture of the fault detection system for data set one (a) and two (b).

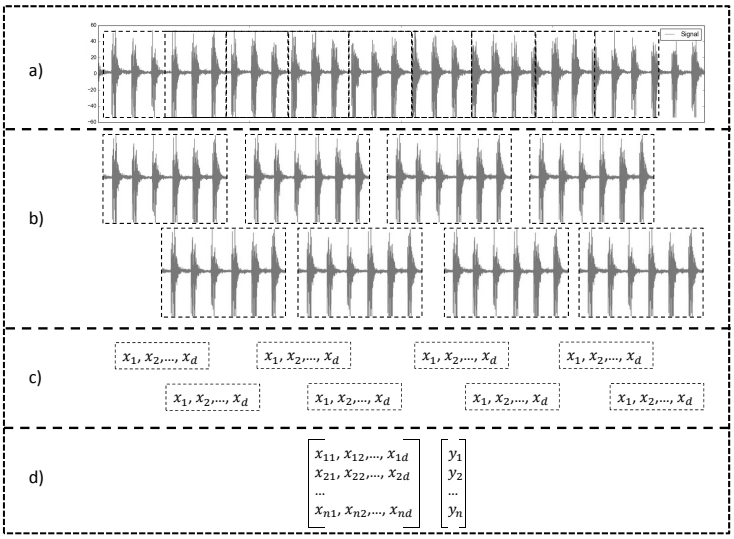


Figure 3.9: From a vibration signal in the time domain (a), overlapping windows are extracted (b). From each individual window, several features are extracted (c). Every set of features extracted per window results in a sample in the eventual design matrix (d).

on the classification results. Only the fact that windowing is used, has a significant impact on the eventual results.

3.5.1.2 Feature extraction

First of all, three statistical features are calculated: RMS, kurtosis, and crest factor. These features are chosen as they have been proven useful for bearing fault detection [52, 54]. The RMS, kurtosis and crest factor are calculated according to Equation 3.2; 3.3 and 3.4 respectively, where $\mathbf{x}$ is a vector of N samples in a window, and μ and σ respectively denote the mean and the standard deviation of $\mathbf{x}$.

$$RMS = \sqrt{\frac{1}{N}\sum_{i}^{N} x_i^2} \tag{3.2}$$

$$Kurtosis = \frac{\sum_i^N (x_i - \mu)^4}{N\sigma^4} \tag{3.3}$$

$$Crest = \frac{\max(|\mathbf{x}|)}{RMS} \tag{3.4}$$

When a rolling element of a bearing hits a fault in the outer raceway, the natural frequency of the raceway is excited, resulting in a high frequency burst of energy which decays and is then excited again as the next rolling element hits the fault. This high frequency impulse is superimposed on a carrier signal (with a frequency of 25 Hz for our experiments) which originates from the rotating machine. A simulated example of this can be seen in Figure 3.10.

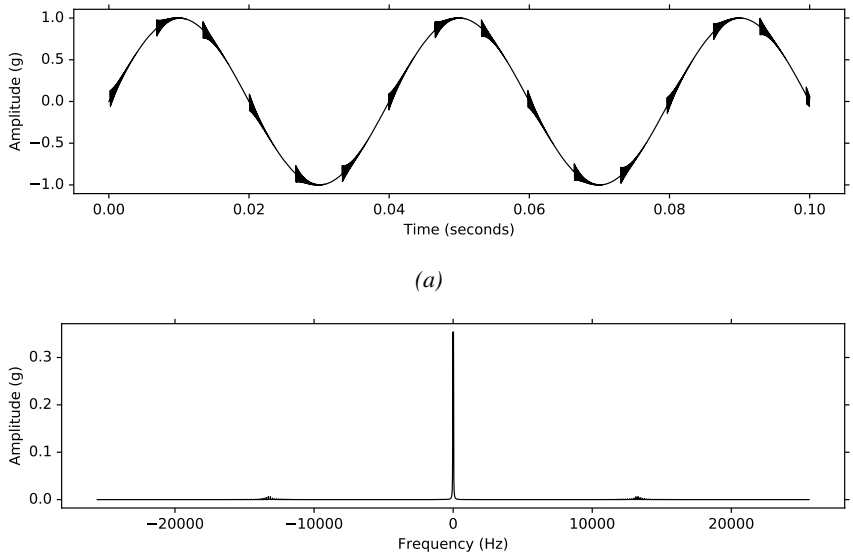


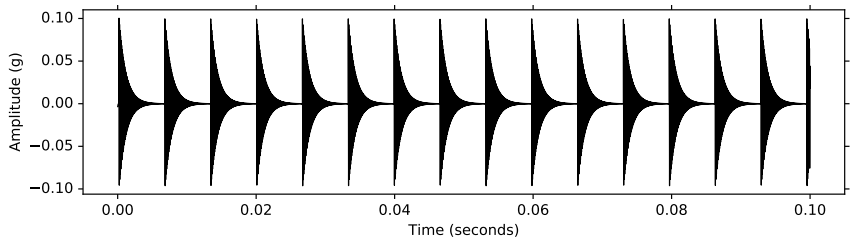
Figure 3.10: Simulated signal of a carrier wave on which high frequency burst are superimposed. (a) shows the signal and (b) the frequency spectrum of the signal.

To identify a fault, it is necessary to detect the frequency of occurrence of these bursts in the signal's frequency spectrum, together with its amplitude. The frequency at which these bursts occur is called the ball pass frequency of the outer raceway (BPFO). The BPFO can be calculated using Equation 3.5, where n is the amount of rolling elements, f the rotation frequency, d the diameter of the rolling elements, D the diameter of the rolling-element cage and α the contact angle. This results in a BPFO at 150.41 Hz for the chosen bearings.

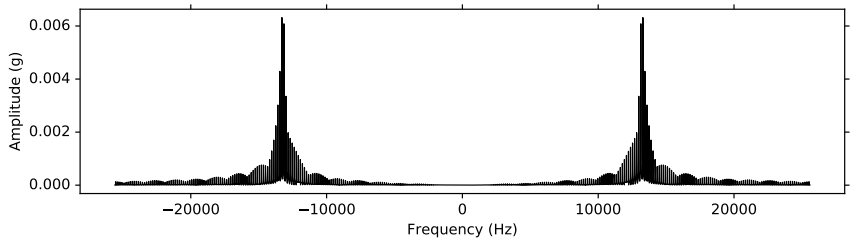
$$BPFO = \frac{1}{2}nf(1 - \frac{d}{D}\cos\alpha) \tag{3.5}$$

To detect this BPFO and its corresponding amplitude in the frequency spectrum, envelope detection is applied. The different steps of envelope detection are explained next together with illustrations of the results of each step on the simulated signal. The first step in envelope detection is a high-pass filter to separate the

superimposed signal from the low frequency signals. The high-pass filter's cutoff frequency is set to 1000 Hz. This cutoff frequency should be much lower than the resonance frequency. This is because the resonance frequency should not be filtered out. Additionally, there are sidebands around the resonance frequency as a result of outer-raceway fault, which also should not be filtered out. For the simulation of the signal in Figure 3.10, the resonance frequency is set to 13200 Hz. The cutoff frequency should also be higher than the rotation speed. As a result the bursts are isolated as can be seen in Figure 3.11a.



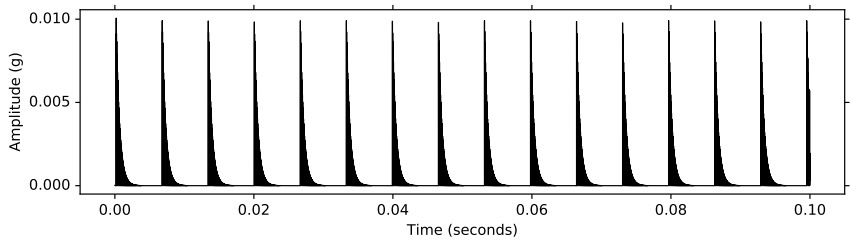
(a)



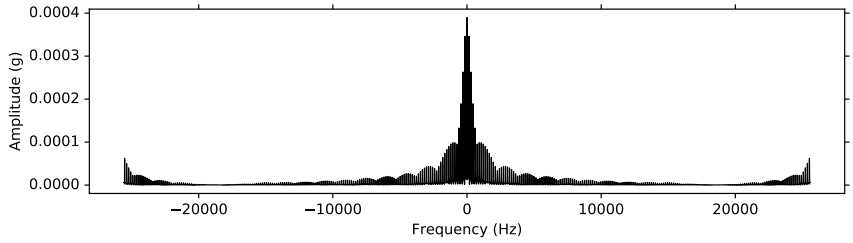
(b)

Figure 3.11: The signal after applying a high-pass filter (a) and its frequency spectrum (b).

The remaining signal (Figure 3.11a) can be seen as the result of amplitude modulation where the BPFO signal can be considered as the message and the resonance frequency as the carrier signal with frequency f_c [77]. Hence, the reason why in the frequency spectrum (Figure 3.11b), there are sidebands surrounding the resonance frequency (13200 Hz). To extract the BPFO signal in the next step, amplitude demodulation is done using the square-law demodulation technique [78]. The signal is squared, which results in a convolution of its frequency spectrum with itself [79]. This causes half of the energy to shift towards higher frequencies ($[-2f_c-2w; -2f_c+2w]$ and $[2f_c-2w; 2f_c+2w]$, where w is the cutoff frequency of the sidebands) and half the energy to lower frequencies ($[-2w; 2w]$) [80]. The result can be seen in Figure 3.12. Note that a part of the higher frequencies lie outside of the frequency spectrum shown in Figure 3.12b.



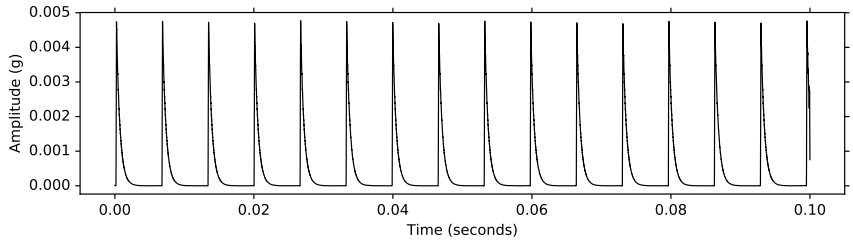
(a)



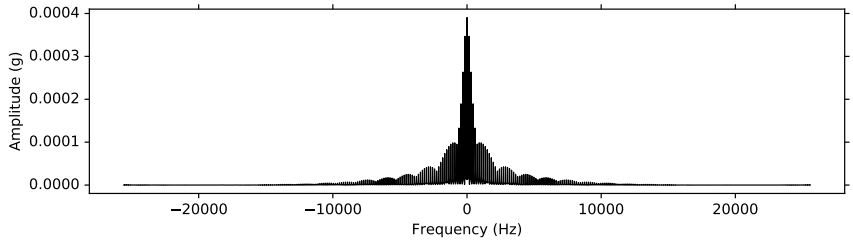
(b)

Figure 3.12: The signal after squaring it (a) and its frequency spectrum (b).

As the BPFO is a relatively low frequency compared to the carrier frequency (resonance frequency), we only have to keep the low frequencies. Hence, a low-pass filter is applied. The cutoff frequency should be larger than $2w$ and lower than $2f_c - 2w$, but as both w and f_c are not known in a real-life situation, the cutoff can also be determined visually. The cutoff frequency utilized here is 10000 Hz. The result of this can be seen in Figure 3.13.



(a)

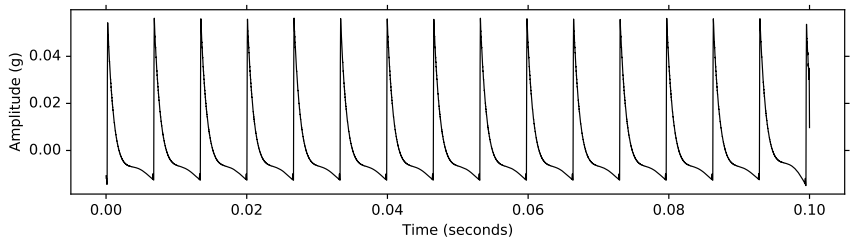


(b)

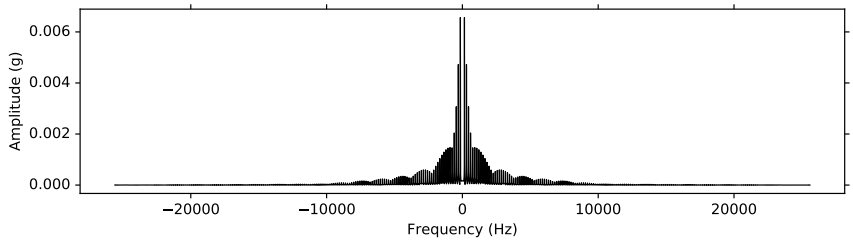
Figure 3.13: The signal after applying a lowpass filter (a) and its frequency spectrum (b).

This results in an envelope of the impacts. However, a DC component is still present in the signal due to the squaring and the signal is not in the scale of the original signal. Hence, first the square root is taken of the signal and subsequently a high-pass filter is applied. This high-pass filter is set to 100 hz as it should be higher than the DC component and lower than the BPFO. The result can be seen in Figure 3.14.

In the frequency spectrum of the envelope signal, the BPFO frequency will have the highest amplitude as can be seen in Figure 3.15. This frequency can be isolated better using a band-pass filter as is done for the actual measurements.



(a)



(b)

Figure 3.14: The signal after taking the square root and applying a high pass filter (a) and its frequency spectrum (b).

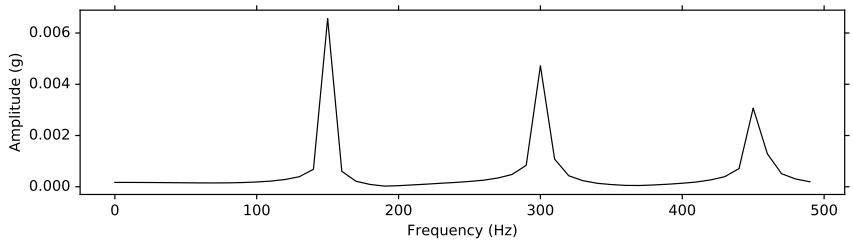
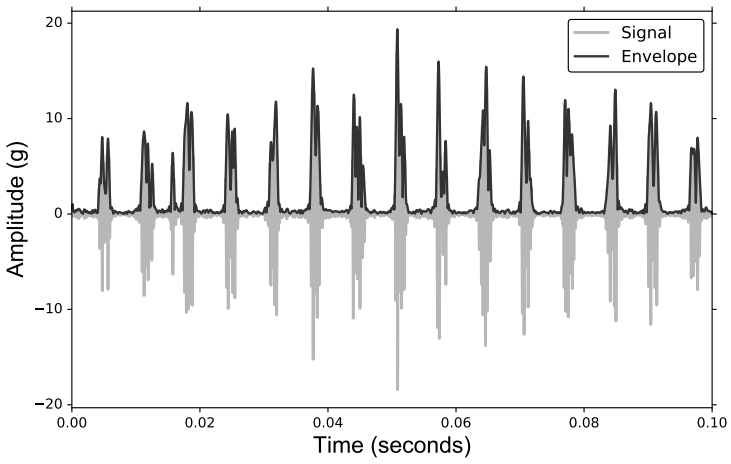


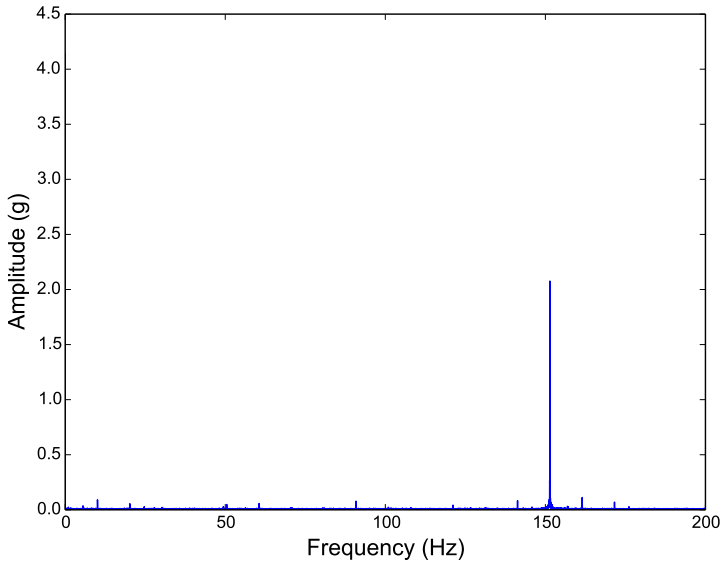
Figure 3.15: Part of the frequency spectrum of the envelope signal highlighting the BPFO.

To summarize, if the frequency of the envelope signal is near the calculated BPFO and has a high amplitude, it can be concluded that an outer-raceway fault is present. An example of this envelope signal determined on the measurements can be seen in Figure 3.16. From the envelope frequency, the maximum amplitude and the corresponding frequency are extracted as features. These two features are calculated for both vibrations signals (x-direction and y-direction per measurement).

All these features are extracted from overlapping windows on the vibration signals, resulting in 19 samples per measurement.



(a)



(b)

Figure 3.16: Vibration signal generated by an outer-raceway fault together with its frequency spectrum. (a) The signal together with the corresponding envelope. (b) Frequency spectrum of the envelope signal.

3.5.1.3 Classification method

Classifying between the different bearing conditions is a complex task as there are many features and the data is not linearly separable. Several classification algorithms are suitable for this task such, as a support vector machine, neural network or random forest classifier. In an empirical evaluation done by Caruana et al. [81] on 11 classification problems, it was shown that the random forest classifier outperforms support vector machines and neural networks. These results should be taken with a grain of salt, as many aspects of the data set can influence these results. However, they can be taken as a guideline. Hence, the random forest classifier is chosen here.

More information about the random forest algorithm for classification purposes can be read in Section A.5.1. It should also be noted that, as was presented in Chapter 2, random forests can provide insights into the data by providing feature importance, which also motivated the choice of classification algorithm.

The random forest classifier has several hyperparameters, however the amount of classification trees in the random forest is the most important hyperparameter. The more trees, the better the results, however at a certain point this will stop and the random forest can possibly start to overfit. Therefore, this hyperparameter was optimized using grid-search, which resulted in in an optimal number of trees which was 200.

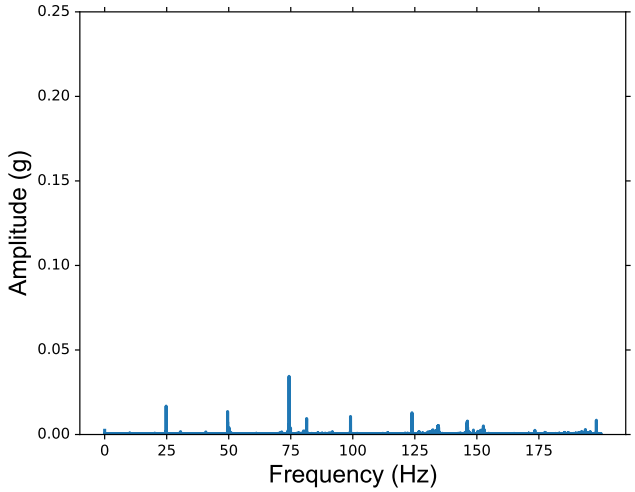
The conditions determined by this pipeline are combined with the discovered conditions in pipeline two as is discussed next.

3.5.2 Pipeline two

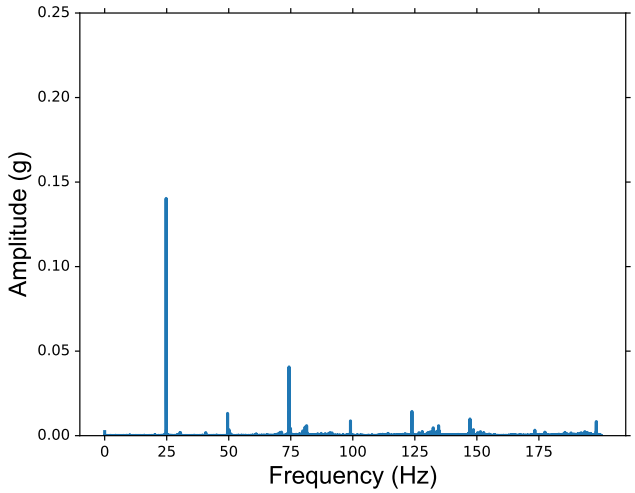
The goal of pipeline two is to determine if there is imbalance, and if so how much, regardless of the presence of a bearing fault.

3.5.2.1 Feature extraction

As discussed in the literature review, imbalance is detectable by observing if there is a high amplitude at the rotation frequency of the machine. The first step to extract the amplitude at the rotation frequency is windowing as was described for pipeline one. Per window, the discrete Fourier transform (DFT) of the signals is taken. An example of a frequency plot is given in Figure 3.17. As can be seen, a peak close to the rotation frequency (25 Hz) can be observed when there is imbalance. In the final step, a band-pass filter is applied to isolate the 25 Hz frequency. The amplitude corresponding to the rotation frequency is subsequently used as feature. This feature is extracted for the two vibration signals per window, resulting in 19 samples per test run, each containing two features. After these features are calculated, classification is applied.



(a)



(b)

Figure 3.17: Frequency plots during imbalance and balance. (a) Frequency spectrum of a bearing when the system is in balance. (b) Frequency spectrum of a bearing during imbalance. The peak at $\sim$ 25 Hz is clearly visible.

3.5.2.2 Classification method

The classification problem in data set one for this pipeline is easy as the distinction between balance and imbalance has to be made. This is a binary classification problem and we observe that the classes are also linearly separable. Hence, the most simple classification algorithm is chosen which is logistic regression. More info on logistic regression can be read in Section A.5.3. For data set two, the classification task is more difficult as a distinction between several gradations of imbalance has to be made. To solve this multiclass classification problem, the random forest algorithm is again chosen as classifier.

In the end the conditions of both pipelines are combined as depicted in Figures 3.8a and 3.8b, to provide the condition of the machine.

In the next section the infrared thermal image-based fault/condition detection system is discussed, where after the system discussed above is combined with the IRT-based system resulting in a multi-sensor system.

3.6 Methodology: single-sensor infrared thermal image-based fault detection

For the IRT-based system also a two-pipeline architecture is proposed similar to the architectures depicted in Figures 3.8a and 3.8b.

3.6.1 Pipeline one

As for the vibration-based system, pipeline one for the IRT-based system detects the different conditions of the bearings. For data set one, pipeline one detects if the bearing is properly, inadequately or extremely inadequately lubricated or contains an outer-raceway fault. For data set two, pipeline one detects if the grease of the bearing contains solid particles, if the bearing contains an outer-raceway fault or if the bearing is healthy. To do this task, pipeline one starts of with the preprocessing of the thermal videos. Afterwards, feature extraction and machine learning are applied.

3.6.1.1 Preprocessing

First, sub-sampling in time is applied, resulting in a frame rate of 1 fps. The ideal framerate is determined experimentally for optimal results. The framerate does not influence the results significantly. However, if too many frames are kept the data will become unmanageable. If too few frames are kept, windowing will not be possible. Windowing is required to average-out the effects of outliers. 19 windows per video are created. Subsequently, for every frame, all the pixel values are transformed to relative temperatures. This transformation is done by subtracting the

ambient temperature, measured by a thermocouple in the darkened room, from every temperature value measured by the thermal camera. Next, the foreground, i.e. the bearing housing which is relatively warm, needs to be isolated from the background, i.e. not-relevant colder part. This segmentation is done using a threshold determined by the Otsu algorithm [82]. If this is not done, the calculated features will be influenced by non-relevant elements in the background.

As the measurements were done over a time-span of weeks wherein the bearings had to be swapped and the camera stored, the infrared thermal camera moved and so did the set-up. As a result translations and rotations are induced in the data set. To enable robustness against these transformations, all the frames need to be aligned to a common reference image (i.e. image registration). By applying image registration, the bearing housing will be in the same position in every frame of every recording. Image registration is done using the following steps:

1. A random reference frame from a random recording is extracted. In the end, every frame should be aligned to this reference frame.

2. From the reference frame, but also from every first frame from every recording, key points are extracted. The key points are corners and are determined using the FAST algorithm [83].

3. Image patches are extracted around every key point for which ORB feature descriptors are calculated [84].

4. Matching key points extracted from the reference frame and a frame that needs to be aligned are selected. This is done by calculating the hamming distance which allows for very fast matching computation compared to for example the euclidean distance [85].

5. Using the matching key points from the reference frame and a frame that needs to be aligned, the affine transformation is determined using the RANSAC algorithm. The affine transformation describes how an image should be translated, rotated and scaled so that it will match the geometry of the reference frame. This transformation is calculated for every recording and applied to every frame, resulting in recordings wherein the housing is approximately in the same place.

As last step, a boxcar filter is applied on every frame to eliminate high frequency components in the image. The goal of this step is to reduce the influence of edges in the feature extraction step as these are not informative for fault detection. Note that all these preprocessing steps, i.e. temperature conversion, background subtraction, image registration and filtering, are required to make sure that conditions/faults are detected based on the heat in the component and not based on other factors. For example, if all recordings for condition A are taken on a hot day (i.e

high absolute temperature) and all recordings of condition B on a cold day (i.e. low absolute temperatures), then by just using the mean temperature the condition of the component can be determined. However, this is not correct as the ambient temperature is not related to the condition/fault. The same example holds true for the camera placement.

From the preprocessed frames features are extracted.

3.6.1.2 Feature extraction

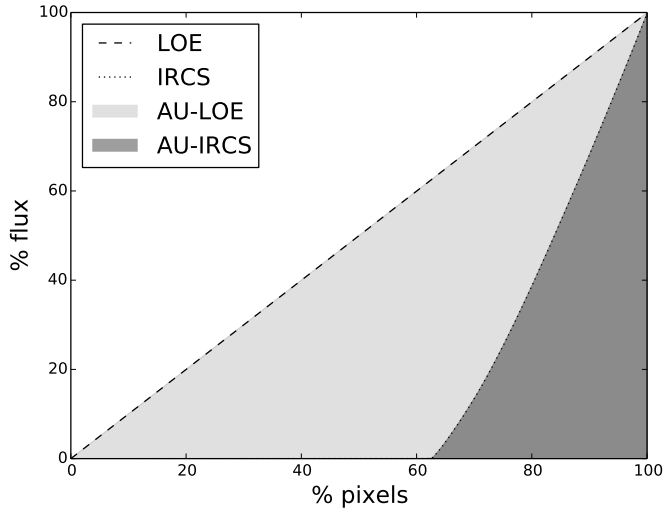
In total three features are extracted form the preprocessed frames. The literature overview indicates several usable and reproducible data-driven image processing features such as the central moments, standard deviation, mean, skewness, kurtosis, variance, maximum and minimum from the image histogram. For the specific faults considered here, only the standard deviation proves to be valuable. Therefore, the standard deviation of the pixel values is chosen as first feature (Equation 3.6 where μ is the mean and x_i the i-th sample). The standard deviation is a measure that is used to quantify the amount of variation or dispersion of a set of data values.

$$SD = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(x_i - \mu)^2} \tag{3.6}$$

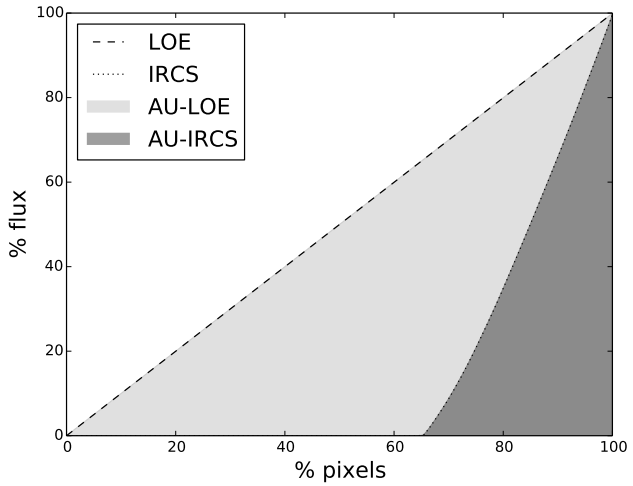
The second feature is the Gini coefficient (GC) which is often used in economics and astronomy to measure inequality [86]. As opposed to the standard deviation, the GC is not a metric based on central tendency, i.e. deviation from the mean, but a general measurement of dispersion. To calculate the GC, two cumulative distributions are required. The first is the cumulative distribution of the sorted pixel values of the IR image (IRCS). The second is the line of equality (LOE), which is a cumulative distribution of the image pixels values as if they were equal. Examples of these curves can be seen in Figure 3.18a and 3.18b. In these figures, the light shaded grey area represents the area under the LOE (AULOE) and the dark shaded gray area represents the area under the IRCS (AUIRCS). If the pixel values of the IR image are completely uniformly distributed, i.e. if all pixel values are equal, the AULOE is equal to the AUIRCS. The GC is directly related to the size of the area between the LOE and IRCS. Therefore, the GC is calculated according to Equation 3.7. If the GC is 0 % there is total equality and 100 % if there is total inequality.

$$\frac{\text{AULOE} \cap \text{AUIRCS}}{\text{AULOE}} \tag{3.7}$$

Both the SD and the GC measure dispersion, however, there is a fundamental difference between the two statistics. The SD will be maximal if half the samples



(a)



(b)

Figure 3.18: Graphical representation of the Gini Coefficient. (a) Example of a healthy bearing with Gini coefficient equal to 66.47 %. (b) Example of a MILB with Gini coefficient equal to 68.74 %. The Gini coefficient for the MILB case is 2.27 % higher compared to the HB case. Flux, i.e. heat flow, corresponds to the brightness of the pixels.

are at each bound, i.e. extremes of the range of temperatures, whereas the GC will be maximal if one sample is at one bound and all the rest at the other. This implies that if there is a very narrow hotspot wherein the temperatures are very high compared to the other temperatures, the GC will be very high, however, this will not cause the SD to be at its maximum. Conversely, if half of the pixels are at room temperature and the other half are significantly warmer, the SD will be at its maximum and the GC will not be abnormally high. This is due to the fact that there is still equality, as the amount of pixels at room temperature is the same as the amount of pixels that are warm.

An example from economy to better understand the difference between the GC and SD: if one person has almost all the wealth, the GC of the population will be maximal (the wealth is extremely unequally distributed). However the SD will not be maximal as one outlier will not influence the mean a lot, which is used in the calculation of the SD. Conversely, if 50 % of the population has all the wealth and the other 50 % nothing, the SD will be maximal. This is due to the fact that the two extremes are maximally removed from the mean. The GC on the other hand will not be maximal as the wealth is still quite equally distributed. The fact that the two metrics behave differently is mainly due to the fact that the SD is a measurement of central tendency

We observe that when the bearing is mildly inadequately lubricated and there is no grease reservoir, the temperature of the seal of the housing will be high. Due to the fact that the heat is concentrated on a small area, the GC will be closer to 100 %. However, this is not immediately noticeable when the bearing is extremely inadequately lubricated. This is due to the fact that also a part of the housing near the bearing will be noticeably hot, not resulting in a significantly larger GC. Also, when the bearing is healthy, the GC will not be noticeably high. The SD will be high for the conditions when there is a grease reservoir due to the fact that there are two large groups of pixels, those that are heated up (the housing close to the bearing) and those that are not heated up (e.g. bolts, bottom of the housing). Conversely, the SD will be lower when there is no grease reservoir.

To summarize, the GC proves to be useful to detect the different levels of lubrication inadequacy when no grease reservoir is used as can be observed in Figure 3.19. In this figure there is clearly no overlap between the EILB samples and the MILB samples. The SD is useful to make the distinction between a bearing housing containing a grease reservoir and without a grease reservoir. However, there is no complete separation between the two conditions in the features space, hence a third feature is added, which is called the Moment of Light.

The Moment of Light, is a measurement of concentration related to the spatial temperature distribution and is also often used in astronomy [86]. The Moment of Light is also called the second-order moment of the pixels collectively containing 20 % of the brightest pixels (M_{20}). M_{20} is calculated according to Equation 3.8,

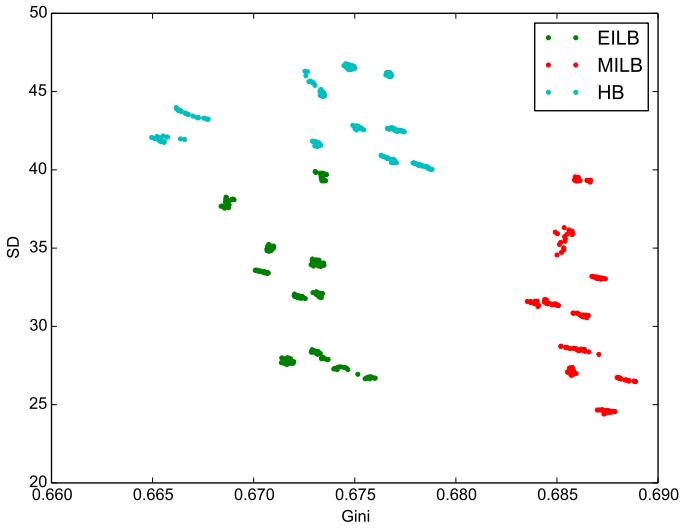


Figure 3.19: Scatter plot of the GC and the SD. No overlap exist between the EILB and MILB samples.

where M_i is the value of each pixel f_i multiplied by the squared distance to the hottest/brightest point (Equation 3.9). For the denominator of Equation 3.8, M_i is summed over the set of all the pixels (denoted as I).

The most interesting property of the M_{20} is that it will be more negative if there is only one distribution of significant size, and higher (i.e. close to zero) when there are two or more distributions of significant size. Significant size means that it is taken into account in the nominator of Equation 3.8 due to the fact that it contains points that are a part of the 20 % of the highest points. More information about this property can be found in Appendix E.

The following phenomena have been observed: (1) if a grease reservoir is present, which is the case for HBs(-IM) and ORFs(-IM), the hottest point will generally be on top of the housing closest to the bearing; (2) If there is no grease reservoir, as for MILBs (-IM) and EILB (-IM), the hottest point will be at the seal near the shaft, due to friction between the seal and shaft;(3) most of the 20 % of the brightest pixels are usually situated on top of the housing for this set-up as this is always a hot area due to its close proximity to the bearing. As a result, there is only one hotspot of significant size when a grease reservoir is present and the distance between the hottest pixels and the hottest points will be small when a grease reservoir is present, resulting in a small nominator of Equation 3.8 and a low M_{20} score. On the other hand, when no grease reservoir is present, there will

be two hotspots of significant size. Hence, the nominator will be much larger and the M_{20} will be higher (i.e. closer to 0) [2]

$$M_{20} = \log_{10}\left(\frac{\sum_{\forall \in I_{20}} M_i}{\sum_{\forall \in I} M_i}\right) \tag{3.8}$$

$$M_i = f_i \cdot ((x_i - x_c)^2 + (y_i - y_c)^2) \tag{3.9}$$

Since the M_{20} exhibits properties related to the presence of a grease reservoir, it is very useful to help indicate if a grease reservoir is present or not. Furthermore, if the M_{20} and the SD are combined, a clear distinction can be made between the samples created when a grease reservoir was present, i.e. HB and ORF, and the samples without a grease reservoir, i.e. EILB and MILB as depicted in Figure 3.20. In this figure it can be observed that the samples with and without a grease reservoir do not overlap and are clearly separated.

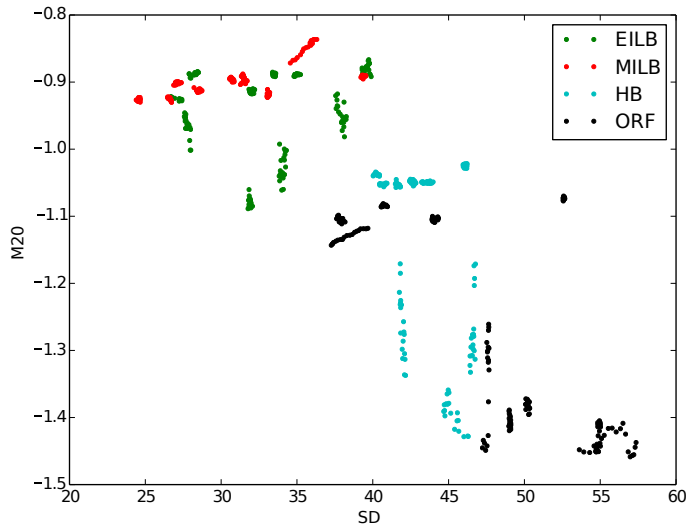


Figure 3.20: SD and M20 of the samples. Two non-overlapping clusters can be observed. One cluster containing the EILB and MILB samples for which no grease reservoir was present and another cluster containing the HB and ORF samples for which a grease reservoir was present.

Using these three features most of the conditions can be detected, however as Figure 3.21 illustrates, there is still some overlap between the ORF and HB classes for data set one. This is also the case for data set two as can be seen in Figure 3.23

[2] Lotz et al. [86] observed this fact also as they observed that the M_{20} is sensitive to mergers of galaxies (i.e. multiple galaxy nuclei).

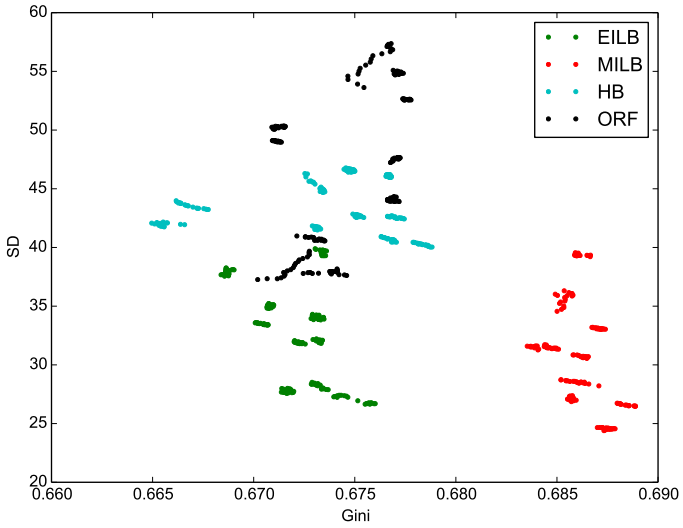
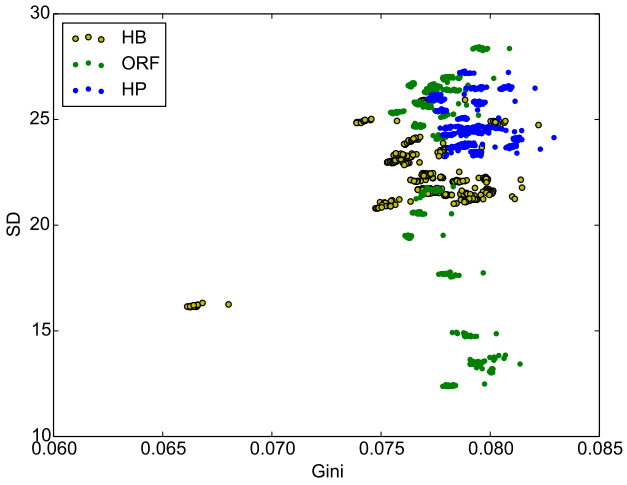
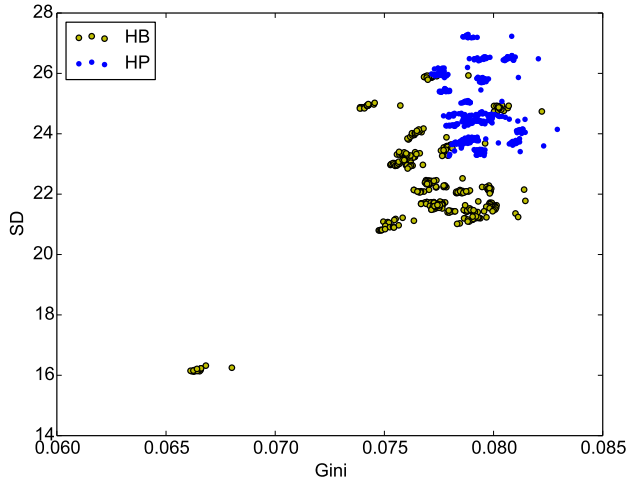


Figure 3.22: 2D plots of the GC and the SD. Some overlap exists between the ORF and HB samples.



(a)



(b)

Figure 3.23: (a) Scatter plot of the healthy bearings samples, the bearings with an outer-raceway fault and bearings with hard particle contamination. (b) Scatter plot of the samples of the healthy bearings and the bearings with hard particle contamination.

3.6.1.3 Classification method

Similar to the single-sensor system that uses vibration signals, here too a random forest classifier is used because there are multiple classes which are not linearly separable. However, only 100 decision trees were required in the random forest to perform optimally.

3.6.2 Pipeline two

The goal of pipeline two is to detect and quantify imbalance. To do this, as in pipeline one, first the videos are preprocessed.

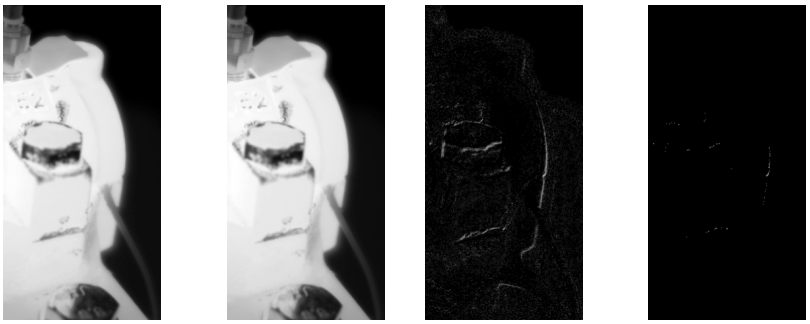
3.6.2.1 Preprocessing

As opposed to the preprocessing in pipeline one, in this pipeline no image registration or temperature correction is done. However, an additional step is required in this pipeline.

Due to imbalance the set-up will vibrate. Although, this vibration is very small, it can be seen in the video. In Appendix D the relationship between the amount of imbalance and the observable displacement in the IRT video is described analytically and quantified (which is not common knowledge). This observerable displacement can be isolated by differencing two consecutive frames. For example when subtracting Figure 3.24a from Figure 3.24b, Figure 3.24c is obtained showing the outline of the housing as a result of the movement due to imbalance. A threshold is additionally applied to remove background noise in the differenced image, as is displayed in Figure 3.24d. The optimal threshold value is empirically determined. Note that only the right side of the image is used as the movement of the shaft, i.e. rotation, is not related to the imbalance condition. Hence, if the shaft is included in the image lower classification performance is achieved.

3.6.2.2 Feature extraction

To detect the degree of imbalance, the difference between frames needs to be quantified resulting in features for the classification step. In order to do so, for each differenced frame, along each column of pixels, the sum of the pixels is taken. Similarly, along each row of pixels, the sum of the pixels is taken. The operation on the rows can be expressed as $\mathbf{x} = \mathbf{j}\mathbf{X}^T$ and the operation on the columns can be seen in expressed as $\mathbf{y} = \mathbf{i}\mathbf{X}$; where $\mathbf{X} \in \mathbb{R}^{N\text{x}M}$; $\mathbf{i} = [1, 1, ..., 1] \in \mathbb{R}^{1\text{x}N}$; $\mathbf{j} = [1, 1, ..., 1] \in \mathbb{R}^{1\text{x}M}$ and thus $\mathbf{x} \in \mathbb{R}^{1\text{x}N}$ and $\mathbf{y} \in \mathbb{R}^{1\text{x}M}$. Examples of the output vectors displayed in bar charts can be seen in Figure 3.25a and 3.25b. As can be seen, the width of these bar charts indicates the amount of movement in the differenced frame. Generally, if there is imbalance, the width will be larger.



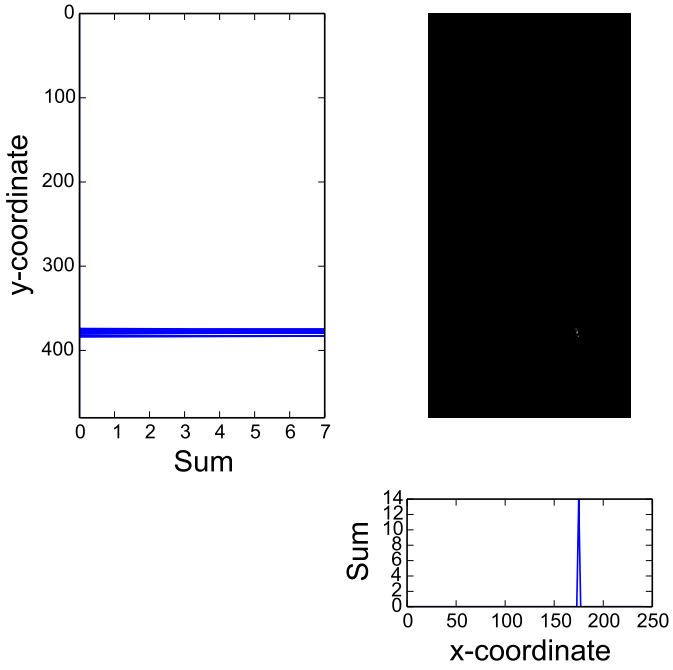
(a) *(b)* *(c)* *(d)*

Figure 3.24: Movement detection. Parts of consecutive frames (a)(b) are differenced resulting in (c). Afterwards a threshold is applied (d).

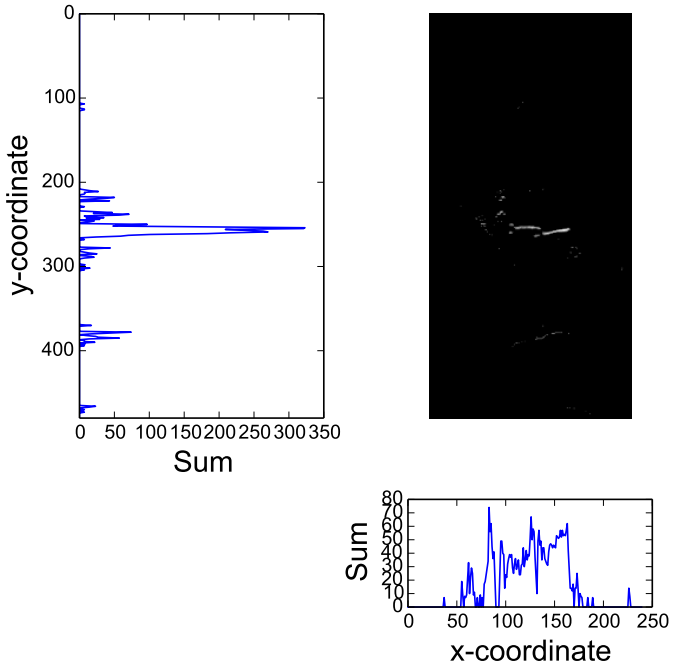
Hence, the standard deviation (SD), calculated on the output vectors, is chosen as feature. Also, the kurtosis is used as it describes how peaked the bar charts are.

When the SDs of the sum vectors along both axis, i.e. rows and columns, are plotted for data set two, four regions corresponding to the four balance/imbalance conditions can be observed (Figure 3.26a). It can be stated that the higher the SDs, the more imbalance is present in the system. Nevertheless, the clusters of points do partially overlap. Generally, a classifier will already be able to make a good distinction between the different levels of imbalance using the SDs, however, by adding the kurtosis the classifier's performance improves.

Figure 3.27 illustrates the preprocessing and feature extraction steps for this pipeline.

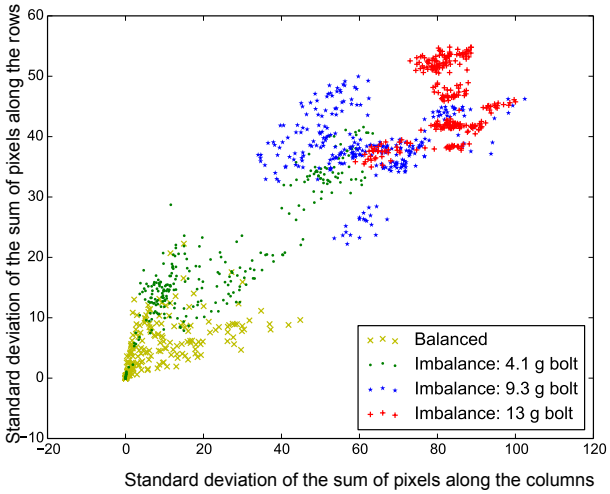


(a)

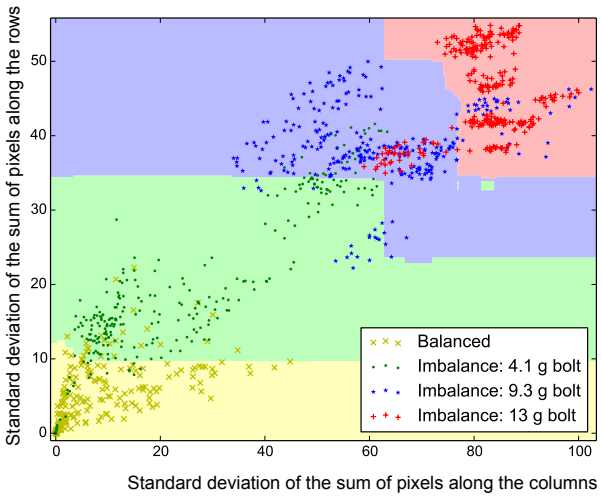


(b)

Figure 3.25: Examples of the bar charts representing the sums along the x axis and y axis for a healthy bearing when in balance (a) and imbalance (b).



(a)



(b)

Figure 3.26: (a) Scatter plot of the standard deviations along the image axis for the different levels of imbalance. (b) When these samples are classified, clear, logical regions are noticeable according to the level of imbalance.

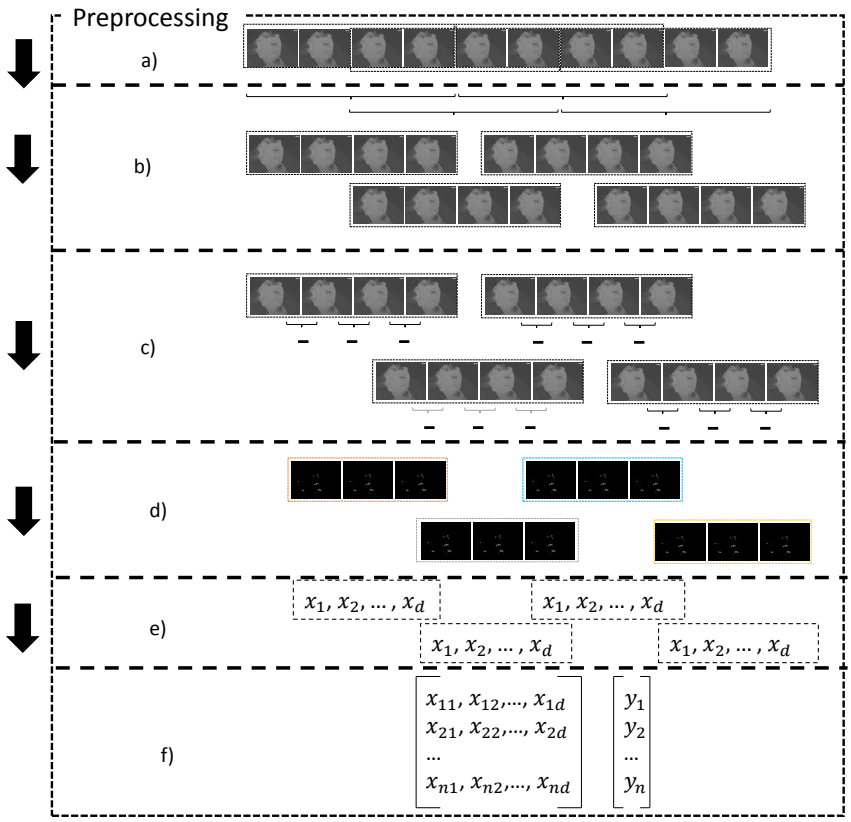


Figure 3.27: Preprocessing and feature extraction on the infrared videos (a) for imbalance detection. In (b), windowing is applied. Next, frame differencing (c) is done resulting in (d). Finally, per differenced frames features are extracted which are aggregated per window. The final result is a design matrix (f) which is ready to be used in the classification step.

3.6.2.3 Classification method

As in pipeline one, a random forest classifier is also used for pipeline two. When plotting the decision boundaries that the random forest determined, the regions which correspond to the level of imbalance, become visible (Figure 3.26b).

3.7 Methodology: Multi-sensor fault detection

Detecting the many different conditions of the component is a difficult task. Hence, in order to accurately detect these conditions, the features extracted from the two types of sensor data (i.e. IRT and vibrations) are used together in a multi-sensor

system. The features listed in Table 3.5 are used for the respective pipelines and data sets. To classify the samples, random forest classifiers are used.

Data set	Pipeline	Vibration features	IRT features
One	One	Amplitude at the BPFO Amplitude at the rotation frequency	GC SD M20
One	Two	Amplitude at the rotation frequency	SD along the y-axis
Two	One	Detected BPFO frequency RMS Kurtosis Crest Amplitude at the rotation frequency	SD
Two	Two	Amplitude at the rotation frequency	SD along the y-axis

Table 3.5: Features used for the multi-sensor system for both data sets and pipelines. Note that pipeline one detects bearing conditions and pipeline two imbalance gradation.

3.8 Results

In this section the results for the two single-sensor fault/condition detection systems are discussed and also the results of the multi-sensor system. First, the evaluation procedure is presented. The evaluation metric used is accuracy but also confusion matrices.

To objectively evaluate the performance of the classifiers, leave-one-bearing-out cross-validation is used as depicted in Figure 3.28 for data set one. Hence, from the 40 recordings, 32 recordings, i.e. 608 samples, are used to train the random forest classifier, and 8 recordings, i.e. one recording for each condition from a single bearing resulting in 152 samples, are used to test the classifier. This procedure is done five times so that every bearing is used once for testing, assuring that the system provides a generalized solution.

As the goal is to classify a recording according to the different conditions, and 19 samples per recording are classified, a majority vote is done per recording when testing. For the balance-imbalance case this means that whenever (for a single recording), more than 10 of the 19 samples are classified according to imbalance, the recording is classified as imbalance, and vice versa.

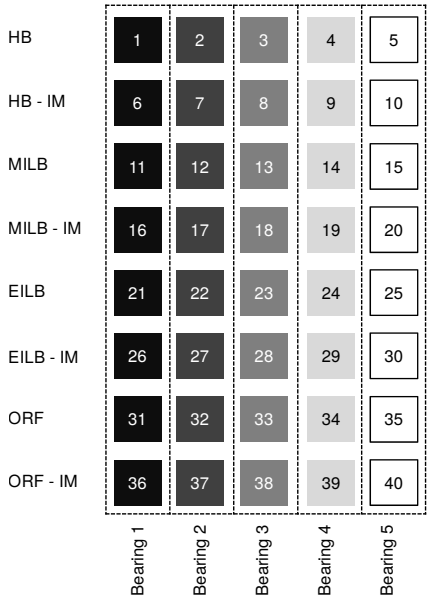


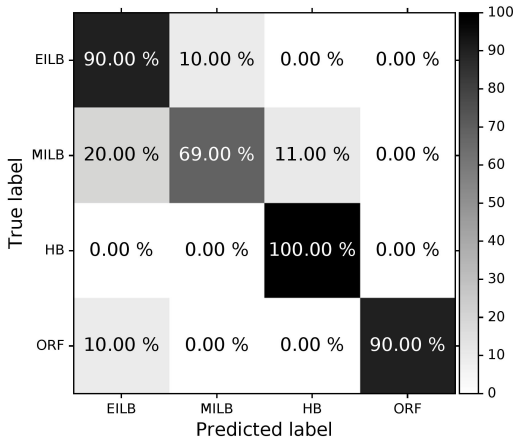
Figure 3.28: Leave-one-bearing-out cross-validation represented visually. Every test iteration all samples of one bearing are used as test set and the remaining samples as training set.

3.8.0.4 Results: data set one

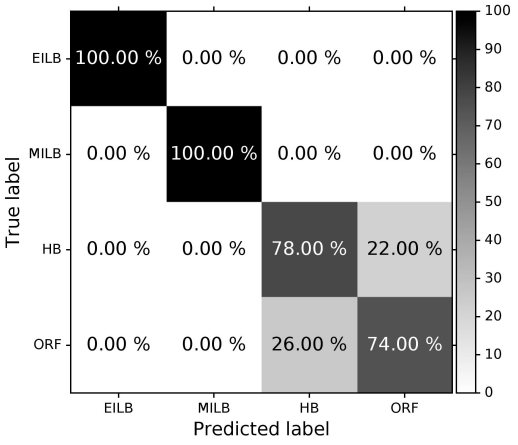
The results for the two single sensor systems and the multi-sensor system can be seen in Table 3.6. Detecting if the machine suffers from imbalance or not can be done perfectly using both the single-sensor systems and the multi-sensor system. However, detecting the specific bearing condition (EILB, MILB, ORF or HB) is more difficult. When observing the confusion matrix for the single-sensor IRT based system (Figure 3.29b) and the vibration based system (Figure 3.29a), it can be concluded that the IRT-based system has difficulty detecting outer-raceway faults. Conversely, the vibration-based system has more difficulty detecting lubrication related conditions. Both single-sensor system exhibit a specific weakness that can be nullified by the strengths of the other sensor. Hence, combining them in a multi-sensor solution results in a better overall system (Figure 3.29c). Furthermore, there is an additional general improvement in all classes due to the interaction of features in the machine learning algorithms on top of eliminating the weaknesses of the respectively single sensor solutions. This way, for example, outer-raceway faults can perfectly be classified (100 % accuracy), while at most 90 % accuracy could be achieved using solely vibration data, respectively 74 % using only IRT data. Using a multi-sensor solution, the classification task becomes easier as more information is available resulting in a perfect classification results.

IR	VIB	Conditions	Accuracy
IR		MILB, EILB, HB, ORF	88.25 % ($\sigma = 8.07$ %)
	VIB	MILB, EILB, HB, ORF	87.25 % ($\sigma = 8.10$ %)
IR	VIB	MILB, EILB, HB, ORF	100.00 % ($\sigma = 0.00$ %)
IR		balance and imbalance	100.0 % ($\sigma = 0.00$ %)
	VIB	balance and imbalance	100.0 % ($\sigma = 0.00$ %)
IR	VIB	balance and imbalance	100.0 % ($\sigma = 0.00$ %)
IR		All 8 conditions	88.25 % ($\sigma = 8.07$ %)
	VIB	All 8 conditions	87.25 % ($\sigma = 8.10$ %)
IR	VIB	All 8 conditions	100.0 % ($\sigma = 0.00$ %)

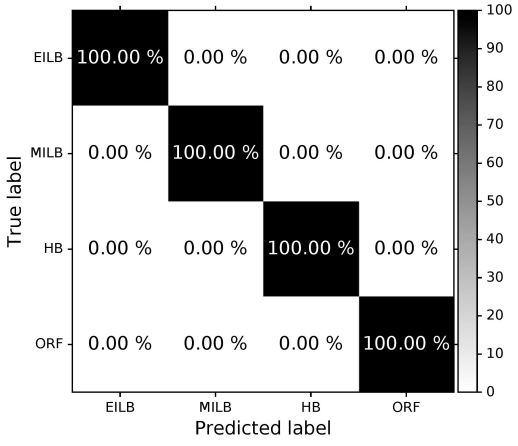
Table 3.6: Classification results for data set one.



(a)



(b)



(c)

Figure 3.29: Confusion matrices of the (a) vibration-based fault detection system, (b) the IRT-based fault detection system and the (c) multi-sensor fault detection system for data set one.

By using a random forest classifier it is possible to extract how important each feature is to the model [87], which is listed in Table 3.7 for the multi-sensor system. The table illustrates that the amplitude at the rotation frequency is not very important to the model, which is to be expected since little improvement is gained when adding it to the model (2.5 % accuracy) and it is not directly related to a specific condition. On the other hand, the amplitude at the ball pass frequency of the outer raceway is very important to the model and adding this feature to the IRT features results in an accuracy gain of 9.25 %. In total, by using the strengths of both sensors, the accuracy rises by 11.75 % compared to an IRT-based system and 12.75 % compared to a vibration-based system.

Metric	**Multi-sensor based**
Gini Coefficient	29.41 % ($\sigma = 1.07$ %)
SD	22.22 % ($\sigma = 3.84$%)
$\mathbf{M_{20}}$	20.29 % ($\sigma = 3.26$ %)
Amplitude at the BPFO	25.63 % ($\sigma = 1.82$ %)
Amplitude at the rotation frequency	2.45 % ($\sigma = 0.40$ %)

Table 3.7: Feature importance in the final multi-sensor system according to the random forest classifier.

3.8.0.5 Results: data set two

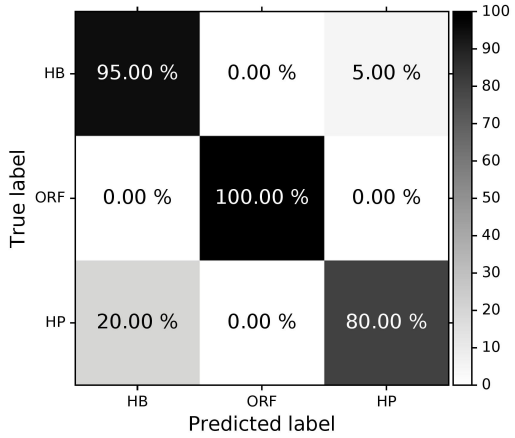
The results of the two single-sensor system and the multi-sensor system can be seen in Table 3.8. For pipeline two a relatively low accuracy is achieved due to the overlap between the ORF and HB samples using IRT features. As the confusion matrix in Figure 3.30a illustrates, the ORF class is hard to detect. This was also noticeable for data set one. Conversely, vibration-based bearing condition detection achieves a very high accuracy, and the classes do not get confused very often (Figure 3.30a). When features from both systems are combined, the three bearing conditions can be detected perfectly (Figure 3.30c).

IR	VIB	Conditions	Accuracy
IR		HP, ORF, HB	65.00 % ($\sigma = 16.16$ %)
	VIB	HP, ORF, HB	91.67 % ($\sigma = 12.91$ %)
IR	VIB	HP, ORF, HB	100.00 % ($\sigma = 0.00$ %)
IR		Imbalance gradation	88.33 % ($\sigma = 12.47$ %)
	VIB	Imbalance gradation	75.00 % ($\sigma = 9.13$ %)
IR	VIB	Imbalance gradation	90.00 % ($\sigma = 6.24$ %)
IR		All 12 conditions	55.00 % ($\sigma = 11.31$ %)
	VIB	All 12 conditions	66.67 % ($\sigma = 21.08$ %)
IR	VIB	All 12 conditions	90.00 % ($\sigma = 6.24$ %)

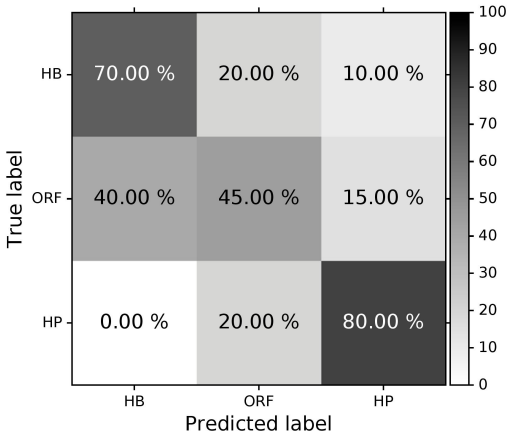
Table 3.8: Classification results for data set two.

The detection of the amount of imbalance is more difficult. Generally, the IRT-based system seems to outperform the vibration-based system. However, when both features from both modalities are used, the overall accuracy increases. The confusion matrices in Figure 3.31 indicate that the classifier can confuse a certain imbalance condition with another imbalance condition for which the weight difference is small.

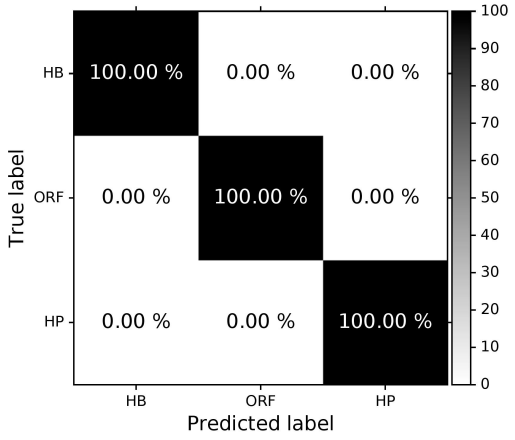
Pipeline one detects the different amounts of imbalance and pipeline two detects the specific REB conditions. In total there are 12 conditions. When solely using IRT data the accuracy score is 55.00 % ($\sigma = 11.31$ %) and when solely using vibration data the accuracy score is 66.67 % ($\sigma = 21.08$ %). However, when both modalities are combined there is a major improvement as the multi-sensor system achieves an accuracy score of 90.00 % ($\sigma = 6.24$ %). When observing the overall confusion matrix (Figure 3.32), it can be seen that when imbalance occurs together with contamination in the lubricant, it is more difficult to detect the conditions.



(a)

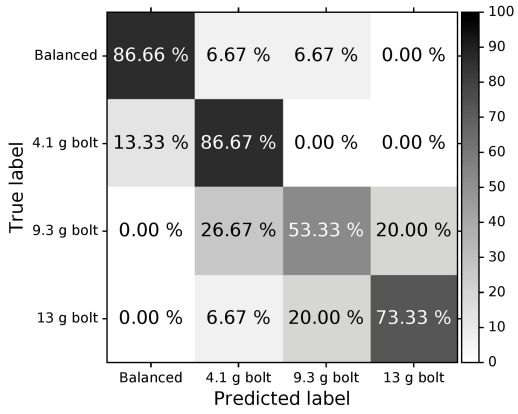


(b)

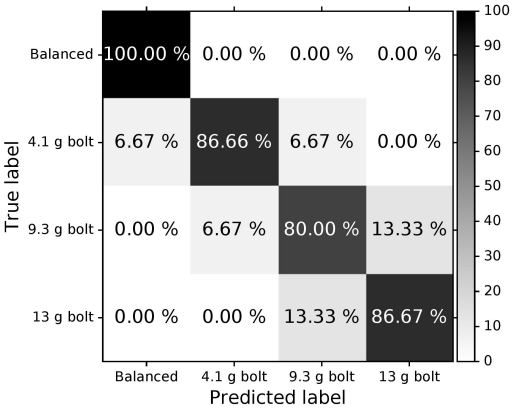


(c)

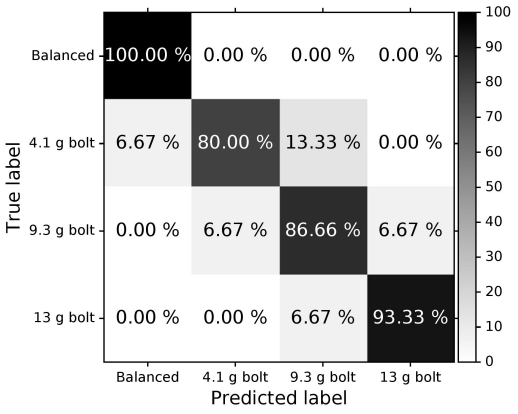
Figure 3.30: Confusion matrices of the (a) vibration-based fault detection system, (b) the IRT-based fault detection system and the (c) multi-sensor fault detection system for data set two.



(a)



(b)



(c)

Figure 3.31: Confusion matrices for (a) vibration-based imbalance detection, (b) the IRT-based imbalance detection and (c) multi-sensor based imbalance detection for data set two.

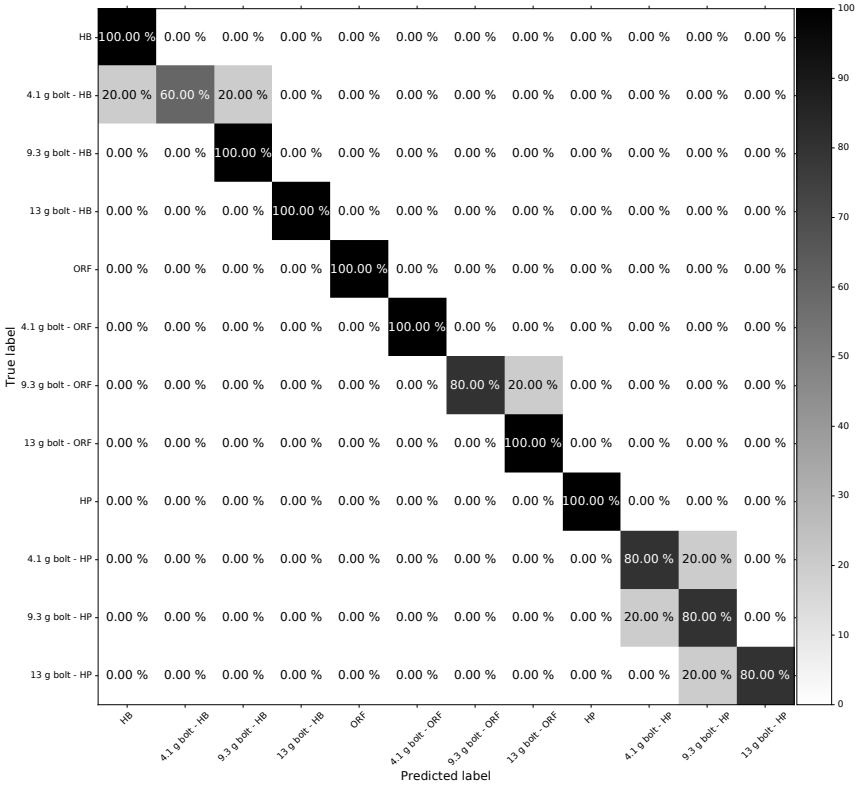


Figure 3.32: Confusion matrix of the multi-sensor system for data set 2 for all 12 condition.

3.9 Discussion

In this discussion section we reflect on three aspects of this chapter. The first is the generalizability of the proposed system. Second, the choice of classifier and third the majority voting approach.

3.9.1 Generalizability

The systems presented in this chapter work well. However, the system trained on data gathered from this setup will not perform well for a completely different set-up. For example, if the bearing type is changed, the BPFO will most likely be different. Hence, the bandpass filter to isolate the BPFO will have to be modified, and the classifier will have to be retrained. Similar for the infrared thermal data, new data will have to be gathered and the classifiers will have to be retrained.

Nevertheless, the methodology proposed in this chapter is re-useable. To illustrate this fact, in Appendix C, the methodology to create the IRT-based system is employed in a use case where a much larger bearing is used and a re-circulatory lubrication system is present in the set-up. Although this set-up is different compared to the set-up described in this chapter, using the methodology of this chapter and IRT imaging to create an automatic oil-level detection system, very good results can be achieved. In order to do so, the machine learning models were retrained on data from this new set-up. Note that, in general, potentially also new features will have to be designed to improve results.

3.9.2 Choice of classifier

There are many possible classification algorithms that can be used. However, the random decision forest algorithm is used almost exclusively in this chapter. This is due to the fact that random forests perform very well in general [81]. Besides the fact that a random forest is a very well performing algorithm, it has several other advantages. The first is that almost no preprocessing is required. The features do not have to be rescaled and the mean does not have to be removed. A random forest can deal with both discrete and continuous variables. The algorithm can handle a lot of data as the individual trees are trained on subsets of the data which can be done in parallel on different machines. Furthermore, random forests seldom overfit in practice, which is not the case for support vector machines and neural networks. Random forest will only tend to overfit if too many trees are used. Hence, this is the most important hyperparameter to tune. Other hyperparameters such as the depth of individual trees, or subset size, or subset of features to choose from when constructing individual trees seldom have a large impact on the random forest's performance. Hence, random forests are very robust classifiers, which are very easy to use. Compared to for example neural networks and support vector

machines, random forests have a much lower entrance barrier to use. When using neural networks hyperparameters such as the number of layers, number of nodes per layer, activation function and loss function have to be optimized, which can be a lot of work. When using a support vector machine a suitable kernel has to be chosen and subsequently the hyperparameters of the kernel have to be optimized, which is also considerable more work compared to the work required for using a random forest.

In general, from our experience it can also be concluded that when using a feature engineering based approach, the chosen features have the most impact on the end results compared to the classification algorithm. If the features are badly chosen/created, no algorithm will be able to achieve a good score. However, if the features are well designed, the performance difference between algorithms such as random forests, neural networks, support vector machines or K-nearest neighbors classification will often be small.

3.9.3 Majority voting

For every signal windowing is applied. In the end, every signal has to be classified based on the extracted windows. To do this, every signal in a window is classified, and the results of this classification step are used in a majority voting process. This means that if condition A is detected in the majority of the extracted windows from the original signal, the entire signal is assigned the label of condition A. When dealing with a binary classification problem (as for example is the case when detecting imbalance in data set one) this majority voting is in fact a check to see which label has more than 50 % of the "votes". Hence, in this discussion we want to make the reader aware that this threshold can also be optimized depending on the fault detection requirements/needs, or operator preferences. If we want to make sure that even if a few of the windows are classified as condition A, an alarm should be triggered, then this threshold value should be lowered. This can however result in more false positive classifications. Conversely, if we want to make sure that the entire signal is classified as condition A before assigning the label of condition A, then this threshold has to be increased. This however can results in more false negative classifications.

3.10 Conclusion

In this chapter, fault detection on component level is investigated. Instead of solely identifying if the machine is not performing as expected, the actual faults are identified. By identifying specific faults and conditions more efficient maintenance is possible. However, additional sensors are required to enable this. To be able to detect a multitude of conditions, a novel multi-sensor system is proposed that

uses both vibration signals and infrared thermal video. Two data sets are created containing conditions that have not yet been researched previously using thermal imaging, such as reduced lubrication and hard particle contamination.

For both types of data, specific features are engineered enabling the system to detect the different conditions. Features extracted from the vibration signal are physics-based features and statistical features. For the thermal infrared data, statistical features are used. We introduce two new features that are valuable for bearing fault detection using IRT images i.e. Gini coefficient and Moment of Light. For imbalance detection we illustrate that the machine's vibrations can be monitored using a camera. Hence, the displacement is used for imbalance detection. Overall, by combining features from the vibration measurements together with features from the IRT video, a multitude of conditions can be identified more accurately compared to when a single-sensor system is used. As the proposed approach can detect many conditions very accurately, it is possible to prevent fault escalation more often.

Based on our experiments and data sets, we can conclude that when measurements from different sensors are combined, improvements can be expected. Improvement is a logical result due to the fact that more information about the conditions become available as a certain aspects of the conditions and faults are only measurable with certain sensors types.

The main disadvantage of the proposed system is that it requires expert knowledge to create suitable features for the specific conditions and faults to be detected. This means that only features can be created for faults which are well understood. Furthermore, if a new fault has to be detectable, a new feature will possibly have to be created. A solution for this problem is presented in the next chapter where features will be learned by the machine learning algorithm.

The work presented in this chapter has lead to the following publications, or are submitted as follows:

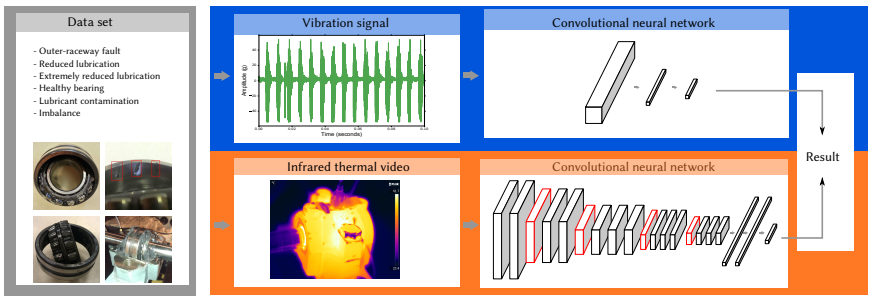
- Olivier Janssens, Raiko Schulz, Viktor Slavkovikj, Kurt Stockman, Mia Loccufier, Rik Van de Walle and Sofie Van Hoecke, *Thermal Image Based Fault Diagnosis for Rotating Machinery*, Infrared Physics & Technology, vol. 73, pp. 78-78
- Olivier Janssens, Mia Loccufier, Rik Van de Walle and Sofie Van Hoecke, *Data-Driven Imbalance and Hard Particle Detection in Rotating Machinery Using Infrared Thermal Imaging*, Infrared Physics & Technology, Vol.82, pp. 28-39
- Olivier Janssens, Mia Loccufier, Rik Van de Walle, Sofie Van hoecke, *ViTIS: Vibration and Thermal Imaging Based Multi-Sensor Fault Detection for Rotating Machinery*, ISA Transactions, **under review**

- Olivier Janssens, Lothar Verledens, Raiko Schulz, Veerle Ongenae, Kurt Stockman, Mia Loccufier, Rik Van de Walle and Sofie Van Hoecke, *Infrared and vibration based bearing fault detection using neural networks*, Proceedings of the 13th International Workshop on Advanced Infrared Technology and Applications, Proceedings, pp. 220-223
- Sofie Van Hoecke, Olivier Janssens, Raiko Schulz, Kurt Stockman, Mia Loccufier and Rik Van de Walle, Towards thermal imaging based condition monitoring in offshore wind turbines, Proceedings of the 13th International Workshop on Advanced Infrared Technology and Applications, Proceedings, pp. 291-295

Chapter 4

Component level: Multi-sensor fault detection using feature learning

Chapter highlights



In the previous chapter, fault detection systems were proposed for which the features were engineered either by expert knowledge or driven by the data. In this chapter, we propose feature learning systems for fault detection wherefore no features are designed manually. For both the thermal infrared imaging data and the vibration data, convolutional neural networks are researched and developed which are end-to-end machine learning systems. We show that for most faults in our data sets, the proposed feature learning systems outperform the feature engineering systems. We also show that transfer learning from natural images to infrared thermal images is feasible. Finally, using the convolutional neural networks, we illustrate that insights can be extracted from the infrared thermal data.

4.1 Introduction

Feature engineering is mainly used for automatic (component-level) fault detection and can be data-driven or model-driven. Data-driven feature engineering entails creating features that describe measurable characteristics that can be related to a condition by observation. How these characteristics occur as a result of the underlying physics is not taken into account. Model-driven feature-engineering requires reasoning based on the underlying physics of the conditions to deduce what resulting phenomena can occur and how to extract them from the measurements and quantify them in features.

In the end, whether the features are data-driven and/or model-driven, to automatically determine what fault/condition a component is suffering from, classification algorithms (i.e. machine learning) are required. It has been demonstrated in the previous chapter that by using feature engineering and machine learning, very well performing fault detection systems can be created using vibration data or/and IRT imaging data. However, feature engineering requires domain knowledge, data processing effort and feature engineering time as complex machines contain many components that can have many conditions for which characteristics are measured using different sensors. On that of that, by manually engineering features, there is a possibility that the system will not perform optimally as the feature engineering depends on (one or more) expert(s) with knowledge about the components and mechanical engineering or statistics. These experts might not be able to devise features that fully describe the dynamics of the signals that are required for correct classification as not all possible faults are yet completely understood. Furthermore, a fault/condition detection system can be designed for a specific set of conditions of the components, hence, when new conditions should be detectable, an expert possibly has to implement new feature extraction capabilities into the system.

In order to circumvent this problem, feature learning/representation learning can be employed. As opposed to feature engineering, wherein a human creates the features, feature learning uses a machine learning algorithm to learn and create useful features from raw data. The algorithm learns features that optimally represent the raw data for the required task. Feature learning, in essence, enables end-to-end machine learning systems.

Beside the fact that feature engineering is fundamentally different from feature learning, it should be noted that feature learning is also different from feature selection. Feature selection is used to select the most informative subset of features from all the available (manually engineered) features. This means that there is no feature learning during feature selection. A schematic representation of the difference between feature engineering, feature engineering in combination with feature selection and feature learning can be seen in Figure 4.1. In the feature engineering part of the figure it can be seen that from the input data ($\mathbf{X}$) features

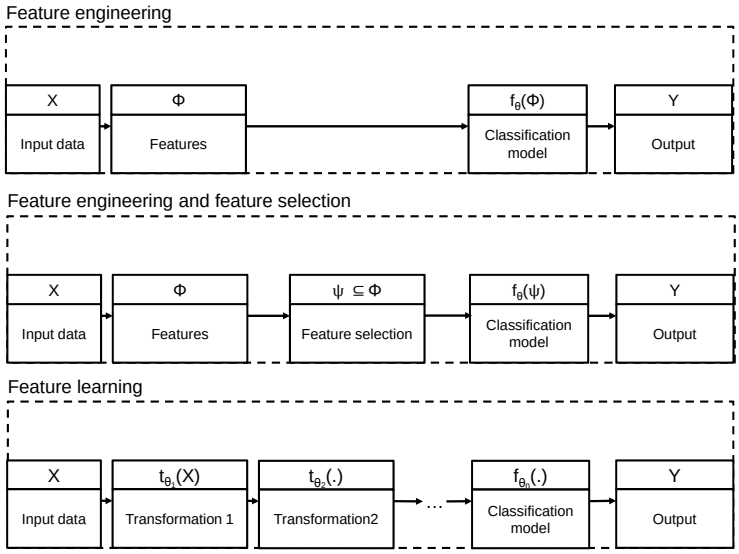


Figure 4.1: Schematic representation of feature engineering, feature engineering in combination with feature extraction and feature learning.

are extracted (ϕ) and used to train a classification algorithm ($f_\theta(.)$) that outputs predictions ($\mathbf{Y}$). The learnable parameters of the classification algorithm are denoted by θ. In the part of the figure about feature engineering in combination with feature selection, a feature selection step is added wherein a subset of the features are selected ($\psi \subseteq \phi$) that are afterwards used in the classification algorithm. In the feature learning part of the figure no manually engineered features are extracted from the input data, instead, the input data is transformed using $t_{\theta_1}(.)$ wherein θ_1 consists of the learnable parameters of the transformation. The transformation will output a new representation of the input data that should be better suited for the classification task. Transformation steps can be repeated many times –each with their own set of learnable parameters– so that at the classification step the data is transformed optimally, i.e., optimal features are learned for the classification task.

In recent years feature learning has become very popular in the form of deep learning (DL) [3]. DL methods are representation-learning methods with multiple levels of representation, obtained by composing simple non-linear modules that each transform the representation (of the data) at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. DL is done using various types of deep neural networks (DNN) wherein every

layer will learn a new representation of the data (i.e. learn features). In recent years DL achieved state-of-the-art results in image recognition, speech recognition, drug discovery, analysing particle accelerator data and natural language processing [3].

In this chapter, DL using convolutional neural networks is applied to thermal infrared videos and vibration signals originating from a rotating machine. We show that transfer learning can be applied on our thermal infrared videos by using a pre-trained convolutional neural network network which was trained on natural images. Furthermore, we show that the network focusses on certain parts of the infrared thermal images indicating that the network has learned relations between certain hotspots and conditions. Finally, we show that by combining a convolutional neural network for vibration data and a convolutional neural network for IRT data, a very well performing fault detection system can be created.

4.2 Background

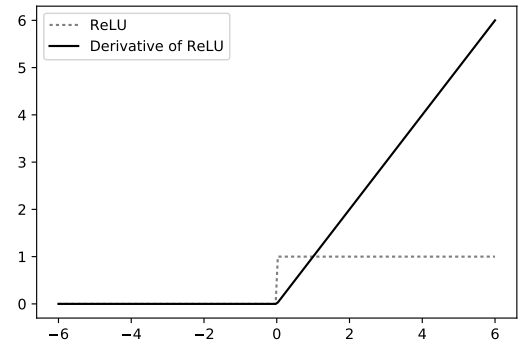
In this chapter we investigate DL methods in the form of neural networks for feature learning purposes. More information on traditional neural networks can be read in Section A.5.2 of the preliminaries chapter.

4.2.1 Deep learning

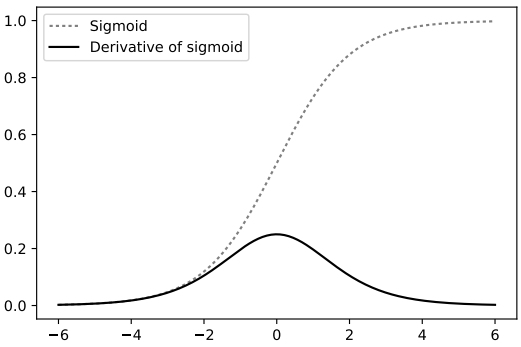
Neural networks have been around for several decades. However, in the late 1990s neural networks were abandoned by most researchers and not even considered by the computer-vision and speech-recognition communities [3]. It was considered that neural networks could not serve the purpose of feature learning with little prior knowledge.

In 2006, researchers at the Canadian Institute of Advanced Research, introduced unsupervised learning of layers [3]. Hence, layers could be trained independently from one-another and resulted in layers which learned features or their own. By stacking such pre-trained layers, and fine-tuning the entire architecture using gradient descent, well performing neural networks could be created.

Up until the year 2012, there was a problem when training NNs with many hidden layers –which were randomly initialized– as a whole, called *vanishing gradient* or *exploding gradient*. During NN training, gradient descent is used and to calculate the gradients, back-propagation is used. Back-propagation in essence is the chain-rule applied to a NN. Hence, the gradient is propagated backward through each layer. With each subsequent layer the magnitude of the gradients gets exponentially smaller (vanishes) thus making the steps also very small, which results in very slow learning of the weights in the lower (first) layers of a DNN. An important factor causing the gradients to shrink are the activation function derivatives (i.e. derivative of a layer's output with respect to its input). When the



(a)



(b)

Figure 4.2: (a) Relu activation function and its derivative and (b) the sigmoid activation function and its derivative.

sigmoid activation function is used in the network, the magnitude of the sigmoid derivative is well below 1 in the function's range causing the gradient to vanish as can be seen in Figure 4.2b. To solve this problem, in 2012 Krizhevsky et al. [88] proposed another type of activation function, called the rectified linear unit (ReLU, see Equation 4.1 and Figure 4.2a) which does not suffer from this problem. Hence, the vanishing gradient problem was mostly solved enabling much deeper NNs to be trained resulting in many new state of-the-art-results.

$$f(x) = max(0, x) \tag{4.1}$$

A neural network is commonly dense and fully connected, meaning that every

neuron of a layer is connected to every other neuron in the subsequent layer. Each connection is a weight totalling many parameters. This number of parameters is difficult to train as the network will memorize the data (overfitting), especially when too little data is available. If possible, a partial solution to this problem is to gather more data. Nevertheless, the training procedure will take very long. Another partial solution is implicitly provided in convolutional neural networks (CNN) [89] as CNNs are designed to deal with images and therefore exploit certain properties resulting in faster training, but also less parameters to train.

4.2.2 Convolutional neural networks

As opposed to a dense NN, a CNN is designed to use input data that exhibits a topological structure, such as a grid of pixels (such as image) or multiple sequences of values (such as signals). A CNN layer computes a similar transformation as the one in Equation 4.2 of a standard fully connected layer, with the difference that the adjustable parameters of the layer (i.e. weights) are organized as a set of filters (or filter bank) and cross-correlated over the input to produce the layer's output. The output of a CNN layer is a 3D matrix, which consists of a stack of matrices called feature maps, and can be used as input to a higher level layer of the CNN model. The CNN transformation can be represented as in Equation 4.3.

$$\mathbf{x}_n = f(\mathbf{W}_n \mathbf{x}_{n-1} + \mathbf{b}_n) \tag{4.2}$$

$$X_k^{(m)} = f\left(\sum_{c=1}^{C} W_n^{(c,m)} * X_{n-1}^{(c)} + B_n^{(m)}\right) \tag{4.3}$$

In Equation 4.3 the layer of the network is denoted with n as before, and the $*$ operator is used for the 2D cross-correlation of channel $c = 1, \ldots, C$ of the input X_{n-1} and the filter $W_n^{(c,m)}$, which is responsible for the m-th output feature map $X_k^{(m)}$, where $m = 1, \ldots, M$. The matrix $B_n^{(m)}$ contains the bias weights. Finally, a nonlinear activation function f is applied to the sum of convolutions to obtain the final output. A visual representation of this operation can be seen in Figure 4.3 and an example can be seen in Figure 4.4.

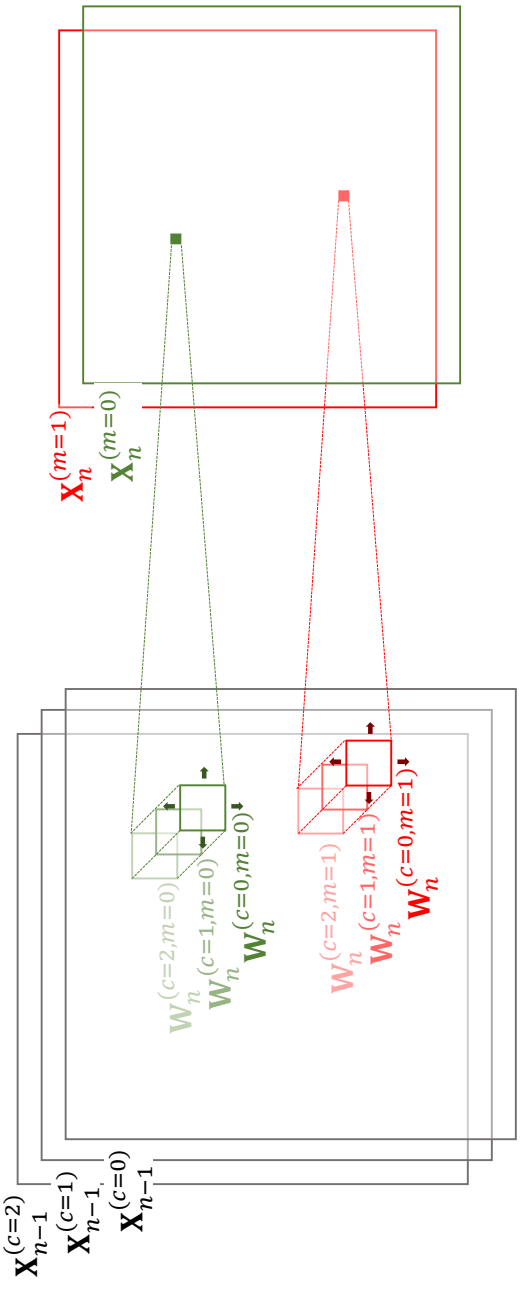


Figure 4.3: Visual representation of an example of a convolution layer. At the left-hand side there is for example an image with three channels (RGB). Two filters, each with three channels, are cross-correlated over the input image. This results in the output at the right-hand side, which has two channels.

Every set of weights in a convolutional layer can be thought of as a filter. Similar to filters in the image processing domain, these filters extract specific properties from the given data. However, as opposed to the filters in the image processing domain, the filters in a CNN are trained/adapted for the task at hand.

In a convolutional layer not every input value is connected to every node in the subsequent layer. Only small, local regions are connected to specific hidden nodes. This property is called *local connectivity*. Furthermore, instead of having a unique set of weights for every local area, the weights are shared. The idea behind this property, i.e. *weight sharing*, is that the set of learned filters will extract the same feature in different areas of the input. By convolving a filter over the input, a feature map/output is obtained. The output can have reduced dimensions and parts of the output will be suppressed by the filter and some parts will be made more noticeable. For example, a specific filter can have learned to detect edges in an input image. Because of the weight sharing property, edges will be detected in the entire input image. To be able to detect many features, multiple feature maps and accompanying filters are required. In the example in Figure 4.3 an image with three channels is provided as input. The output consists of two feature maps with their respective set of weights which are convolved over the input image. Note that a set of weights has the same amount of channels as the input, however, the amount of channels in the output, i.e. feature maps, is a hyperparameter that has to be tuned.

Beside convolutional layers, a CNN often also uses pooling layers. *Pooling* is done after a convolutional layer and reduces the dimension of the feature maps. Pooling is applied by sliding a small window over the feature maps while extracting a single value from that region by the use of for example a max or mean operation. Because of this property, i.e. *pooling*, a feature map will reduce in size, resulting in less parameters and less computations in subsequent layers.

Due to these three properties, i.e. local connectivity, weight sharing and pooling, a CNN contains significantly less parameters than a dense NN, making the CNN less susceptible to overfitting and faster to train.

4.2.3 Transfer learning

Data sets are often very small for tasks in specialized fields compared to the required amount of data to train a DNN. Hence, DNNs will tend to overfit. To overcome this problem, pre-trained networks can be used which are NNs trained for another task for which a lot of data was available. Such networks are for example alexnet [88], VGG network [90] and the GoogLeNet [91]. In essence, the weights of the already trained network are re-purposed for the new tasks. It has been shown that such a NN will learn general features that can be used for other tasks [92, 93]. It has also been shown that NNs, which are trained on images of everyday scenery,

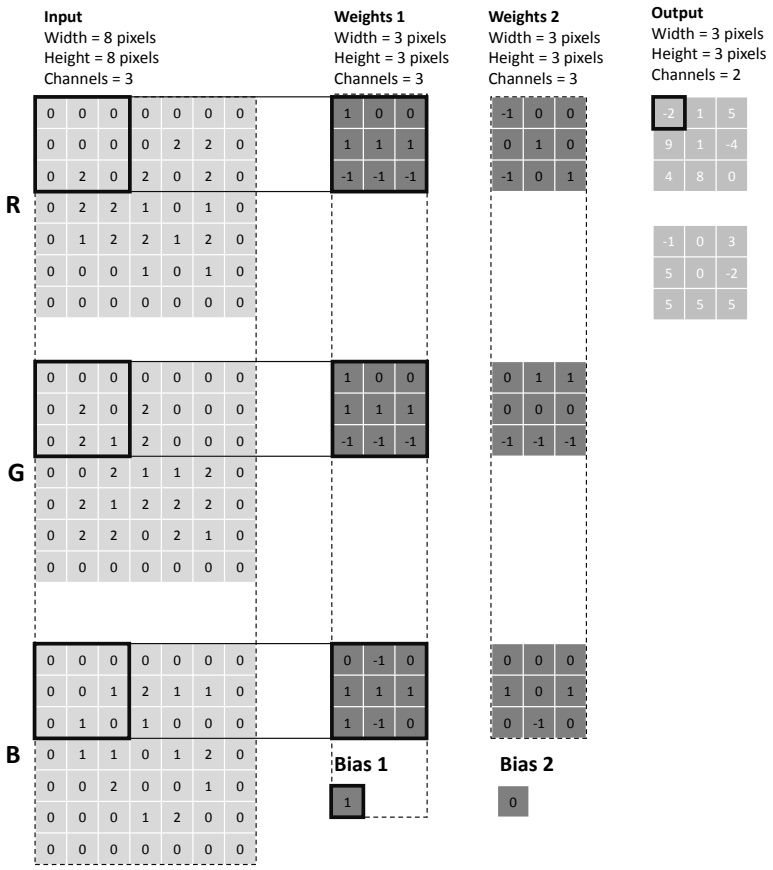


Figure 4.4: Visual example of a convolutional layer. The input is for example an image of 8 by 8 pixels. The image has three channels (red, green, blue). There are two filters/weights, hence there are two channels in the output. Weights 1 result in channel 1 in the output, and weights 2 in channel 2. As the weights slide over the image with stride 2, and each step results in a single value in the output, the output has a width and height of 3 pixels.

can be re-purposed and modified to be applicable in tasks which require domain specific images, such as medical images [94] or aerial images [95]. The process of re-using and modifying a trained NN is called transfer learning. There are several methods to apply transfer learning [92]:

- **Option A**: Remove the last layer (k). Hence, by providing the modified pre-trained NN with input samples the network will output intermediary abstract representations of the data that can be given to a new classifier such as a support vector machine. The idea behind this approach is that the network has learned re-usable features, which are useful for the task at hand, and that only the classifier has to be trained for the task at hand.
- **Option B**: Remove multiple layers ($k, .., k-t$) and use the output of the last remaining layer to train another classifier such as a SVM. This is similar to option A, but by using the output of an intermediate layer, a more abstract representation of the data is obtained.
- **Option C:** In addition to removing one or more layers, it is also possible to attach a new layer to the modified pre-trained network. The idea behind this method is that the initial layers have learned useful weights, but that the last layer (layer that will classify the data using the softmax function), was not initially designed for the task at hand. Hence, it has to be replaced and trained. During the training phase all the layers can be trained, or only the newly added layers. It is often useful to retrain all the layers because weights of subsequent layers actually depend on each other.
- **Option D:** Similar to option C, but multiple new layers are added to the network. In this option the assumption is made that the initial layers have learned useful representations of the data, but the final layers have not. Again, a decision has to be made to train the entire network, or only the newly added layers.

The different options are also visualized in Figure 4.5

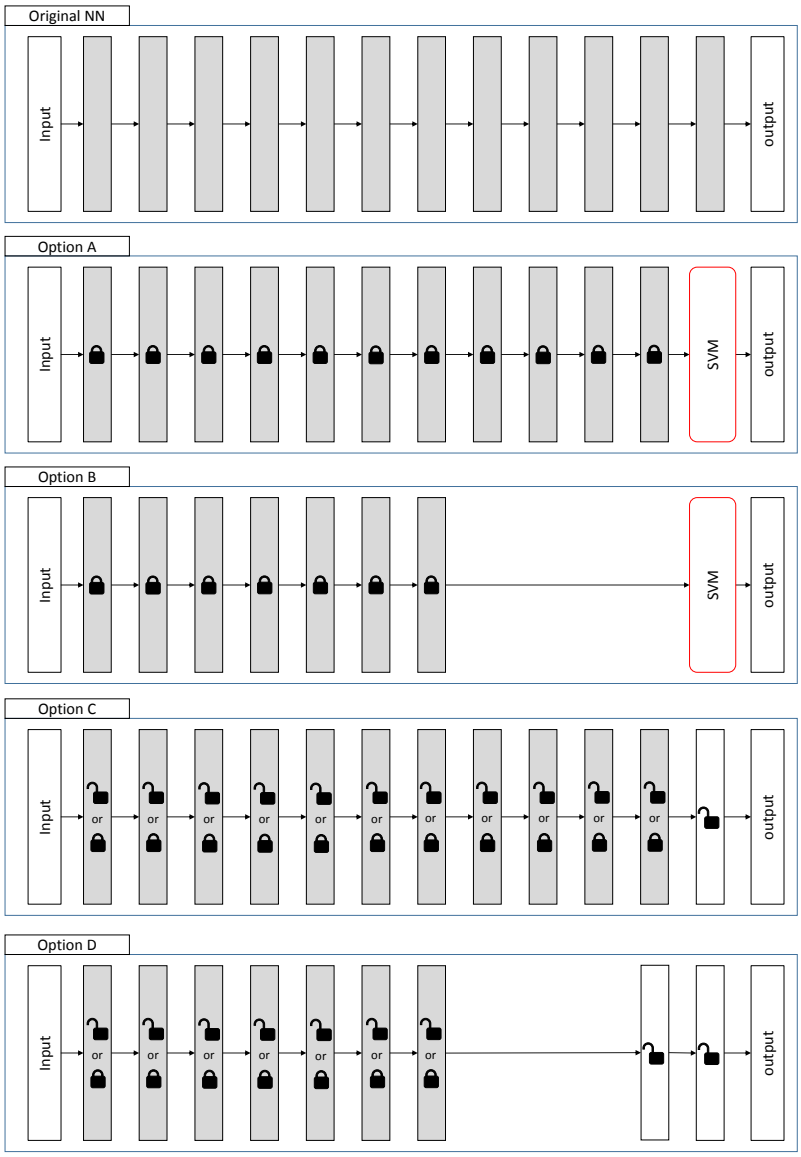


Figure 4.5: Visual representation of the different transfer learning options. Gray boxes indicate pre-trained layers, white boxes indicate newly added layers, a closed lock icon indicates that the layer is not trained in the training phase and finally, the open lock icon indicates that the layer is trained during the training phase.

4.3 Related literature

Neural networks (NN) have been used for many decades. However, most often they are used in combination with features engineered by an expert [96, 97]. In contrast, feature learning uses a raw representation of the input data and lets an algorithm learn and create a suitable representation of the data, i.e. features. An example of such a process using NNs is given in [98], wherein vibration spectrum images are created and given to NNs for rolling element bearing fault classification. Feature learning can be done using both supervised and/or unsupervised methods. Within the condition monitoring domain, feature learning has not been used extensively, only unsupervised methods, more specifically auto-encoders have been used for bearing fault detection using vibration measurements [99]. Auto-encoders are NNs which are designed to replicate the given input. The NN has a single hidden layer containing less nodes than the input layer. The purpose of this hidden layer is to learn a compressed representation of the input data. An auto-encoder is used to extract features which are given to a classification algorithm. It should be noted that many auto-encoders can be stacked on top of each-other to form a deep neural network. Although, there is little related work on deep learning for condition monitoring, the techniques are very promising as is illustrated in this chapter.

4.4 Methodology

The data sets from Chapter 3 are also used in this chapter, this way enabling a comparison between the feature engineering approach and the feature learning approach for CM on the component level.

4.4.1 Architecture

Since a condition can be assigned two labels (i.e. a REB condition and a machine condition), as in Chapter 3, the automatic detection of a specific condition is regarded as a combination of two multi-class classification problems. Hence, for each modality two neural networks are trained. One CNN will distinguish between the REB conditions (for data set one this is HB, MILB, EILB and ORF; for data set two this is HB, HP and ORF). The second CNN distinguishes between the different gradations of imbalance. If only one network for all conditions would be used, the classification task would become more difficult as there would be more classes.

Next the CNN for the vibration measurements is discussed.

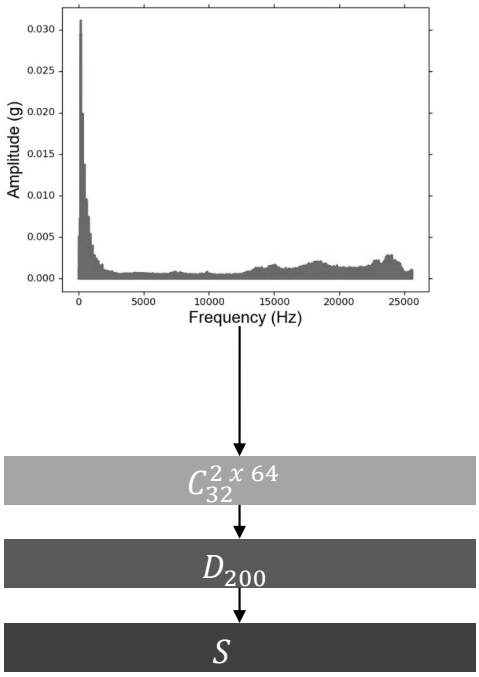


Figure 4.6: Architecture of the CNN model using vibration measurements.

4.4.2 Vibration-based fault detection

The proposed CNN approach is similar to the one proposed in [100]. However, here, the network is given both the frequency spectrum of the signal containing accelerations in the x-direction and also the frequency spectrum of the signal containing accelerations in the y-direction. The short-hand notation of the architecture of the network is as follows: $C_{32}^{2\times 64} - D_{200} - S$ where $C_k^{h\times w}$ denotes a convolutional layer with k feature maps, h height and w width. The height of the convolutional layer corresponds to the two signals originating from the accelerometers. D_n denotes a dense fully connected layer with n nodes. S denotes a softmax layer. For the convolutional layer and dense layer the tanh activation function is applied and cross-entropy is used as loss function. Figure 4.6 shows a diagram of the proposed CNN architecture. The convolutional layer corresponds to Equation 4.3, and the fully connected layer to Equation 4.2.

The input signals are preprocessed in order to train the model. First, the accelerometer signals are scaled to have zero mean and unit variance. Then, from the training set signals, non-overlapping windows are extracted containing one second

of measurement samples. For each window of extracted samples, the DFT is calculated. The amplitudes of the frequency decompositions are then used as training samples for the neural network model. It was experimentally determined that the accelerometer's sampling resolution could be lowered without affecting the output of the model. Therefore, 5-fold subsampling is applied on the original accelerometer data. It should be noted that only the amplitudes of the frequency spectrum are used and not the frequency values themselves. This is because the sequence of the amplitudes remain unchanged in the data and hence the order of the frequencies is implicitly present due to the order of the amplitudes. The CNN model was trained using minibatch gradient descent and momentum [101], using 100 training examples per minibatch.

There are three notable aspects about the proposed CNN architecture:

- The architecture contains no pooling layers as pooling will reduce the dimensionality of the feature maps obtained from the convolutional layers by selecting the maximum activation in a small region of a feature map. For example, a filter has learned to detect a large amplitude in a region. If such a large peak occurs somewhere in the frequency spectrum, the result of applying the filter to that region will yield a large value, but a small value in other regions. By subsequently applying pooling, information of occurrence is kept but the location is lost. The information of where it occurred is however important to detect a specific fault as large amplitudes at specific frequencies are related to certain faults. The usefulness of pooling layers was also experimentally validated and pooling did not improve results.

- It has been shown that by using a deep architecture, i.e., a network containing layers, the network becomes more robust to the variation in the data [102]. Hence, if the dataset has a lot of variation, a deep architecture is required. As the manifestation of the different faults considered here shows little variation, a shallow architecture suffices. Furthermore, the initial layers of CNNs learn the fastest, hence a short training time is sufficient to achieve convergence [102]. Several variations of the proposed network were tested by varying the number of convolutional and fully connected layers, and the number of units per layer. For our particular use case, it was determined that a deep version of the proposed architecture does not yield better results. In other words, adding additional layers to the network will not cause an increase in accuracy.

- Changing the width of the filters in the convolution layer does influence the results. Currently, the width is low, given the length of the signal. Lowering the width further will slightly decrease the accuracy. For example, halving the width will lead to an approximate accuracy loss of 1 %. However,

increasing the width will decrease the accuracy much more rapidly. For example if the width is doubled, the accuracy will be approximately 5 % lower. The amount of filters does not influence the results much compared to the width of a filter. However, if too few filters are chosen i.e. 8 instead of 32, then the accuracy will drop by 1 % because too few features will be detected.

Depending on the task at hand, the amount of outputs will be *four* to determine the bearing condition in data set one; *two* to determine if there is imbalance or not for data set one; *three* to determine the bearing condition in data set two and *four* to determine the gradation of imbalance in data set two.

4.4.3 Infrared thermal video based fault detection

As opposed to vibration data, images have many more variables. An input sample for the vibration CNN has 2×5120 input variables. An image has many more parameters (640×480), hence a deeper network is required. However, it was observed that the data sets contained too little data to properly train a DNN for IRT data. Hence, as discussed in the related literature section we opted to apply transfer learning.

Various transfer learning methods were tested, however, option C, described in Section 4.2.3 resulted in the most robust solution providing the best results. We opted to use a pre-trained convolutional neural network from the visual geometry group (VGG) at the University of Oxford [90]. There are several pre-trained networks available, however, at the time of writing the VGG network achieved the state-of-the-art results on the image-net data set and the network is made publicly available. Hence, the network was trained on natural images. The goal of the VGG network was to classify images in one of a thousand categories. The VGG network uses rectified linear activation functions in every layer except the last layer, which is a fully-connected layer, where softmax activation functions are used.

For transfer learning purposes the last layer of the VGG network was removed as our data-set has fewer classes. Removal of additional layers did not perform any better. A new fully-connected layer was attached to the network. This layer also uses softmax activation functions, and less weights, as there are less classes to distinguish for the task at hand. Tests were also done where multiple layers were attached, but no improvements were observed. Furthermore, more layers results in more training time. In the end this means that all except for one layer of the VGG network (which are pre-trained) are reused in our network and only the last layer is new. As has been demonstrated in other research, the fact that a network's layers have been trained using a certain type of images, does not mean that transfer learning is not possible for totally different types of images. Hence, we hypothesis that a pre-trained DNN, such as the VGG network, can be re-used for component condition detection using IRT images.

As the input layer of the VGG network is also re-used, our dataset also has to be preprocessed as the data that was initially provided to the original VGG network. Hence, preprocessing as described in [90] was applied. Images (i.e. frames) are pre-processed by removing the mean value. Furthermore smoothing is also applied using a Gaussian kernel with a standard deviation of 3 pixels. Another requirement is that the input image has a width and height of 224 pixels. Hence, the images are manually cropped after an image registration step (as in Chapter 3).

As the last layer of our network is not trained for our task, a training phase is required. In this training phase mini-batch gradient descent is used to update all the weights of the network, even the weights of the pre-trained layers. The idea behind this methodology is that neighbouring layers co-adapt during training, which can only be done when training all layers [92]. However, the learning rate for the mini-batch gradient descent algorithm should be smaller than the original learning rate to minimally influence the already pre-trained layers. The learning rate was set to 1.10^{-5}. The architectures of our network is as follows: $C_{64}^{3\times3} - C_{64}^{3\times3} - P - C_{128}^{3\times3} - C_{128}^{3\times3} - P - C_{256}^{3\times3} - C_{256}^{3\times3} - C_{256}^{3\times3} - P - C_{512}^{3\times3} - C_{512}^{3\times3} - C_{512}^{3\times3} - P - C_{512}^{3\times3} - C_{512}^{3\times3} - C_{512}^{3\times3} - P - D_{4096} - D_{4096} - S$. As activation function the ReLu activation function is used for all the layers except the last one where the softmax function was used. As loss function the cross-entropy loss is used.

The network can also be seen in Figure 4.7.

In order to detect imbalance using infrared images the same network structure and transfer learning methodology is applied. However, the input data is different. It was determined that imbalance could not be detected by solely using spatial information of the heat distribution in the component. Temporal information is required to make the vibrations visible in the images. Hence, subsequent images are differenced (i.e. $I_{t-1} - I_t$). By differencing these frames, the movement in the images become visible as described in chapter 3, and as can be seen in Figure 4.8. Afterwards, these images are rescaled to 224×224, normalized and used to train a CNN.

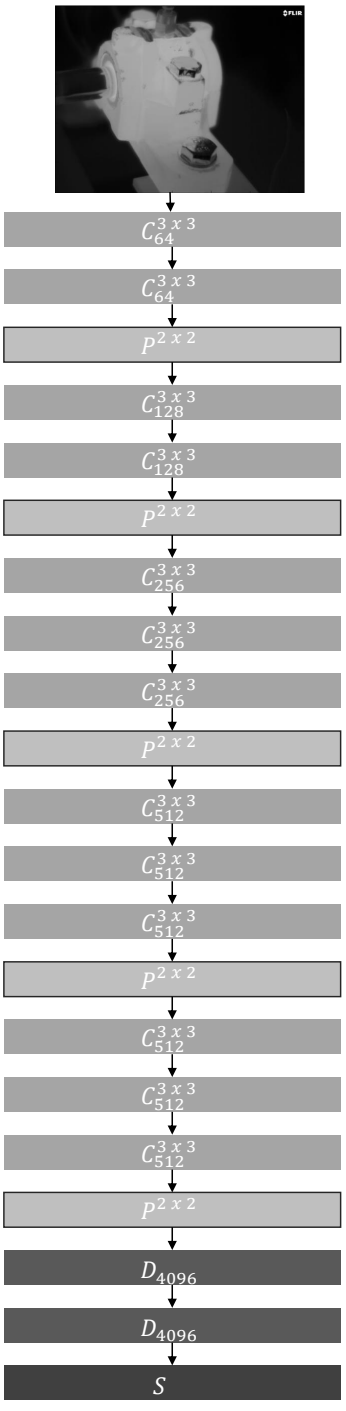


Figure 4.7: Architecture of the CNN model using infrared thermal imaging data.

Figure 4.8: Visible movement in the differenced frame.

4.4.4 Multi-sensor approach

We observed that the single-sensor approaches could still be improved. Hence, similar to Chapter 3, in this section the potential of a multi-sensor system is researched.

In Figure 4.9 the multi-sensor architecture can be observed. As can be seen, the networks for the respective modalities are combined. In order to make a decision based on the output of the two networks, the output of the two networks are averaged. It should be noted that the networks are not trained in this configuration. The networks are trained separately on their respective data and afterwards combined. This is possible as the decision fusion does not require a layer to be trained as it requires an averaging operation. In the end, to classify all the different conditions two such multi-sensor architectures are required: one to detect the bearing conditions and another to detect the (gradation of) imbalance. In the next section the result of the CNN approaches are presented and discussed.

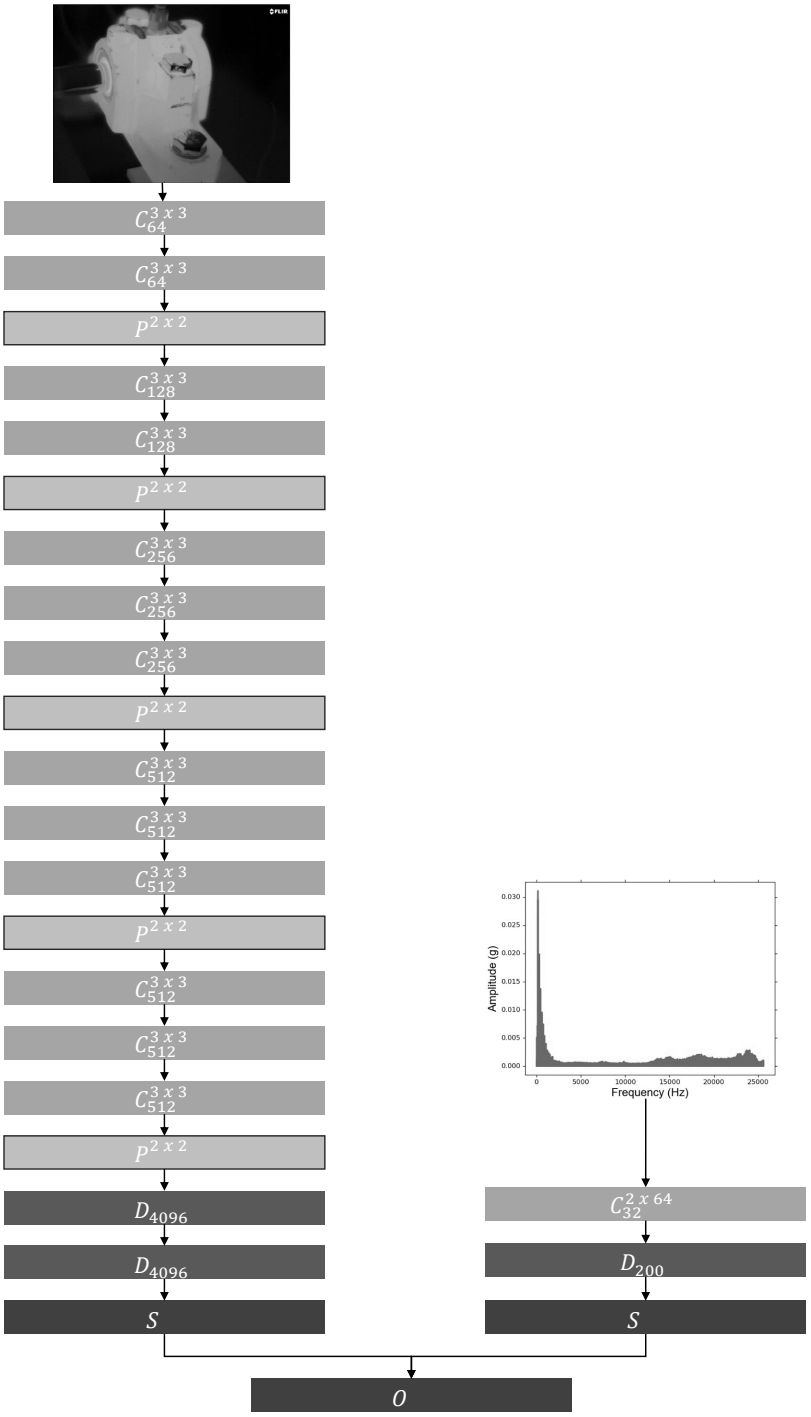


Figure 4.9: Architecture of the multi-sensor fault detection system. $C_k^{h \times w}$ denotes a convolutional layer with k feature maps and receptive field of dimension $h \times w$. P denotes a pooling layer. D_n denotes a dense fully connected layer with n neurons. S denotes a softmax layer and O denotes the output layer.

4.5 Results

To put the results of the feature learning based approach for component-level fault detection in perspective, the results obtained in this chapter are compared to the results from Chapter 3. Again, the results for both data sets are discussed.

4.5.1 Results data set one

The results for both the feature engineering based approach and the feature learning based approach (i.e. the CNNs) on data set one are listed in Table 4.1. The first two columns indicate which modalities have been used to obtain the result. The third column identifies what method has been used (i.e. feature learning or feature engineering). The fourth column specifies which conditions were targeted and the last column lists the accuracy.

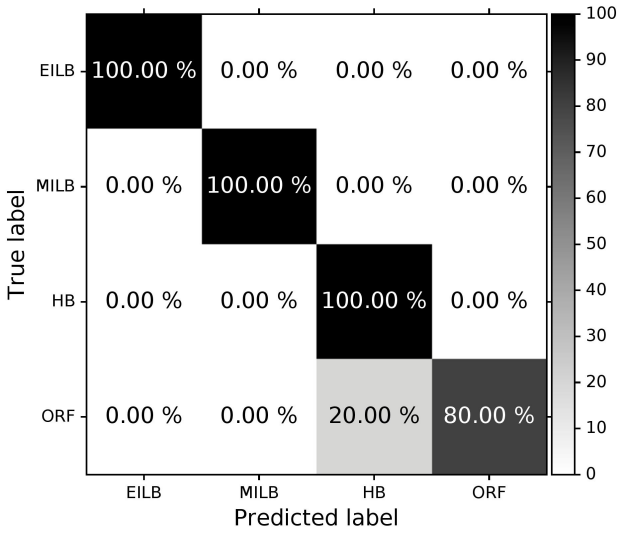
Three notable insights can be observed in this table:

1. Both feature engineering and feature learning are able to distinguish between all 8 conditions perfectly.
2. A multi-sensor approach, using two modalities, improves the results significantly for feature engineering (88.25 % $\Rightarrow$ 100.00 %)
3. A multi-sensor approach, using two modalities, also improves the results up to 100 % (similar to the feature engineering system), however, single sensor based approaches already achieve a high accuracy scores.

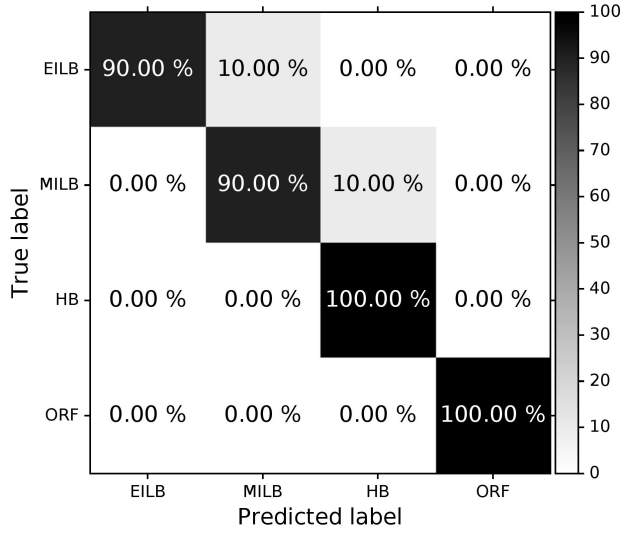
Finally, when observing the confusion matrices (Figures 4.10) it can be seen that when using IRT data, ORFs are not perfectly detected, and when using vibration measurements, faults related to lubrication conditions are less well detected. These observations are in line with the conclusions drawn in Chapter 3. However by using feature learning, the amount of errors are reduced. Finally, by combining the two systems in a multi-sensor system, perfect classification results are achieved.

IR	VIB	Method	Conditions	Accuracy
IR		FE	MILB, EILB, HB, ORF	88.25 % ($\sigma = 8.07$ %)
IR		FL	MILB, EILB, HB, ORF	95.00 % ($\sigma = 6.12$ %)
	VIB	FE	MILB, EILB, HB, ORF	87.25 % ($\sigma = 8.10$ %)
	VIB	FL	MILB, EILB, HB, ORF	95.00 % ($\sigma = 6.12$ %)
IR	VIB	FE	MILB, EILB, HB, ORF	100.00 % ($\sigma = 0.00$ %)
IR	VIB	FL	MILB, EILB, HB, ORF	100.00 % ($\sigma = 0.00$ %)
IR		FE	balance and imbalance	100.0 % ($\sigma = 0.00$ %)
IR		FL	balance and imbalance	100.0 % ($\sigma = 0.00$ %)
	VIB	FE	balance and imbalance	100.0 % ($\sigma = 0.00$ %)
	VIB	FL	balance and imbalance	100.0 % ($\sigma = 0.00$ %)
IR	VIB	FE	balance and imbalance	100.0 % ($\sigma = 0.00$ %)
IR	VIB	FL	balance and imbalance	100.0 % ($\sigma = 0.00$ %)
IR		FE	All 8 conditions	88.25 % ($\sigma = 8.07$ %)
IR		FL	All 8 conditions	95.00 % ($\sigma = 6.12$ %)
	VIB	FE	All 8 conditions	87.25 % ($\sigma = 8.10$ %)
	VIB	FL	All 8 conditions	95.00 % ($\sigma = 6.12$ %)
IR	VIB	FE	All 8 conditions	100.0 % ($\sigma = 0.00$ %)
IR	VIB	FL	All 8 conditions	100.0 % ($\sigma = 0.00$ %)

Table 4.1: Results of both the feature engineering and feature learning based approach on data set one. FE = feature engineering, FL = feature learning.



(a)



(b)

Figure 4.10: Confusion matrices for the CNN using IRT data (a) and vibration data (b) for bearing fault detection.

4.5.2 Results data set two

The results for both the feature engineering based approach and the feature learning based approach on data set two are listed in Table 4.2. To distinguish between the 12 conditions using the multi-sensor system, it was observed that for the imbalance detection the vibration based CNN did not improve the results. Hence it is not used for imbalance detection. Solely the IRT based CNN is used for this part of the multi-class and multi-label problem.

From the results, six interesting aspects can be deduced:

1. In the end, feature learning slightly outperforms feature engineering (93.33 % vs. 90.00 % respectively)

2. Imbalance detection using feature learning and vibration data is difficult (55 %). This is discussed further in Section 4.5.4.

3. A multi-sensor approach, using two modalities gives a huge improvement with respect to feature engineering (66.67 % $\Rightarrow$ 90.00 %)

4. A multi-sensor approach, using two modalities improves the results up to 93.33 %. However, the single sensor based systems already achieve high accuracy scores.

5. Outer-raceway faults seem to be detectable using IRT data and feature learing. This can also be seen in Figure 4.11a.

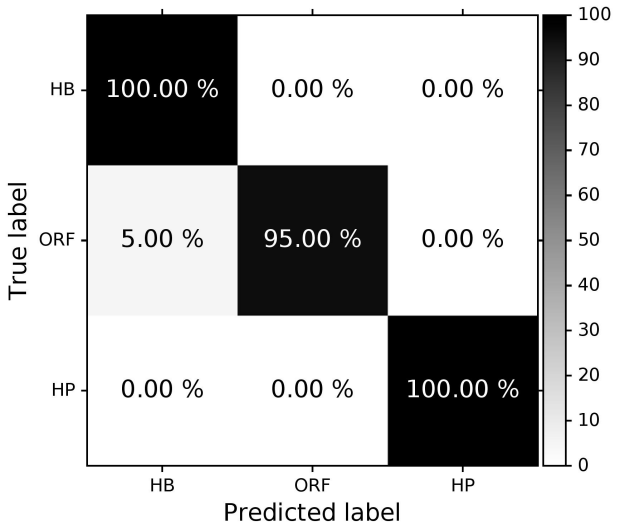
6. When observing the confusion matrix of the feature learning multi-sensor system (Figure 4.12), it can be concluded that the system only makes some mistakes for imbalance conditions while there is an outer raceway fault present at the same time, which can be expected as both ORFs and imbalance are known to generate vibrations, which can be difficult to distinguish using IRT data using the current pre-processing approach.

When comparing the confusion matrices of the feature engineering approach and the feature learning approach regarding bearing condition detection, it can be observed that when using vibration measurements, the feature learning approach provides better results. Only a small error is made where the CNN confuses HBs for REBS with HPs (Figure 4.11b), which is also present in the confusion matrix for the feature engineering approach (Figure 3.30a).

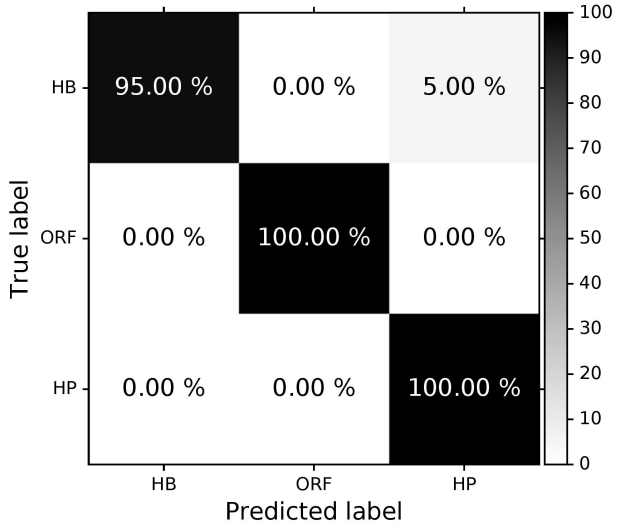
In general it can be concluded that the CNN approach gives very good results on both data sets without requiring much expert knowledge about the problem. However, NNs in general are black box system, meaning that their inner-workings are not human-interpretable. Nevertheless, some insights can be derived from them, which are presented next.

IR	VIB	Method	Conditions	Accuracy
IR		FE	HP, ORF, HB	65.00 % ($\sigma = 16.16$ %)
IR		FL	HP, ORF, HB	98.33 % ($\sigma = 3.33$ %)
	VIB	FE	HP, ORF, HB	91.67 % ($\sigma = 12.91$ %)
	VIB	FL	HP, ORF, HB	98.33 % ($\sigma = 3.33$ %)
IR	VIB	FE	HP, ORF, HB	100.00 % ($\sigma = 0.00$ %)
IR	VIB	FL	HP, ORF, HB	100.00 % ($\sigma = 0.00$ %)
IR		FE	Imbalance gradation	88.33 % ($\sigma = 12.47$ %)
IR		FL	Imbalance gradation	93.33 % ($\sigma = 9.72$ %)
	VIB	FE	Imbalance gradation	75.00 % ($\sigma = 9.13$ %)
	VIB	FL	Imbalance gradation	55.00 % ($\sigma = 4.08$ %)
IR	VIB	FE	Imbalance gradation	90.00 % ($\sigma = 6.24$ %)
IR	VIB	FL	Imbalance gradation	78.33 % ($\sigma = 12.47$ %)
IR		FE	All 12 conditions	55.00 % ($\sigma = 11.31$ %)
IR		FL	All 12 conditions	91.67 % ($\sigma = 9.13$ %)
	VIB	FE	All 12 conditions	66.67 % ($\sigma = 21.08$ %)
	VIB	FL	All 12 conditions	55.00 % ($\sigma = 4.08$ %)
IR	VIB	FE	All 12 conditions	90.00 % ($\sigma = 6.24$ %)
IR	VIB	FL	All 12 conditions	93.33 % ($\sigma = 9.72$ %)

Table 4.2: Results of both the feature engineering and feature learning based approach on data set two.



(a)



(b)

Figure 4.11: Confusion matrices for the CNN using IRT data (a) and vibration data (b) for bearing fault detection.

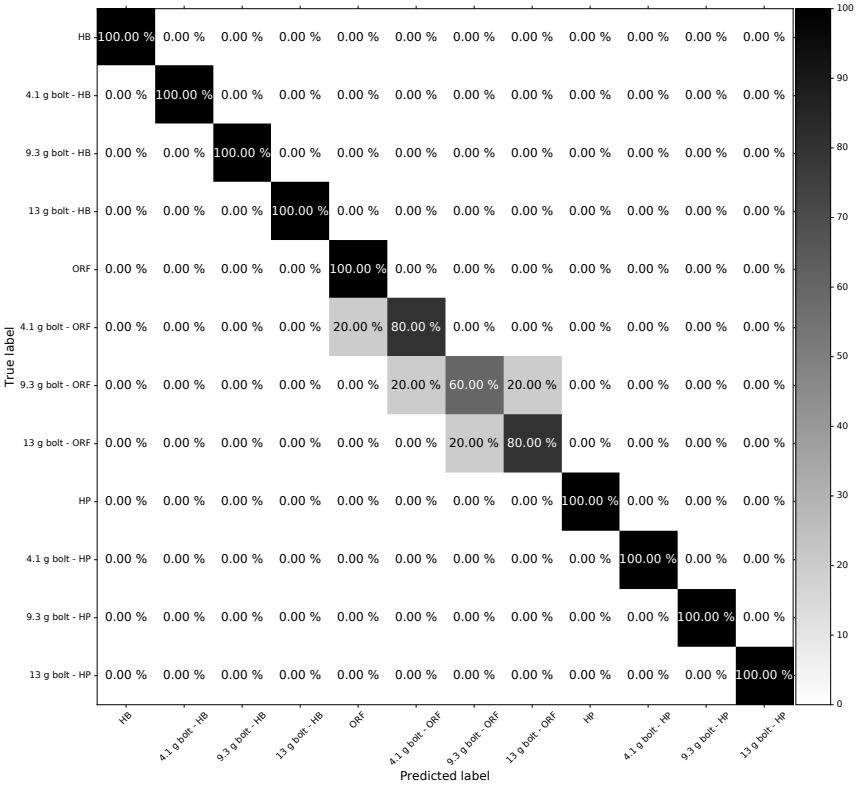


Figure 4.12: Confusion matrix of the multi-sensor feature learning system for all 12 conditions in data set two.

4.5.3 Insights into infrared thermal data

Generally, when using infrared thermal imaging for fault detection, it is hard to know where to look in the image to be able to detect a specific machine condition. However, as presented in the previous section, using data-driven feature-learning techniques it is possible to detect a condition without having any prior knowledge. Hence, it can be concluded that information related to the condition is present in the thermal images. Therefore, in this section we apply the technique described by Zeiler et al. [102] to investigate what exactly is important in a thermal image for the specific conditions.

The method has three steps that are iterated over:

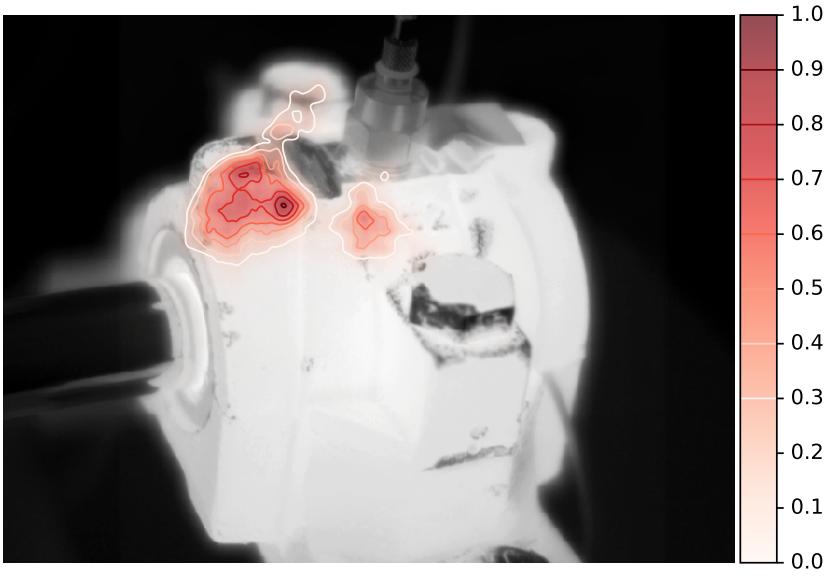
1. The first step masks a part of the input image (i.e. a 7×7 square of pixels is set to a constant value).

2. In step two, the modified incomplete image is classified by the trained CNN. The CNN has softmax activation functions in the output layer which give a probability for every possible class.

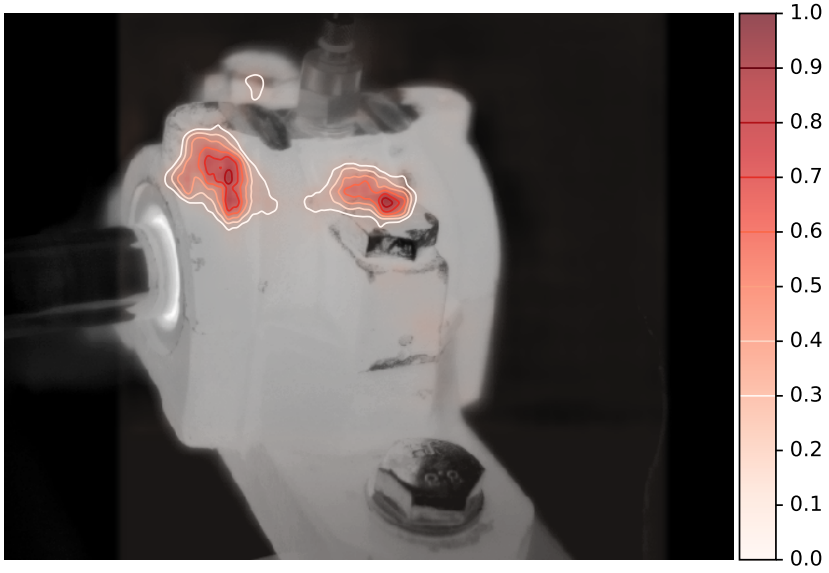
3. In the third step the class probability corresponding to the correct class is saved in a matrix with the same dimensions as the image. The probabilities are stored in the location corresponding to the location that was masked in the original image.

These three steps are iterated over so that every part of the image is masked once. The idea behind this method is that if an important and crucial part of the image is masked, the probability for the correct class will be low (i.e. closer to zero). Hence, if such a drop in probability is observed when a specific part of the image is masked, it can be concluded that said part of the image is crucial for that particular class. An intuitive example is given in [102], where a CNN is trained to detect objects in natural images. One of the possible classes is “dog”. Hence, if a picture of a dog is given to the NN where the face of the dog is hidden by the mask, the probability for the class dog, provided by the network, will be much lower compared to the case when the dog’s face is not masked.

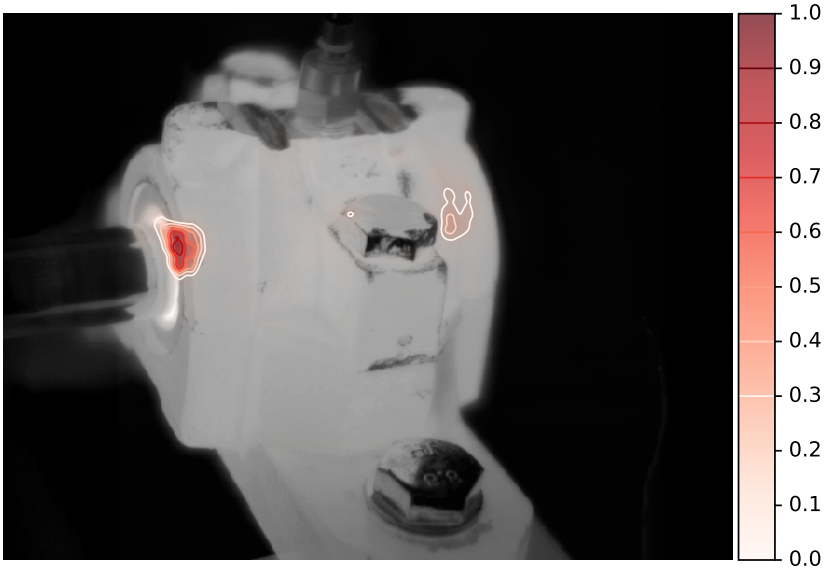
In Figure 4.13 the output of this method is visualized for the six bearing conditions. However, to make the areas more clear, we have visualized one minus the output probability and scaled in the range between 0 and 1. It can be seen that different areas on the REB’s housing are important for specific conditions. For example, to identify if a REB is extremely inadequately lubricated, the area around the seal is very important, which can for example be due to the heat originating from the increased friction between the shaft and the seal. In general, these locations can help to make a link to the underlying physics. In Appendix C, convolutional neural networks are applied on another use case where the oil level in a bearing is determined. In that use case, insights into the CNN’s decisions are more easily to link to the underlying physics.



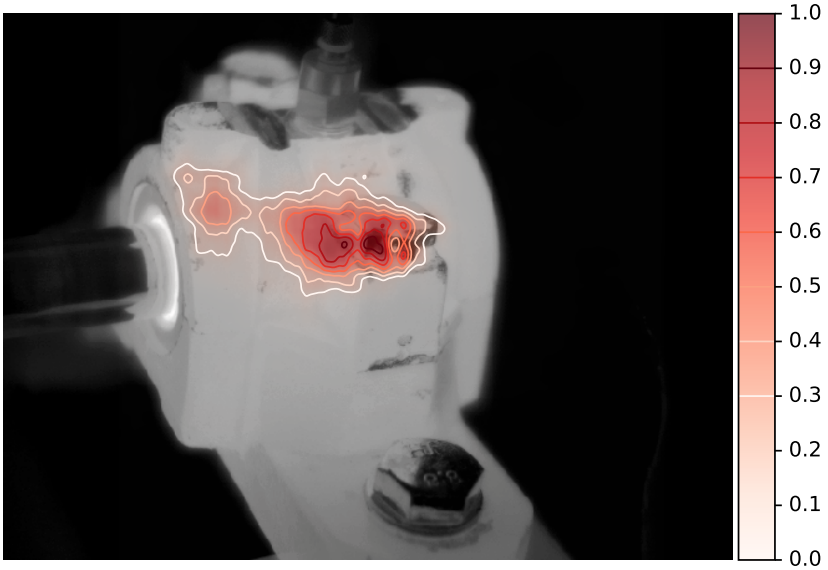
(a)



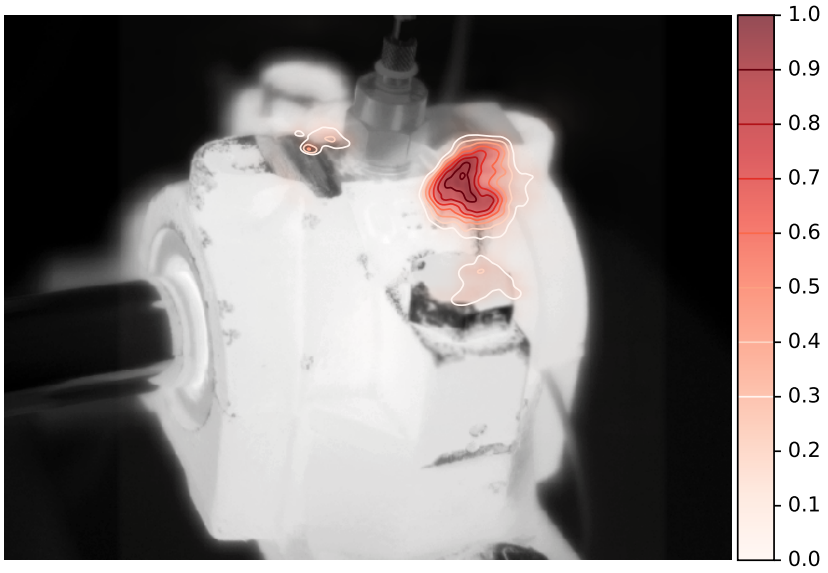
(b)



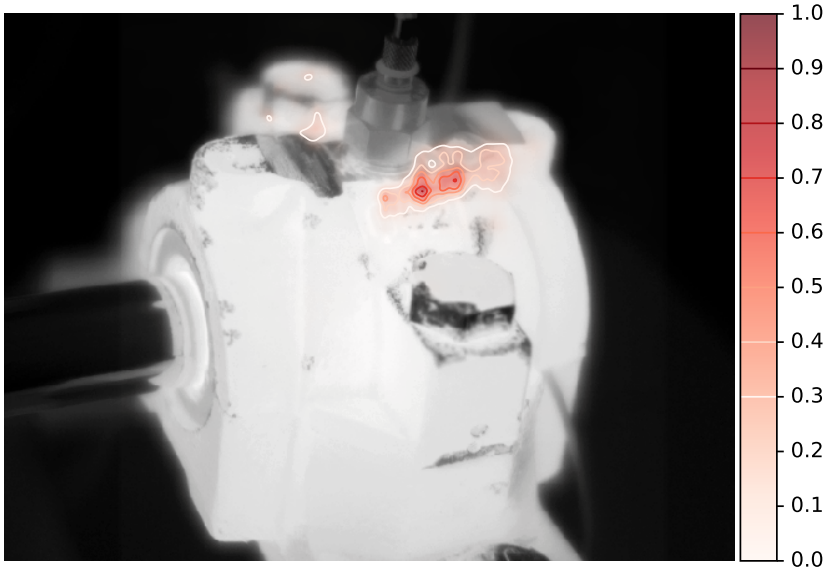
(c)



(d)



(e)



(f)

Figure 4.13: Regions that influence the CNNs output for a healthy bearing (a), mildly inadequately lubricated bearing (b), extremely inadequately lubricated bearing (c), outer-raceway fault at the 10 o'clock position (d), outer-raceway fault at the loaded zone (e), hard particles (f).

4.5.4 Imbalance detection

The results indicate that detecting the gradation of imbalance using vibration data is difficult resulting in an accuracy of 55 %. This result is counter intuitive as vibration data should contain information on the amount of imbalance. It is known that the amplitude at the rotation frequency depends on the amount of imbalance. To investigate this unexpected result, the frequency spectrum of every signal in data set two was visually inspected. It was determined that a small number of signals displayed unexpected behavior. The signals had a large DC component and had no significant peak at the rotation frequency. From the 60 recordings, 11 were removed for subsequent tests. The CNN was trained and tested again, similar to as was done initially. The accuracy of the CNN rose to 58.43 %. This means that this was not the main reason why the accuracy is low.

As the problem could not solely be attributed to the quality of the data, the imbalance gradations were investigated. The amplitude at the rotation frequency is expected to increase if the mass of the bolt causing the imbalance is increased. In an ideal situation, this amplitude would rise and would be noticeably different from the amplitude at a lower imbalance gradation. However, due to noise added to the vibration signal as a result of for example bearing mounting, sleeve tightening and amount of lubrication, this amplitude will fluctuate. Hence, there could be the problem that due to these additional effects, the intraclass variation per imbalance gradation is large. Additionally, because there are four levels of imbalance, the interclass variation could be low, hence making the signal indistinguishable from one-another.

To tests this hypothesis, convolutional neural networks were trained to distinguish between neighboring imbalance gradations. This means that three binary classification CNNs were created. The first distinguishes between balance and 4.1 g of imbalance; the second distinguishes between 4.1 g of imbalance and 9.3 g of imbalance; and the third distinguishes between 9.3 g of imbalance and 13 g of imbalance. If our hypothesis is correct then the three binary neural networks would achieve a low accuracy score. This however was not the case. The CNNs achieved 70 %; 93.33 % and 80 % accuracy respectively. These results indicated that our hypothesis is not correct.

The experiments above indicated that binary classifiers work well, hence, in addition to the three binary CNNs trained to distinguish neighboring imbalance gradations, three additional binary CNNs were trained. The first distinguishes between balance and 13 g of imbalance; the second distinguishes between 4.1 g and 13 g of imbalance; and the third distinguishes between balance and 9.3 g of imbalance. These additional binary classifiers also provided high accuracy scores (96 %; 88.67 % and 96.67 % respectively). Hence, the different gradations of imbalance are independently distinguishable to a certain extent.

Subsequently, an experiment was done to solve the multi-class imbalance de-

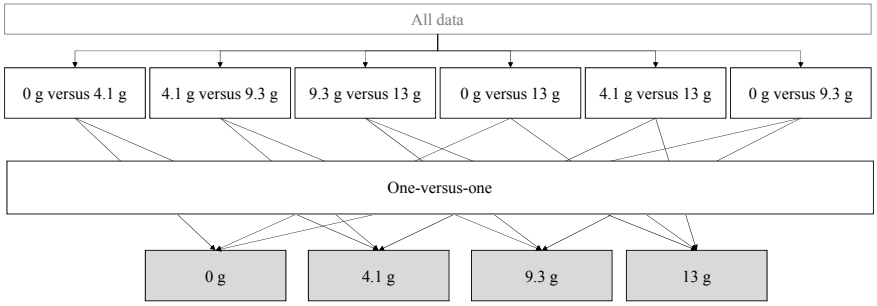


Figure 4.14: Block diagram of the one-versus-one approach combining six binary convolutional neural networks. All data is given to each binary CNN which can then vote for a certain imbalance gradation (lightgray boxes).

tection problem by combining the six binary CNNs using one-versus-one approach. Given a sample, each of the six binary CNNs can vote for one of the two classes they were respectively trained on. Hence, any class (i.e. imbalance gradation) can receive at most three votes. This strategy is illustrated in Figure 4.14. When testing this approach, the accuracy did not improve a lot (59.29 %). Even though the binary CNNs work well individually, when combined the accuracy gain is lost. It seemed that this approach did not work well because each binary classifier has to predict for all the data. For example, the binary CNN that distinguishes between 4.1 g and 13 g has to predict for every sample, even if a sample has a ground truth label of 0 g, for which that specific binary CNN hasn't been trained. Hence, such a classifier is more likely to misclassify, resulting in an overall accuracy reduction.

To solve this problem a one-versus-all approach was tested next. A one-versus-all approach requires four classifiers to be constructed (i.e. four binary CNNs). Each binary CNN is trained to detect if a sample belongs to one specific class or not. As the CNNs use the softmax activation function in the last layer, probabilities are provided as output. Each binary CNN provides the probability that a sample belongs to its specific class. This approach is illustrated in Figure 4.15. This approach requires less binary CNNs to be trained, but unfortunately does not provide better results. Each individual CNN is trained using imbalanced data as a result of the one-versus-all approach which results sub-optimal classifiers.

In the next experiment, a hierarchical combination strategy was tested. In this approach there are two classification levels. In the first level, a binary CNN distinguishes between small/no imbalance and large imbalance, i.e. 0 g and 4.1 g versus 9.3 g and 13 g. When all the data is provided to this single binary CNN, it provides a label for each sample in the data set. Using this label, the data set is split into two parts which are given to the second level binary CNNs. In the second level there are two binary CNNs. One that distinguishes between 0 g and 4.1 g imbalance and another that distinguishes between 9.3 g and 13 g. A block

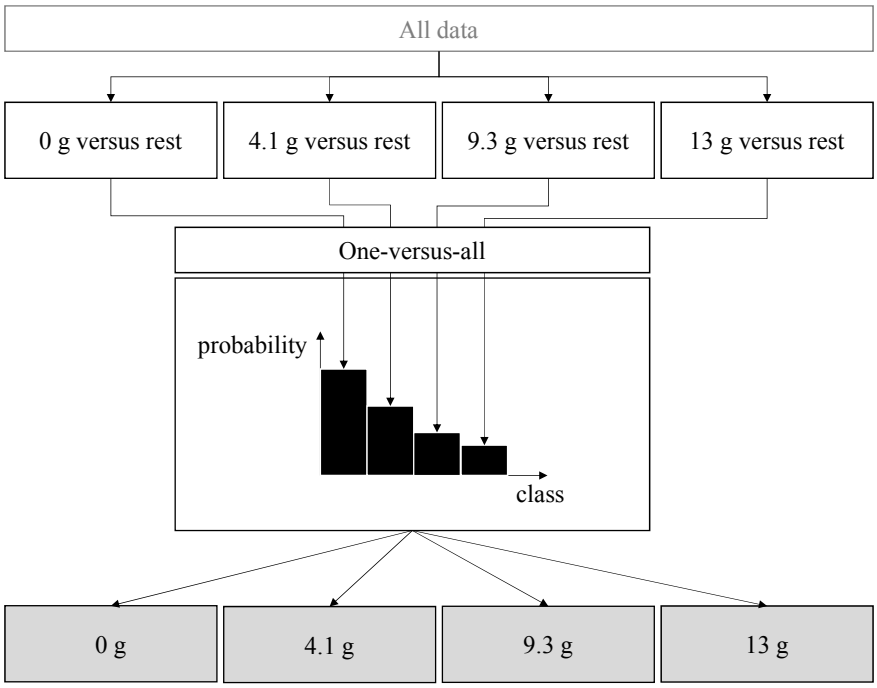


Figure 4.15: Block diagram of the one-versus-all approach. All data is provided to three binary convolutional neural networks. Each convolutional neural network will provide a probability of each sample belonging to a certain class, i.e. imbalance gradation. Hence, per sample, the class corresponding to the highest probability is assigned to the sample.

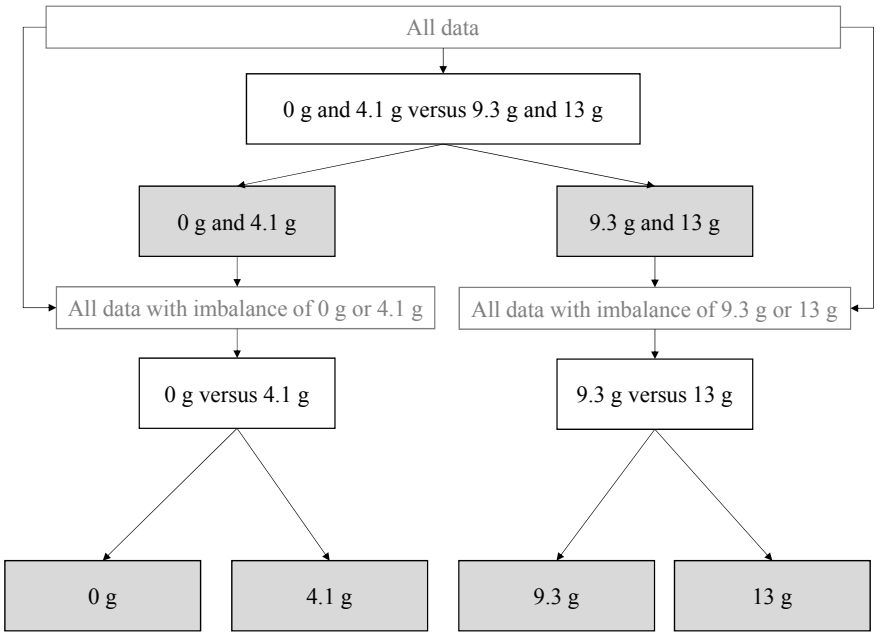


Figure 4.16: Block diagram of the hierarchical approach. All data is first provided to a single binary convolutional neural network. The CNN predicts one of two labels per sample which is used to split the original data set in two. The two subsets are given two two binary CNNs respectively which provide the eventual labels for the samples.

diagram of this approach can be seen in Figure 4.16. This approach only requires three binary CNNs and will result in balanced data at each stage. Furthermore, it will exploit the logical sequential order (ordinal) of the labels, i.e. balance $<$ 4.1 g imbalance $<$ 9.3 g imbalance $<$ 13 g imbalance. This resulted in an accuracy of 75.31 %. Which is a large improvement.

This approach illustrates that exploiting the ordinal nature of the labels is beneficial to detect imbalance gradation more accurately. Hence, in the final experiment the original multi-class neural network is used. However, two modifications are made to the network. First, the dense output layer only has one node and the softmax activation function is replaced by a linear activation function. As a results of this linear output unit, the network will predict a single continuous value for a given sample. This value corresponds to the amount of imbalance. Second, the loss function, which was previously the cross-entropy loss, is replaced by the squared error loss (see Equation 4.4, where y is the ground truth and $\hat{y}$ is the prediction provided by the CNN). The original loss function, i.e. the cross-entropy loss, is a loss function designed for classification purposes. Hence, the loss value will not differ if the predicted imbalance gradation is much larger than the ground

truth. The loss function only takes into account that the prediction is wrong. In contrast, the squared error loss value depends on how much the predicted value deviates from the ground truth. These two modifications convert the original CNN into a regression CNN that inherently takes the sequential order of the amount of imbalance gradation into account. For example, the network performs less well if it predicts 13 g if the label is actually 4.1 g than when it predicts 9.3 g.

$$f(\hat{y}, y) = (\hat{y} - y)^2 \tag{4.4}$$

Because the CNN now provides continues values due to the single linear output node, discretization is required to evaluate its performance. This means that in order to calculate the accuracy score, discrete values are required. Discretization is done according to Equation 4.5. The threshold values are simply set to half the interval between two imbalance levels. In Figure 4.17, this new approach can be seen next to the original approach. The regression-based approach results in a large increase in accuracy (81.17 %) which indicates that it is better to cast the detection of the gradation of imbalance as a regression problem instead of a classification problem.

$$f(\hat{y}) = \begin{cases} 0 \text{ g} & \text{if } \hat{y} < 2.05 \text{ g} \\ 4.1 \text{ g} & \text{if } 2.05 \text{ g} \leq \hat{y} < 6.7 \text{ g} \\ 9.3 \text{ g} & \text{if } 6.7 \text{ g} \leq \hat{y} < 11.15 \text{ g} \\ 13 \text{ g} & \text{if } 11.15 \text{ g} \leq \hat{y} \end{cases} \tag{4.5}$$

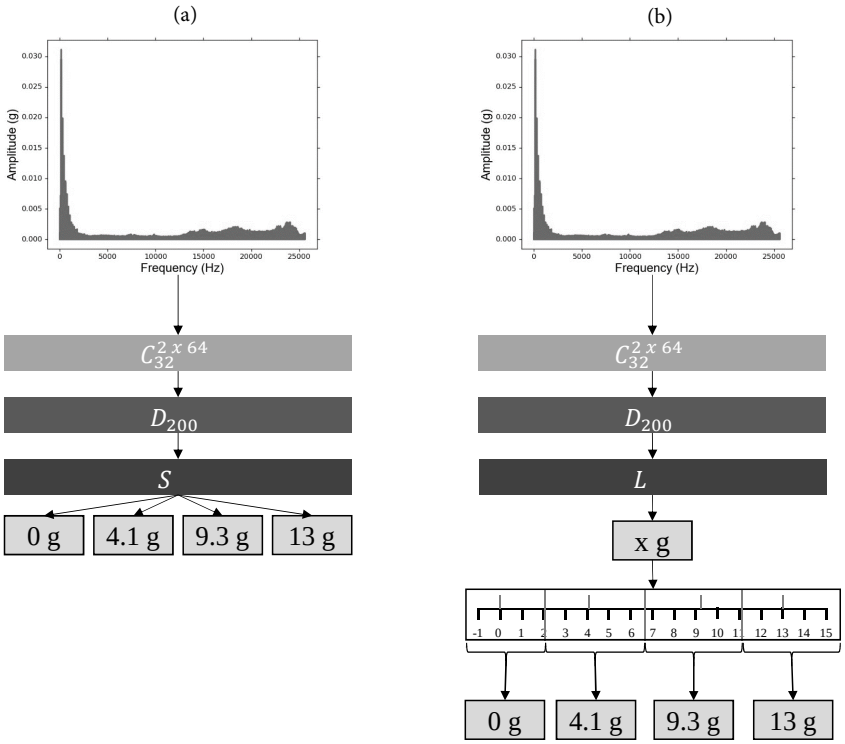


Figure 4.17: Block diagrams of (a) the original architecture where S indicates a fully connected layer with the softmax activation function, and (b) the modified architecture to solve imbalance detection. In architecture (b), L indicates a fully connected layer with linear a activation unit. Discretization is subsequently applied to the continuous value.

4.6 Discussion

In this discussion we briefly reflect on the added value of getting insights into the convolutional neural networks and the implications of this technique on the required expert knowledge for fault detection.

When using feature engineering, especially on infrared thermal video, good results can be achieved as is demonstrated on our two data sets. However, to improve a feature-engineering-based approach insights into the machine condition and how these manifest are required. When it comes to vibrations, already a substantial amount of knowledge is available which can be used to create more advanced features. However, for infared thermal data new insights are necessary to improve feature engineering-based fault detection. Convolutional neural networks, and in general feature learning, can aid to acquire these insights in a data-driven manner.

Feature learning enables a data-driven solution for fault detection resulting in less required domain knowledge. However, more knowledge is required on machine learning. Feature learning has the advantage of being applicable on different types of data, making it a solution that requires limited, but specific knowledge.

4.7 Conclusion

In this chapter, the potentials of feature learning are illustrated. For both the vibration measurements as well as the infrared thermal imaging data, convolutional neural networks are researched, developed and trained. Convolutional neural networks are end-to-end machine learning systems that do not require engineered features as input, but rather raw data.

For the vibration measurements, we have created a shallow convolutional neural network which is designed to processes the frequency spectra of the two acceleration signals simultaneously. For the infrared thermal imaging data, a deep convolutional neural network is constructed as a result of transfer learning. We show that transfer learning, using a deep neural network trained on natural images, can be applied on thermal infrared data, resulting in large improvements regarding infrared thermal imaging-based fault detection on our data sets.

By combining both the infrared-based convolutional neural network and the vibration-based convolutional neural network, a multi-sensor system is created. The system does not require features to be engineered. The main advantage of such a system is that for conditions for which no physics-based features exist, feature-learning can be employed. This is especially useful if the expert does not know what the condition is, causing for example under-performance, but does want to be able to detect it in the future.

Finally, insights into the convolutional neural networks trained on infrared thermal imaging are given. We have shown that the networks looks at specific

parts of the bearings' housing for specific faults that can possibly be related back to the underlying physics. By using methods to gain insights into the convolutional neural networks, human-interpretable knowledge can possibly be extracted about the physics of machine faults.

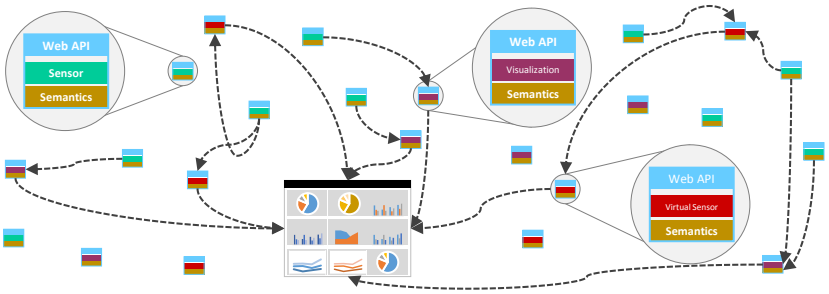
The work presented in this chapter has lead to the following publications, or are submitted as follows:

- Olivier Janssens, Viktor Slavkovikj, Bram Vervisch, Kurt Stockman, Mia Loccufier, Steven Verstockt, Rik Van de Walle, Sofie Van Hoecke, *Convolutional Neural Network Based Fault Detection for Rotating Machinery*, Journal of Sound and Vibration, vol. 377, pp. 331-345, 2016
- Olivier Janssens, Rik Van de Walle, Mia Loccufier, Sofie Van Hoecke, *Deep Learning for Infrared Thermal Image Based Machine Health Monitoring*, IEEE/ASME Transactions on Mechatronics, **Accepted for publication**

Chapter 5

Fleet level: Dynamic dashboard

Chapter highlights



In the previous chapters, systems were proposed for performance monitoring and fault/condition detection for offshore wind turbines. The contribution of this chapter is a dynamic dashboard wherein these systems can be deployed as virtual sensors. Additional to these virtual sensors, also real sensors can be used in the dashboard. The goal of the dynamic dashboard is to dynamically visualize values from these sensors. This is done by web-enabling the sensors and augmenting them with semantic information, hence enabling the dashboard to visualize sensor data dynamically.

5.1 Introduction

On average, a Belgian offshore wind park contains 36 wind turbines [9]. To easily monitor the performance of many wind turbines individually, a data-driven approach as proposed in Chapter 2 is suitable enabling power production prediction to be tailored to every individual wind turbine. This is necessary as due to degradation, location and maintenance the performance of two turbines will not be the same [103]. These individual models can be constructed in a limited amount of time with little effort and expertise knowledge. Furthermore, they only require data from the wind farm. Additional to monitoring the performance of wind turbines in the wind farm, the methods in Chapter 3 and 4 enable the condition of wind turbine inner components to be monitored. However, visualizing the outputs of all these models for an entire wind park is a daunting task but required in order to monitor an entire fleet. Hence, the contribution of this chapter is a dynamic dashboard that enables models such as described in Chapter 2, 3 and 4 to be put in to practice. A model is transformed into a virtual sensor for which the dashboard can visualize values. Additional to the virtual sensors, there are also a vast number of sensors that produce data in a wind farm. The dynamic dashboard proposed in this chapter is also able to use these data and visualize it. However, optimally visualizing the data is a difficult task. Hence, the dynamic dashboard can automatically reason on the available (virtual) sensors to find suitable visualization options.

Some examples to illustrate the usefulness of such a dynamic dashboard are:

- Instead of plotting the power production of every wind turbine in separate time series plots, the dynamic dashboard can automatically recommend to visualize them in a graphical map, displaying wind turbines and the produced power next to them. Hence, the power production of the entire fleet is summarized in one plot (e.g. Figure 5.1).
- To do anomaly detection, the regression models described in Chapter 2 can be thought of as a virtual sensor. A virtual sensor produces the same value as an actual sensor but is in fact not a physical sensor. For example, a virtual sensor can provide the output power given data from other sensors (i.e. yaw, wind speed, wind direction, pitch, temperature,...). The dashboard can automatically provide these required values to the virtual sensor without the need of human interference. Furthermore, the dashboard can subsequently compare the predicted value to the actual sensor value and see if the values are the same. If the actual value deviates from predicted value an anomaly has occurred which can be brought to the operators attention automatically (e.g. Figure 5.2).
- When an anomaly has occurred at the machine-level (i.e. an unexpected performance drop) it can be because of many reasons. By monitoring the con-

dition of components, the operator can rule out potentially damaged components if the fault detection systems have not detected anything. In the end the operator can start to explore possible options of the root-cause of the problem by making use of the intelligent visualizations in the dashboard. If for example the operator suspects yaw misalignment, the operator can start of by visualizing the wind direction. The dashboard will automatically provide the option to visualize the wind direction together with the turbine's orientation in a suitable plot (e.g. Figure 5.3).

- Measuring how much imbalance a rotor of the wind turbine has cannot be done directly. No sensor exists that measures imbalance. However, the regression-based convolutional neural network proposed in Chapter 4, Section 4.5.4, can predict the weight imbalance using vibration measurements. Hence, this approach enables virtual sensing and can be implemented as a virtual sensor in the dynamic dashboard. An example of how this imbalance can be visualized using a time series diagram can be seen in Figure 5.4.

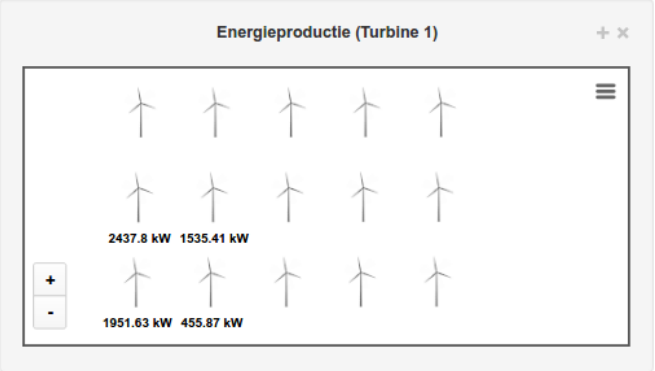


Figure 5.1: Overview visualization of the power production of the wind turbines in a wind farm.

To enable such a dynamic dashboard, the Industry 4.0 design principle of interoperability/interconnection is required [104]. The idea behind this design principle is to enable machines, devices, sensors, software services to connect and communicate with each other via the Internet of Things (IoT). In this chapter the IoT vision is implemented by encapsulating sensors –both physical and virtual– in RESTful web APIs hence transforming them into web-connected devices as described in [105]. Restful web APIs provide a uniform way of accessing data such as sensor values. To enable dynamic visualizations, also the visualization services

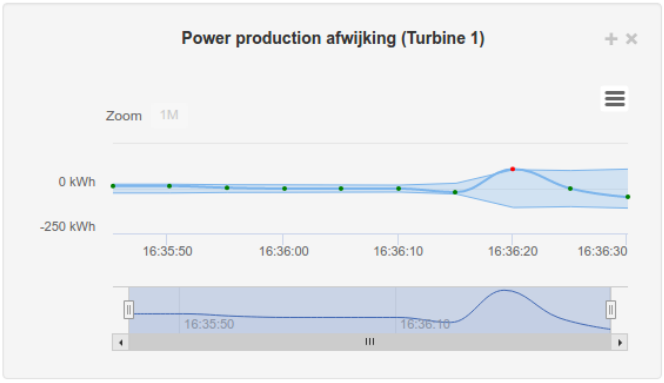


Figure 5.2: Anomaly detection on the power production of a wind turbine.

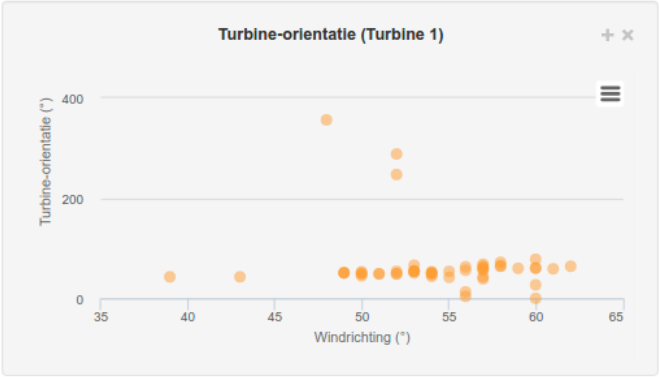


Figure 5.3: Scatter plot suggested by the dashboard. Direction of the wind versus the turbine's orientation for yaw misalignment diagnosis.

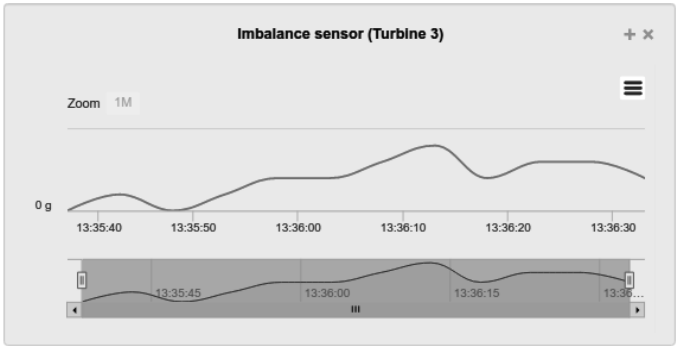


Figure 5.4: Example of how the virtual sensor for imbalance can be visualized in the dynamic dashboard.

are encapsulate in web APIs. All web APIs are semantically annotated[1] and by using Semantic Web reasoners[2], advanced sensor and visualization compositions can be created. These compositions result in sensible visualization instantiations in the dashboard, possibly enabling the detection of complex events that previously would have remained undetected.

In the next section the related literature regarding industrial applications of the Internet-of-Things is provided together with information on commercially available dashboard systems. Afterwards, the inner workings of the dynamic dashboard are discussed and how it is applied on a fictional wind farm. Finally, the evaluation of the dashboard is presented together with a conclusion.

5.1.1 Related work: Internet-of-Things

In recent years, several IoT applications have emerged as is described in the survey by Da Xu et al. [106]. Applications can be found in industries such as manufacturing, healthcare, automotive and mining. However, issues that often return are the lack of a commonly accepted service description languages and heterogeneity of data and communication protocols [107]. This makes the development and integration of physical objects into value-added services difficult as they can be incompatible with different communication protocols or/and will not be discoverable [106]. The Industry 4.0 design principle of interoperability/interconnection [104] solves

[1] Semantically annotating means that a description of the sensor is made, detailing the type of sensor, what output is provided and the possible input that is required (in case of virtual sensor). This semantic description is also made available via the web API.

[2] A semantic web reasoner takes semantic descriptions and draws conclusions based on these descriptions. For example, if the web API of a virtual sensor requires a temperature value as input, the semantic reasoner can use its reasoning the select the right web API that will provide this value.

this shortcoming by enabling machines, devices, sensors, software services to connect and communicate with each other via the Internet of Things (IoT). Restful web APIs can be used to implement this design principle by providing a uniform interface for accessing the sensors [105,108] . By encapsulating a device/sensor in a web API, the HTTP functionality can be used, making also the communication protocol uniform. Requesting a value from a sensor can then simply be accomplished by using a HTTP GET request. Using restful web APIs thus solves both the problems of data and communication protocol heterogeneity.

5.1.2 Related work: Dynamic visualization systems

Data visualization has been researched and commercialized in recent years by for example dashboard.io[3], QlikView[4] and Tableau[5], but still requires the end-users to have detailed knowledge of the data and to handcraft visualizations. To support automated crafting and integration, the use of semantics is required. Current state-of-the-art research solutions using Semantic Web technology and/or reasoning in dashboards are more low-level (requiring SPARQL[6] queries to be written) [109–112].

In the dynamic dashboard proposed in this chapter, dynamic composition of visualization services and sensors (both real and virtual sensors), is possible by using Semantic Web technology and reasoning. The dynamic dashboard automatically suggests correct visualization options for specific types of data.
In the next section the architecture and the components of the proposed dynamic dashboard are discussed.

5.2 Methodology

A high level architecture of the dynamic dashboard can be seen in Figure 5.5. The dashboard has several components which are required to dynamically combine sensors, virtual sensors and visualization services.

[3] `https://dashboard.io`
[4] `www.qlik.com`
[5] `www.tableau.com`
[6] Semantic query language for databases, able to retrieve and change data stored in resource description framework (RDF) format.

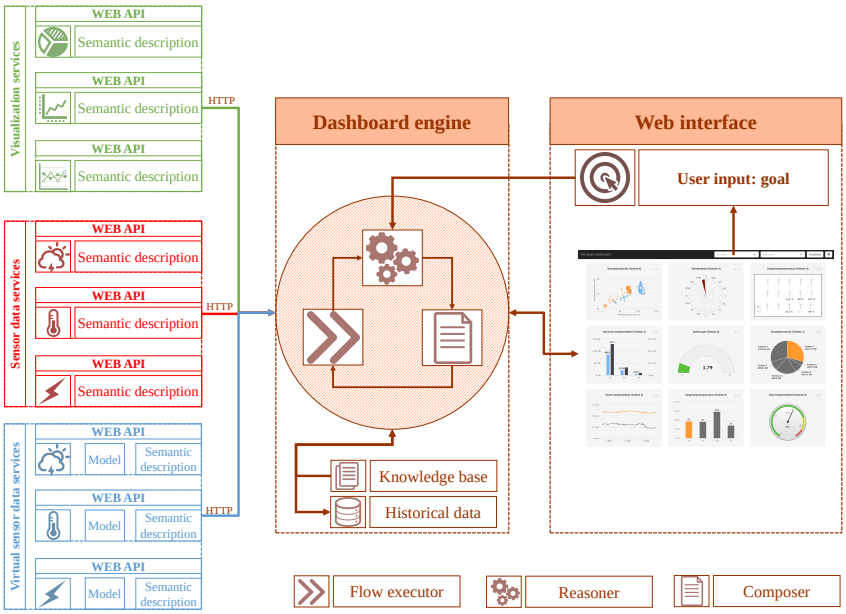


Figure 5.5: Platform for dynamic visualization of multi-sensor architectures.

5.2.1 Sensor data services

Sensors are encapsulated in web APIs to cope with the heterogeneous data representation standards and communication protocols. A sensor value can hence be requested using a HTTP GET request as can be seen in Figure 5.6. A web API is semantically annotated which can be requested using a HTTP OPTIONS request. A semantic description specifies the functionality of the web API. Using semantic annotations and reasoning, a higher level coupling can be achieved without any additional configuration required and the system will be less application specific.

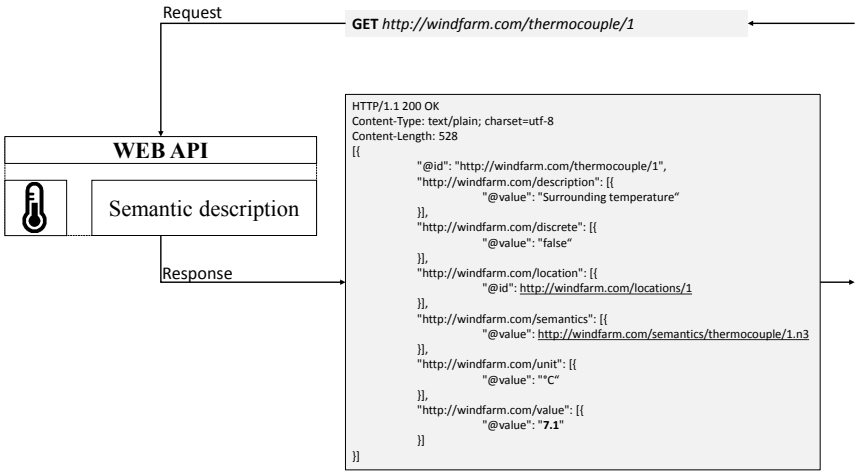


Figure 5.6: A value of a sensor can be requested using a HTTP GET request for which a response will be returned containing the required value.

Because of the confidentiality of the data of existing wind farms, a virtual offshore wind farm with 15 turbines (each turbine containing 9 sensors and one virtual sensor) is simulated. Three major sensor categories are created, each but one producing fictitious sensor values. In order to truthfully simulate the wind farm, the values are restricted and depend on the values of related sensors.

The sensor data is semantically annotated using json-LD. Json-LD provides mappings from json to the RDF model using the context concept, linking json object properties with ontology concepts [113]. An example of the sensor data in json-LD can be seen in Listing 5.1. As can be seen, more than just the sensor value is provided. Additional (semantic) information is provided which can be used. If for example a sensor value in “Bar” is required instead of “Pascal”, then using the additional semantic information a check can be done to make sure that the web API provides sensor values in the required unit.

The semantic description of the web APIs functionality itself, is done using

RESTdesc [114]. RESTdesc is designed for compactness. An example of such a description can be seen in Listing 5.2. This semantic description specifies that the resource is an air pressure sensor and that if you want an air pressure observation (a single measurement), then you can execute a GET request to the resource's URI.

```
[{
    "@id": "http://windfarm.com/
            air_pressure_sensors/1",
    "http://windfarm.com/description": [{
        "@value": "Surrounding temperature"
    }],
    "http://windfarm.com/discrete": [{
        "@value": "false"
    }],
    "http://windfarm.com/location": [{
        "@id": "http://windfarm.com/locations/1"
    }],
    "http://windfarm.com/semantics": [{
        "@value": "http://windfarm.com/semantics/
        air_pressure_sensors/1.n3"
    }],
    "http://windfarm.com/unit": [{
        "@value": "Pa"
    }],
    "http://windfarm.com/value": [{
        "@value": "101,325"
    }]
}]
```

Listing 5.1: Example of the data (in json-LD) returned by a sensor web API.

```
@prefix ex: <http://example.org/>.
@prefix http: <http://www.w3.org/2011/http#>.

<http://windfarm.com/air_pressure_sensors/1>
    a ex:AirPressureSensor.

{
    ?sensor a ex:AirPressureSensor;
        ex:value ?value;
        ex:unit ?unit;
        ex:location ?location;
        ex:description ?description;
        ex:discrete ?discrete.
}
=>
```

```
16 {
17     _:request http:methodName "GET";
18               http:requestURI ?sensor;
19               http:resp [ http:body _:observation ].
20     _:observation a ex:AirPressureObservation;
21                   ex:value ?value;
22                   ex:unit ?unit;
23                   ex:location ?location;
24                   ex:description ?description;
25                   ex:discrete ?discrete.
26 }.
```

Listing 5.2: Example of semantic description in restdesc for a air pressure sensor.

5.2.2 Virtual sensors

Beside real sensors, it is also possible to have virtual sensors. Virtual sensors are created to emulate a physical sensor and can be used in machine fault detection techniques, such as anomaly detection. As described in Chapter 2, a machine learning algorithm can be used to train a model that can predict the power output of a wind turbine. This model requires data from physical sensors. The model is created offline manually, and is then encapsulated in a web API and semantically annotated to deploy it as a virtual sensor, similar to the process for physical sensors. To use such a virtual sensor one has to do a HTTP GET request to the virtual sensor's web API including a value of the wind speed. The web API will subsequently return a prediction for the power output. For the dynamic dashboard a real sensor and a virtual sensor are indistinguishable. The only difference exists in the fact that the former does not require input values and the latter does.

5.2.3 Visualization services

The main responsibility of the visualization services is to visualize the submitted (virtual) sensor data. Many visualization libraries exist, and it suffices to enclose one of them in a web API. For the proof-of-concept dashboard, the HighCharts JS [7] library is used to create nine different visualization service web API, but any other JavaScript-based visualization could be used. On request, a visualization service provides source code to the dynamic dashboard resulting in a visualization widget in the dashboard. To update the generated visualization, an update function, to which real-time data can be provided, is also present in the generated source code. This update function is called periodically, with a frequency specified by the current user.

[7] https://www.highcharts.com

5.2.4 Broker component

The broker component has several tasks. The first task is to retrieve, and afterwards store, all available visualization and sensor web API URIs. To retrieve the URIs a discovery mechanism is used, enabling a loosely coupled system. The discovery mechanism consists of performing a GET request to both the domain name of sensor web APIs and to the domain name of the visualization web APIs. A list of available sensor web APIs or visualization web APIs is subsequently returned.
It should be noted that the discovery mechanism also actively monitors the statuses of discovered sensors by the use of pinging. Alternatively, sensor discovery can also be done passively by listening to a sensor's heartbeat (i.e. periodical value change). A sudden disconnection may indicate a component failure, so in this case all dashboard users are notified.
The second task is to get sensor resources and match them to suitable visualization APIs. To do this several sub-components are required.

- **Reasoner**: The reasoner takes semantic descriptions and will suggest options. It is used to confirm if a goal is completed, to recommend visualization options and to help discover new options to explore. The reasoner provides a proof (i.e. set of instructions) which is given to the composer. The EYE reasoner is used in the dashboard [115].

- **Composer:** The composer uses the proof and composes a set of HTTP requests to execute the proof.

- **Flow executor:** The flow executor is a small component that uses the composed requests and executes them.

- **Knowledge base (kb)**: The kb contains central semantic descriptions of logical rules (described using RESTdesc) which is required to perform sensor and visualization composition. A first description contains an ontology describing the taxonomy hierarchy of the sensors and the visualization services. A second description lists the possible visualization options for the different types of sensors. Note that reasoning can already be done using these two descriptions. For example, using the ontology it can be deduced that a temperature sensor provides real values (i.e. it is a real-valued sensor). Using the second description it can be deduced that if a sensor provides real values, a time series visualization web API can be employed. A third description describes what the related sensors are of a senor, e.g.: wind direction is related to the pitch. This description enables the broker to recommend additional sensors to be visualized. Finally, the kb also contains descriptions of goals. These are very brief and solely state what kind of resource is required.

- **Historical database (hdb)** The hdb is responsible for the storage of historical data, as sensors generally have no or limited storage. The Semantic Web data model is closely connected to the relational database model [116]. When the current visualization service is able to display historical data, the HDB has to be considered as a primary (pseudo) API.

The reasoner, composer and flow executor are used in a generic broker algorithm that is able to retrieve a sensor resource from a web API. The algorithm is discussed in more depth in the next section.
Using the sensor resource, semantic description of the sensor web API, the kb and visualization web API semantic descriptions, the reasoner will provide options on how the sensor web API's values can be displayed in the dashboard. This process will not return a visualization instantiation to the user, but rather a list of possible visualization options from which he/she can choose.
When the user selects a visualization option from the suggested possibilities, the broker's third task is to put it into action. The reasoner, composer and flow executor are employed to construct and execute the appropriate GET request containing the sensor data. In this phase also historical data from the hdb can be utilized. Afterwards the visualizer service generates the desired source code which results in a visualization instantiation on the dashboard.
Once the connection between data and visualization services is established, it can be exploited unaltered in order to facilitate the stream of up-to-date sensor data. Accordingly, no reasoning is required for the update process.
As sensors and visualization services are already implemented as web APIs and a platform independent application is desired, a web application implementation as dashboard is preferred. The dashboard in its most elementary shape has a search bar, allowing the user to browse the available entities and select the required sensors. Requested visualization instantiations are allocated to widgets on the dashboard as can be seen in Figure 5.7.

Figure 5.7: Platform for dynamic visualization of multi-sensor architectures.

5.3 Internal Design Details

Below is an overview of the main platform design details.

5.3.1 Proposed reasoning algorithm for sensor data retrieval

To visualize the data stream coming from a sensor, several web APIs have to be chained after one another. Especially when considering a virtual sensor, as virtual sensors need a sensor that provides the input value for the prediction, and a web API that is able to make a prediction for said value. Instead of providing the broker component with a description of which web APIs should be queried one after another, we let the broker discover the web APIs itself and combine them. To do this, a generic mutual recursive algorithm is devised that can serendipitously discover new web APIs. The algorithm combines the reasoner, composer and flow executor. A detailed illustration of the specific steps of the algorithm is provided in Figure 5.8. The first function of the mutual recursive algorithm will query a given URI, and by using a goal retrieved from the knowledge base it will check if said goal is reached. Sensor resources have hypermedia links[8] to relevant web APIs. Hence, the broker will subsequently use the semantic description of the web API and the sensor resource to try to find a new web API to query to achieve its goal. This new URI is given to the second recursive function that immediately calls the first recursive function again using the newly discovered URI. Only when no new URI is discovered will the rest of the second recursive function be executed. This remaining part of the second recursive function will backtrack the steps (i.e. query previously queried web APIs) and use its discovered knowledge (i.e. resources), to check if new options become available. This is done because some web APIs require input values and can only be queried if the required resources are available. If new options become available, the second recursive function will execute a function call to itself with the new URI. Only when no new URIs are discovered or if the goal is met, will the mutual recursive functions start to return values that the dashboard requires.

[8] Hypermedia links are URIs that refer to web APIs that can be requested next.

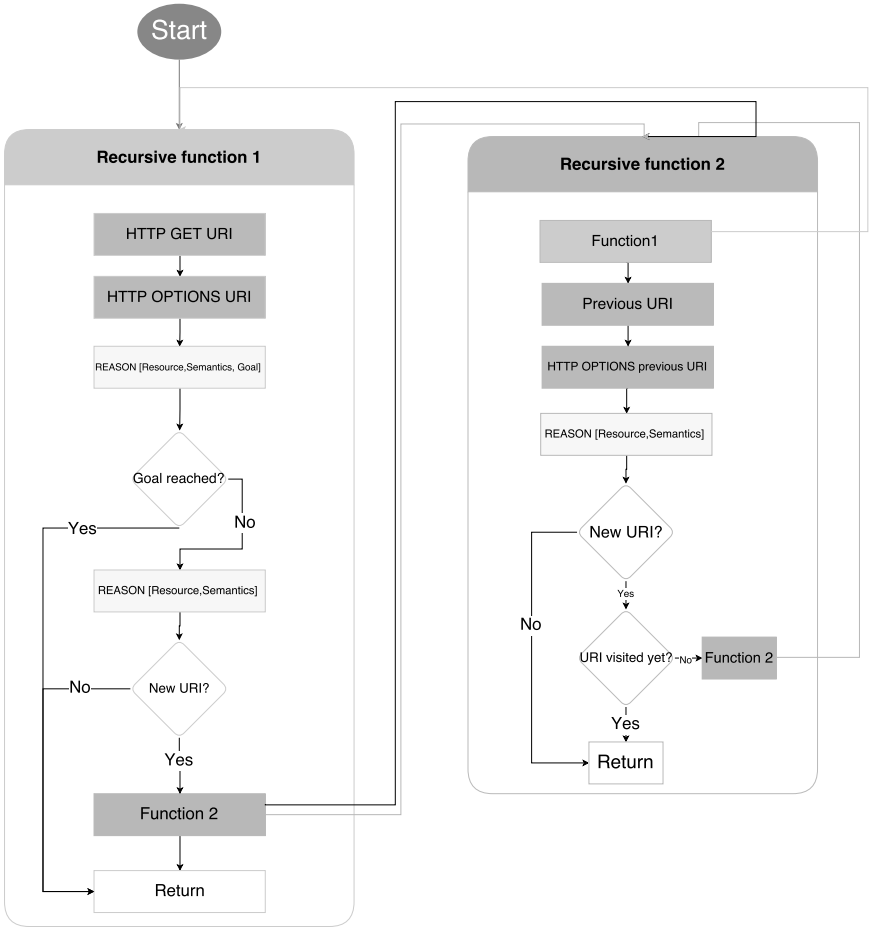


Figure 5.8: Proposed reasoning algorithm for sensor stream visualization illustrating the steps to discover new web APIs and achieve a predefined goal.

5.3.2 Reasoning algorithm: virtual sensors

When dealing with real sensors, the reasoning algorithm for sensor data retrieval will finish quickly as the goal is met after the first GET request. However, as can be seen in Figure 5.9, virtual sensors require more steps:

1. The process starts with a user selecting a virtual sensor in the dashboard. The broker will fetch the required goal from its knowledge base. The goal description specifies what kind of resource is required in the end (i.e. a prediction of the power production in this case).
2. Next, the broker executes a GET request to the web API of the virtual sensor. This web API does not return the required sensor value as it does not have the required input to predict the output. The virtual sensor rather gives a response which is a resource containing another URI (i.e. hypermedia link) to the web API of another sensor (which is the sensor that will provide the input value).
3. The broker subsequently uses a HTTP OPTIONS request to the web API to get the semantic description of the functionality of the web API.
4. The reasoner uses the predefined goal, semantic description and the resource to see if the goal is met, which is not the case.
5. Next, the reasoner combines the virtual sensor resource and its semantic description to provide an option of what a possible next step could be to reach the goal. The semantic description specifies that if the resource has a URI to an input sensor, there exists a GET request resulting in a response containing a value (the required input value) and also a URI to the web API that will be able to make a prediction. The result of the reasoning process is a detailed instruction (i.e. proof, for which an example can be seen in Listing 5.3) what could be the next step. In this case the proof specifies that a GET request can be done to the hypermedia link in the resource.
6. The broker's flow executor will subsequently execute the aforementioned GET request. The GET request will return a resource of the required input sensor containing a value and another hypermedia link.
7. The broker will also do an OPTIONS request to this web API to fetch its semantic description. The semantic description clarifies that if the resource contains a URI to a prediction web API, there exists a GET request to that URI that requires an input parameter and will return a value (i.e. the prediction).
8. Again, the broker will first check if its goal has been met, which is not the case as it does not posses a resource containing a prediction value.

9. Next the reasoner will again use the semantic description and the resource to figure out what the possible next step could be. The reasoner provides a detailed instruction specifying that a GET request can be done to the hypermedia link. The GET request should also contain a (input) parameter.

10. The broker's composer will use the input parameter and construct the required GET request, which is executed by the flow executor. This request will result in a response containing a prediction resource including the predicted value.

11. To fetch the semantic description, an OPTIONS request is done again.

12. Finally, the reasoner will again check if the goal has been reached, which is the case and the cycle will end.

    As with a real sensor, the broker ends up with a sensor resource. Hence, in a next step the dashboard uses the reasoner to find possible visualization options from which the user can choose. When the user has made a decision the reasoner, composer and flow executor are used to combine the (virtual) sensor and a visualization resulting in a widget in the dashboard.

As can be seen, there is no predefined path on how to get a value from a virtual sensor. The broker will start of with one resource that contains a URI to another resource and so on. This procedure enables the broker to serendipitously find its own way towards its goal. This has several advantages. First, if a URI would change, the broker will automatically discover this making the system loosely coupled. Second, one can imagine that a virtual sensor only takes one value as input, however, it is also possible that another virtual sensor can take multiple values as input. The first resource can contain URIs to multiple other resources which all can be queried. If there exists a prediction web API that only requires one input value, the broker will reach its goal as soon as it has discovered the correct input value and a matching prediction web API. However, if there only exists a web API that requires multiple input values, the broker will continue to follow URIs until it has all the required input values for the prediction web API. This is because the broker will follow every URI until there are none left or until its goal is achieved. Third, the dashboards user could specify a new goal in the kb. This will not cause the dynamic dashboard to malfunction. The broker will still try to reach its goal given the initial resource from which it can start its search.

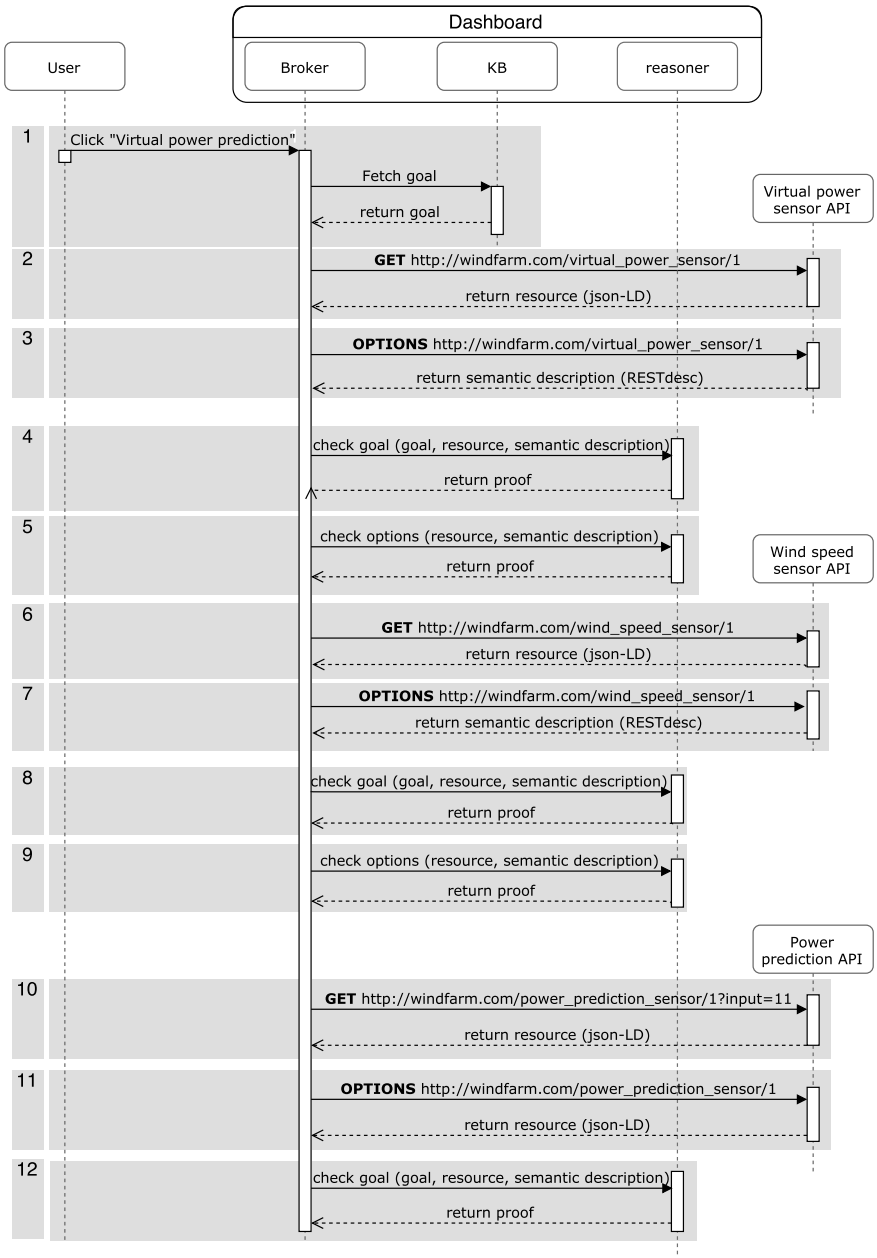


Figure 5.9: Sequential diagram illustrating the different steps the algorithm will execute to discover the required resource.

```
1  PREFIX http: <http://www.w3.org/2011/http#>
2
3  _:sk0
4      http:methodName "GET".
5
6  _:sk0
7      http:requestURI
8          "http://windfarm.com/wind_speed_sensors/1".
9
10 _:sk0
11     http:resp _:sk1.
12
13
14 _:sk1
15     http:body _:sk2.
16
17
18 _:sk2
19     a <http://windfarm.com/WindSpeedSensor>.
20
21 _:sk2
22     <http://windfarm.com/PowerProductionPrediction>
23         _:sk3.
24
25 _:sk2
26     <http://windfarm.com/value> _:sk4.
27
28 _:sk2
29     <http://windfarm.com/unit> _:sk5.
30
31 _:sk2
32     <http://windfarm.com/location>
33             <http://windfarm.com/locations/1>.
34
35 _:sk2
36     <http://windfarm.com/discrete> _:sk6.
```

Listing 5.3: Example of the output of the reasoner. The proof details that a GET request (_:sk0) can be done to a wind speed sensor which will return a wind speed sensor resource(_:sk2), which will have a hypermedia link to a power production prediction sensor (_:sk3). Furthermore, the wind speed sensor resource will have a value (_:sk4), a unit (_:sk5), a location and will be indicate if its a discrete value or not (_:sk6).

5.3.3 Reasoning algorithm: anomaly detection

Within this section, anomaly detection is implemented in order to further clarify the advantage of serendipitous resource discovery.
As discussed in Chapter 2, if a wind turbine performs sub-optimally, its generated power will change unexpectedly. Such a change is defined as an anomaly. More specifically, if the signal deviates from its expected behavior, the deviation can be labeled as an anomaly. It is safe to say that anomaly detection is crucial when it comes to monitoring many wind turbines, as a single individual (i.e. an operator) is not able to monitor and interpret all signals. By employing anomaly detection a user will be notified of deviating behavior which requires no effort from the user's side.
Anomaly detection itself requires two main components. First, a virtual sensor is required. This virtual sensor (i.e. machine learning model) is trained using data from normal operational conditions and will therefore be able to emulate this normal operational condition. Second, a deviation detection module is required that takes a real sensor value as input, and a matching predicted value. Next, the module will compare the two values and see if there is a deviation. If the machine is operating in normal operational conditions, the deviation will be 0 or very close to zero. However, if the machine is performing not as expected, there will be a large deviation, i.e. anomaly. There are several methods to quantify this deviation and subsequently determine if the deviation is anomalous. To illustrate how anomaly detection can be done in the dynamic dashboard, a common method from statistical process control is used. The predicted value is subtracted from the real value resulting in a residual value. Subsequently, a check is done to see if this residual is larger or smaller than three times the standard deviation calculated using a distribution of residuals during normal operation conditions. 99.7 % of the residuals during normal operation conditions lie within this three standard deviations range. Hence, it is very unlikely that a residual will fall outside this range. Therefore, if a residual exists outside the range, it is labeled as an anomaly and is cause for further inspection.

5.3.4 Anomaly detection in the dynamic dashboard

Anomaly detection in the dynamic dashboard uses the same mutual recursive algorithm as for the other sensors, however additional steps are executed. First, as with other sensors, the user selects anomaly detection in the dashboard. This can be done by selecting it in the drop-down list or by activating it in the options screen of an existing widget. By activating anomaly detection, a goal is retrieved from the knowledge base and a GET request is done to the anomaly detection URI of that sensor's web API. The response will not provide the required resource because several steps need to be done beforehand. Instead, the response contains

three URIs (i.e. hypermedia links). The first URI is to the virtual sensor described above, the second URI is to the actual physical sensor and the third URI is to the deviation detection module. The broker will execute an OPTIONS request to anomaly detection web API, which will return the semantic description of the anomaly detection module. The reasoner will then check if the goal has been met, which is not the case. The broker subsequently will use the reasoner to figure out its options as before. The first option is the rule that if the resource contains a URI to a virtual sensor, a GET request can be done to that URI and a response will be returned containing a URI to the web API that can give the input value for the virtual sensor. In fact, these are the steps described above to get a value from a virtual sensor. For brevity the steps will not be repeated here. In the end the broker will end up with a prediction from the virtual sensor which is a resource. The initial anomaly detection resource has two other URIs. As the goal has not been reached yet, the broker will continue to serendipitously explore the URIs to find its goal. The second URI is to the web API of the actual physical sensor. The broker's composer will construct a GET request and the flow executor will execute it. The request returns a response containing among other properties, the value from the physical sensor. The broker fetches the semantic description by executing an OPTIONS request. Next, the broker will again use the reasoner to check if the goal has been reached. Which is not the case. Although the broker has a prediction and a real value, the deviation detection step is still required. The broker uses the semantic description and the resource to plan its next step. The reasoner will return that there are no more available options. Hence, the broker has run into a dead end. As with the virtual sensor, the broker will back-track its steps and with the new information, investigating if new options have become available. Eventually, the broker will use the semantic description of the anomaly detection module, the virtual sensor resource and the physical sensor resource in the reasoning process to see if a new option has become available. Because the broker has these two resources, the reasoning process reveals that a new option has become available. The broker has serendipitously discovered that there exists a GET request that requires two input values and will return a response which is a deviation detection resource. This resource contains a value (i.e. the residual) and the standard deviation. The broker's composer creates a GET request with the two input values and the flow executor executes this GET request. The broker consequently receives the aforementioned response. The broker will use the reasoner to check if the goal has been met, which is true. As the goal has been reached the reasoning process is done.

As with the physical sensor and the virtual sensor, the broker ends up with a sensor resource, which in this case contains a residual and a standard deviation. As before, the broker will find suitable matching visualization options. The user will receive several options which he/she can select. In a matching visualization ser-

vice a check is implemented as described in Equation 5.1 where x is the residual value and σ the (running) standard deviation. Depending on the visualization, the data point can for example be colored red, or a notification can be placed in the dashboard to alert the user.

$$f(x) = \begin{cases} \text{anomaly}, & \text{if } x > 3\sigma \text{ or } x < -3\sigma \\ \text{no anomaly}, & \text{otherwise} \end{cases} \tag{5.1}$$

By not tightly coupling all the web APIs, a more powerful dashboard is created. For example, someone can decide to create a deviation detection web API that requires a real value and a prediction value and which returns the residual, interquartile range (IQR), Q1 and Q3 [9]. The dashboard can automatically discover this web API and use it in its reasoning process. No modification has to made to the web API or semantic descriptions of the physical or virtual sensors. Moreover, if a web API exists that uses the residual, Q1, Q3 and IQR, the dashboard will automatically recommend this visualization option without requiring additional coding work.

5.4 Implementation

All components are implemented using the Ruby on Rails (RoR) MVC framework. A RoR application offers a full web application stack, and follows a restful, resource-oriented approach. A proper web server is embedded, so besides a working Ruby installation, no additional software is required. RoR provides a database by default and encourages the use of web standards (such as json) for data transfer and html, css and JavaScript for user interfacing.

5.4.1 Resulting dynamic dashboard

Figure 5.7 presents a possible dashboard configuration for monitoring offshore wind farms. Widgets can be easily added and/or removed in a straightforward way. Every widget holds the descriptions of the containing sensors and has a menu to adjust the visualization and according options. Thanks to the semantic descriptions of all sensors and visualization services, related sensor data can also be easily selected from these widgets and added to the visualization.

5.4.2 Fault detection

It should be noted that fault detection is also possible in the dashboard. The systems described in Chapter 3 and 4 are also easily integrated in a web API similarly

[9] Another anomaly detection method is to check if $Q_1 - 1.5IQR < x < Q_3 + 1.5IQR$

to as described for performance monitoring of Chapter 2. To do this, virtual sensors can be created that provide features extracted from measurements of certain components. Furthermore, the classification algorithms can be encapsulated in a web API so that the broker can automatically discover the virtual sensors that provide features and couple them to the right web APIs that will provide a classification. This classification can subsequently be placed in a notification screen in the dashboard. In Figure 5.10 the composition of fault detection using the dashboard is depicted. Similarly, for fault detection using feature engineering and vibration measurements the composition can be seen in Figure 5.11. Using the feature-learning approach the compositions become even more simplified as can be seen in Figure 5.12. For example, the convolutional neural network from Chapter 4 does virtual sensing. Hence, it can be implemented in the dashboard as a virtual sensor. The values predicted by the virtual sensor can subsequently be visualized by the dynamic dashboard. These compositions illustrate that the proposed approach to monitor many machines and components is a generic one.

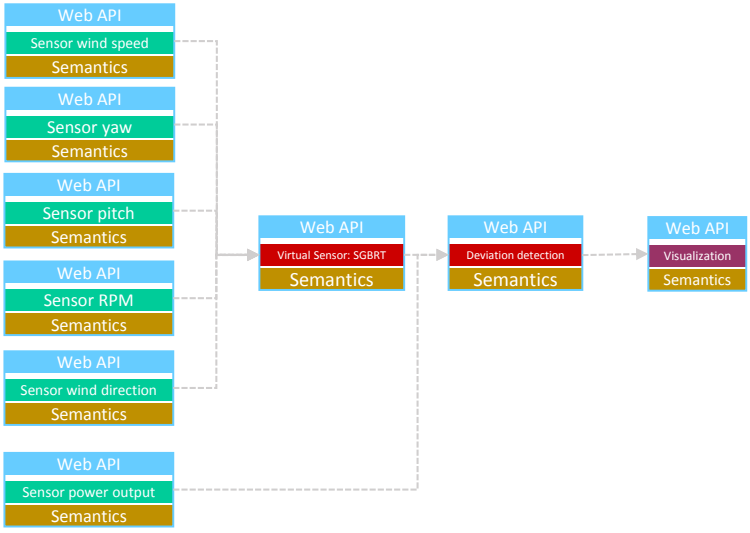


Figure 5.10: Service composition for fault detection.

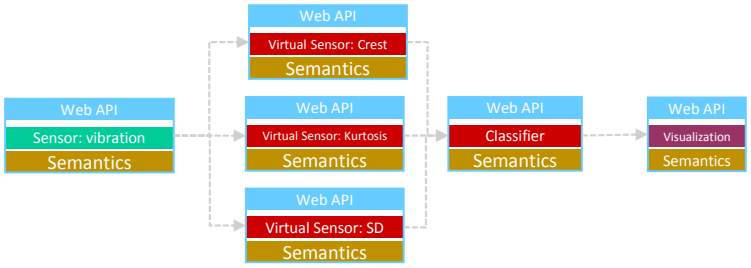


Figure 5.11: Service composition for fault detection using feature engineering applied on vibration measurements.

Figure 5.12: Service composition for component-level fault detection using feature learning applied on infrared thermal data.

5.4.3 Evaluation

The limited adaptability and static nature of traditional dashboards, as well as the automated detection of complex events and creation of advanced compositions, are the driving forces behind the design of the proposed platform. There is a need for dashboards that satisfy current user needs, in a user-friendly way and with minimal configuration required.
Data is fetched and visualized in soft real-time, so users can get no false sense of security. Real-time or soft real-time visualization of new data is achieved as individual sensor requests are independent, and therefore can be executed in parallel. As long as the total retrieval time does not exceed the user defined update interval, the user will not experience noticeable delays.
The reasoning process is the most time consuming, resulting in a few hundred milliseconds delay for average widget creation and dashboard configuration, which is acceptable as creating new widgets and configuring the dashboards only happens occasionally. The reason for this larger delay is an acceptable initialization and networking overhead (in which the required semantic descriptions are collected), but the actual reasoning process is the most time-intensive. Execution times could be reduced by providing the reasoner with additional directives and less generic linking logic. However, this undermines the dynamic nature of the system, which of course is its absolute strength.

5.5 Conclusion

Dashboards present and communicate the condition of monitored fleets to supervising experts. Contrary to static solutions for monitoring dashboards, the proposed platform enables dynamic data visualization.
By adopting the Internet of Things vision and implementing sensors as Web connected devices, semantically annotated using RESTdesc, the presented platform allows to precisely visualize the data produced by sensors in multi-sensor environments by dynamically generating meaningful service compositions. Such a dynamic dashboard application, combined with advanced failure detection mechanisms –such as virtual sensors and anomaly detection– proves to be a very powerful monitoring tool for complex, hard to access and/or critical environments such as wind parks. It enables its users to correctly monitor the condition of the environment and moreover, as a result of the meaningful sensor composition process. In the end dynamic dashboards facilitate condition monitoring of an entire fleet supporting predictive maintenance.

The work presented in this chapter has lead to the following publications, or are submitted as follows:

- Sofie Van Hoecke, Cynric Huys, Olivier Janssens, Ruben Verborgh and Rik Van de Walle, *Dynamic monitoring dashboards through composition of web and visualization services*, Proceedings of the 2nd EAI International Conference on Software Defined Wireless Networks and Cognitive Technologies for IoT, 2015
- Olivier Janssens, Mia Loccufier, Rik Van de Walle and Sofie Van Hoecke, *Dynamic Dashboards for Multi-Sensor Machine Monitoring Using Semantic Web Technologies*, Semantic Web Journal, **submitted to**

Chapter 6

Conclusion and future work

In this chapter a conclusion summary is provided together with possible future research directions.

6.1 Summary

Monitoring the health of wind turbines in a wind farm is a difficult task. In this dissertation this monitoring task is broken down into three levels to make the task more manageable: machine level, component level and fleet level. In order to estimate the overall condition of a wind turbine (machine-level), performance monitoring is done. However, this can be a daunting task as many factors influence the performance of a wind turbine. Creating physics-based models can require considerable expert knowledge, effort and possibly computational resources. As an alternative, data-driven performance modeling is researched within this dissertation. Up until now, these approaches were almost solely used to model the relation between the wind speed and the generated output power of the wind turbine. To improve data-driven performance modeling, in Chapter 2, both univariate and multivariate performance modeling approaches are researched. Several machine learning algorithms are evaluated using both a synthetically generated data set and a data set containing measurements from three actual offshore wind turbines.
First, we illustrated that univariate modeling can be improved using tree-based methods such as stochastic gradient boosted regression trees. Second, we showed that data-driven performance modeling can be improved considerably by using additional variables (i.e. rotations per minute of the rotor, turbine yaw, blade pitch, wind direction and wind speed) as input to the models in order to predict the generated power. Furthermore, we show that by using these tree-based algorithms, insights into the data can be gathered and that the models do in fact use information from these additional variables to predict the output power.
When a deviation is detected in the performance of a wind turbine the cause of it

is unknown. Hence, in Chapter 3 fault detection is done using feature engineering on the level of the components. In this chapter the focus lies on rolling element bearings, due to the fact that these often suffer from damages and hence result in the most down time of a wind turbine. The root-cause of many bearing failures can be linked to lubricant inadequacy. One of the purposes of lubricant is friction control. Hence, a feasibility study is done to see if infrared thermal imaging can be used to detect lubricant conditions.

A wind turbine is a complex machine, therefore, the feasibility study is done using a lab-scale set-up. Two data sets were created containing measurements of a multitude of conditions and faults. The first data set contains conditions such as outer-raceway faults, reduced lubrication, extremely reduced lubrication and also imbalance of the machine. The second data set contains conditions such as outer-raceway faults, hard particle contamination of the lubricant and several gradations of imbalance.

A system was created to detect the introduced conditions and faults using infrared thermal imaging data. To bridge the gap between the current knowledge regarding thermodynamics of bearing faults and applied condition monitoring, data-driven features were researched. In the proposed system, we showed that three specific features, i.e. moment of light, Gini coefficient and standard deviation are useful for infrared thermal imaging-based fault detection. From the feasibility study, which was conducted using a lab-scale set-up, we could conclude that infrared thermal imaging may be a very suitable method to detect conditions related to lubrication in rotating machinery such as wind turbines.

Similar to the infrared-based system, a vibration-based fault detection system was created. Vibration-based approaches have been proven to work well over the years and are the predominant fault detection tool in the wind energy sector. The vibration-based system in Chapter 3 uses both summary statistics (i.e. data-driven) and expert features.

In general, we observe that both the infrared thermal imaging-based and vibration-based approaches work well, but not perfectly. On one hand, the infrared thermal imaging-based system is less well able to detect outer-raceway faults. On the other hand, the vibration-based approach detects the different lubrication conditions less well. Hence, a multi-sensor system is designed and created that nullifies the respective weaknesses and results in an overall better performing system.

The component-level fault detection system proposed in Chapter 3 uses features created using thermal infrared data and vibration data. To create these features, knowledge is required on vibration analysis and image processing. Even though the systems performs very well, it requires substantial knowledge from the practitioner to create these features. Also, if new faults have to be detected, possible new features have to be created. To mitigate this disadvantage, in Chapter 4 a feature learning-based approach is proposed to detect the multitude of conditions and

faults. The technique employed for this purpose is convolutional neural networks. Convolutional neural networks are end-to-end machine learning systems that learn to transform the data to optimally represent it for the classification task. In Chapter 4, convolutional neural networks are researched and designed and subsequently applied to both infrared thermal imaging data and vibration data. For vibration data, a shallow convolutional neural network is developed which is able to detect many of the faults and conditions accurately. Infrared thermal data is a more complex form of data and hence requires a deeper convolutional neural network. However, training such a model requires a lot of data. Hence, in Chapter 4, transfer learning is researched enabling a deep convolutional neural network to be trained and used. We show that a deep convolutional neural network that is trained on natural images can be re-purposed in order to process thermal infrared images. Furthermore, the convolutional neural networks are able to detect the different faults and conditions more accurately compared to the feature engineering based approach which is proposed in Chapter 3.

In the end, a multi-sensor approach is proposed which uses the vibration-based convolutional neural networks and the thermal infrared-based convolutional neural networks, again slightly improving the multi-sensor approach. Additionally, we show that by using the infrared thermal imaging-based convolutional neural networks, insights into the infrared images can be gained by visualizing what the important areas in the images are for the convolutional neural network. Based on the data captured using our controlled lab-scale set-up, we can conclude that different areas in the thermal image can be linked to certain conditions using the trained convolutional neural networks. These insights are a first step towards helping to understand the thermodynamical effects in the bearings from a data-driven perspective.

In order to put the performance monitoring system and the fault detection system into practice, in Chapter 5 a dynamic dashboard is proposed. The dynamic dashboard enables to monitor the signals generated by an entire fleet of wind turbines. To be able to do this, research is done how the industry 4.0 principle of interoperability/interconnection can be implemented. The idea behind this principle is to enable machines, sensors and software services to connect and communicate via the Internet of Things. In Chapter 5, this design principle is implemented by encapsulating sensors in web APIs hence making them web-enabled. Furthermore, the machine learning models are transformed into virtual sensors and also encapsulated in web APIs.

To let the dynamic dashboard interpret what the web APIs can provide and do, they are semantically annotated. Hence, by using a semantic reasoner the dynamic dashboard can combine sensors and virtual sensors. Furthermore, by also encapsulating visualization services in web APIs and semantically annotating them, dynamic compositions of sensors, virtual sensors and visualization services are made

possible. This for example enables the dynamic dashboard to do anomaly detection for every single wind turbine in a wind farm without having an operator configure and program a dashboard to do this.
Finally, by making use of Semantic Web technologies the dashboard can also be easily extended. New sensors, virtual sensors and visualization services can easily be added by encapsulating them in a web API and semantically annotating them enabling the dynamic dashboard to discover them.
To conclude, monitoring of an entire fleet of complex machines, such as wind turbines, can be made easier by dividing the task in three levels. Additionally, data-driven techniques can be used for monitoring tasks, enabling to detect conditions and faults which were up until now, difficult to detect with state-of-the-art techniques. Finally, monitoring of many sensors streams can be made more accessible by web-enabling sensors and virtual sensors so that they can be used in dynamic dashboards. In the end, the approaches and systems researched and proposed in this dissertation have the potential to make condition monitoring more effective and easy to use, resulting in reduced maintenance and operational costs. When the operation and maintenance costs are lowered of offshore wind turbines, the price of electrical energy generated by offshore wind can drop making investments in renewable (wind) energy even more attractive. By doing so, we continue to progress towards a greener world.

6.2 Future work

In this dissertation, one of the goals is to complement physics-based approaches with data-driven ones in order to overcome the current shortcomings. For example, during the research conducted within this dissertation, it has become clear that expert knowledge is most useful in tasks such as pre-processing, interpretation or even feature engineering. Hence, both have a vital role to play in predictive maintenance. One of the possible future research directions, to make the leap to predictive maintenance, is to enhance the synergy between the expert knowledge and the data-driven approaches. Research into better pre-processing of the data, restricting models to enable transferability and generalizability to other machine set-ups, and gaining insights into the data are relevant topics for which both expert knowledge and data-driven approaches are required.
Another research challenge in wind energy is to exploit the large amount of data that is available in a wind park. Not only to optimize the design of the turbines, but also to improve condition monitoring. Data-driven techniques are an ideal tool to tackle this challenge as has been demonstrated in this dissertation. Many more use-cases can be thought of when large amounts of data become available, such as the prediction of wind turbine behavior based on other wind turbines in the surrounding. Furthermore, when data becomes available which has been recorded for

a long time, one can try to predict the remaining useful lifetime of components or even turbines. Recently, the IDlab has begun to participate in a project which will focus on data of entire fleets of machines such as wind turbine parks. This project, called "hypermodelling strategies on multi-stream time-series data for operation optimization", has the goal to reduce the operation and maintenance costs of a fleet of machines, by using innovative modeling and dataprocessing/analysis techniques that are able to cope with large amounts of complex data. Within this project, both data-driven and physics-based approaches are considered for fleet-level anomaly detection. As is demonstrated in Chapter 4, it is possible to detect faults using complex machine learning approaches. However, an expert designed feature can detect a condition with significant less computational effort. Hence, the reason why physics-based models should also be considered even when dealing with a large amount of data.

Not only Big Data is a promising frontier in the wind energy sector, but also new types of sensors. Within this dissertation the feasibility of infrared thermal imaging is researched. Even though the potential of infrared thermal imaging for automated fault detection has been established on experimental set-ups, future work will have to include research into the practical applicability within wind turbines. Also, the costs of an infrared thermal camera is currently high. However, in 2015 and 2016, consumer-grade thermal cameras have started to appear at a fraction of the costs of common thermal cameras. Even though these low-end thermal cameras have a significantly smaller resolution and framerate, they are the first step towards making infrared thermal imaging-based automated condition monitoring a reality.

In order to process the data from new sensors and the large amount of data in general, further research into machine learning in combination with semantics will have to be done. In this dissertation, first steps are taken to combine Semantic Web technology and machine learning, but it would be interesting to continue this line of research to enable root-cause detection for the detected faults within this dissertation in order to identify the sources of the problem. By using the dynamic dashboard with its semantic web technologies and machine learning, (semi-) automated root-cause detection can be improved. To achieve this, several high-level research tasks are identified as future work. A first research task is to research and define semantic models (i.e. ontologies) using expert knowledge concerning root-causes and fault escalation. Hence, inference can be done using these models to find the root-cause. Another research task is to identify root-causes when no domain knowledge is available. In this case, machine learning algorithms can be used to help identify the root-cause. For example, Bayesian networks or PCA together with contribution plots can help identify possible data streams related to the fault. In a third research task, machine learning and semantic technology will come closer together as machine learning can identify interesting data streams related to the problem, which are subsequently presented to the operator through

the dynamic dashboard. The operator can provide feedback which implicitly will update the semantic domain knowledge model. In the end, we can foresee a dashboard that adapts its view based on the anomalies and related root-causes that have been detected again making condition monitoring easier.
The future perspectives can be combined resulting in synergies which further reduce the operation and maintenance costs, hence lowering the cost of renewable energy.

Appendix A

Preliminaries on machine learning

This appendix provides an overview of basic machine learning concepts and techniques required to understand this dissertation. This appendix can be consulted if the reader is not familiar with machine learning.

A.1 Notation

A Matrix is denoted as a capital bold letter e.g. $\mathbf{A}$. The k-th column vector is written as a lower case bold letter e.g. $\mathbf{a}_k$. The value in the matrix located in the i-th row and j-th column is written as a lower case italic letter e.g. a_{ij}. A scalar is written as an italic letter (both upper case as well as lower case).

A.2 Machine learning

Machine learning algorithms are able to learn from examples without being programmed explicitly. This is done by constructing a model which consists on one hand of parameters that have to be optimized for the task at hand, and on on the other, a function that produces a prediction given those parameters. In essence, the goal is to make the model fit the example data well, so it can make predictions about it.

Machine learning can be seen as an optimization problem. A data set, consisting of training examples, is provided and a machine learning model is trained by optimizing a certain objective function. However, as opposed to typical optimization problems, the goal of a machine learning algorithm is to perform well on new, previously unseen data.

Machine learning can be either supervised or unsupervised. Supervised machine learning will learn the relationship between the data and a corresponding output variable. It is used to classify data into groups or predict continuous values. Unsu-

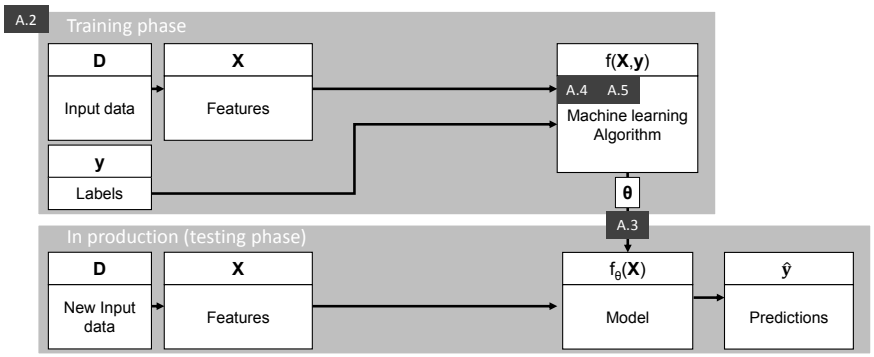


Figure A.1: General block diagram on supervised machine learning. From the input data, features are extracted, and – together with the matching labels/responses– provided to the machine learning algorithm. The machine learning algorithm will create a model using the provided data. Using this machine learning model, predictions can be made for new unseen data.

pervised machine learning on the other hand will learn a relationship to describe a possible hidden structure in the data. Unsupervised machine learning is used to estimate the density, cluster, or reduce the dimensionality of the data set. In this dissertation mainly supervised techniques are employed, hence, supervised machine learning is discussed more in depth.

In Figure A.1, a general block diagram of a supervised machine learning system can be seen where the numbers in dark boxes match the relevant sections in this chapter. When collecting data, such as sensor measurements, accompanying labels/targets/ground truth can be gathered. The goal of using machine learning is to predict such labels, given the input data. In the training phase, data (**D**) together with labels (**y**) are gathered. Using the data, features are extracted. A feature (x) is a distinctive attribute or aspect of the data. Features are organized in a design matrix ($\mathbf{X} \in \mathbb{R}^{nxd}$, n denoting the amount of samples and d the dimensionality/number of features). **X** can be seen in Equation A.1.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1d} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2d} \\ & & \dots\dots\dots\dots & & \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nd} \end{bmatrix} \tag{A.1}$$

The combination of **X** and **y** is called labeled data. An example of **y** can be seen in Equation A.2.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ y_n \end{bmatrix} \tag{A.2}$$

Labeled data is provided to a machine learning algorithm, which will learn the relation between $\mathbf{X}$ and $\mathbf{y}$ using a training procedure, resulting in a machine learning model Θ. The machine learning model can now predict labels given new data. Hence, in a production/testing phase, data is provided to the trained machine learning model which will output predictions ($\hat{\mathbf{y}}$) for the new data.

A.3 Model creation

When creating a machine learning model, different settings and parameters can be tuned in order to optimize performance. The procedures to do this, are described here.

A.3.1 Hyperparameters

As opposed to the parameters of a machine learning model, which are learned by the learning algorithm using data, hyperparameters are not tuned by the learning algorithm. Hyperparameters are parameters which for example determine the model's complexity, such as the number of trees, the maximum depth of a decision tree or the number of layers of a neural network. These are either manually determined or using a hyperparameters optimization algorithm, such as grid-search.

A.3.2 Training and testing

To make sure that a machine learning model will work well on unseen data, it has to be tested on unseen data. To ensure this, a data set is split in a training set and a test set. The model is created using the training set and is subsequently tested on the test set. If enough data is available a data set can also be split into three sets. A training set, validation set and test set. The training set is used to train the model. The validation set is used to test combinations of hyperparameters and select the most optimal combination. The test set is only used in the end when the model has been created and the hyperparameters have been optimized.

A.3.3 Cross validation

Another approach to evaluate a machine learning model is to use k-fold cross validation. Using k-fold cross validation, the data set is randomly partitioned into k equal size subsets. Of these subsets, a single subset is retained as the validation

data for testing the model, and the remaining k-1 subsets are used as training data. The cross-validation process is repeated k times such that each subset is used as validation set only once.

A.3.4 Overfitting and underfitting

The validation procedures described above are designed to test the models regarding their generalization capabilities, i.e, to test how well they predict for unseen data. When a machine learning model is overly complex, for example by having too many parameters compared to the number of training samples, the model will not capture the underlying trend of the data, but rather memorize the data resulting in poor ability to generalize. This is called overfitting. Conversely, when a model is too simple, it is not able to capture the underlying trend and will will not be able to predict well. This is called underfitting.

A.3.5 Grid-search

Grid-search can be used to find the most optimal hyperparameters for a machine learning algorithm. This is done by defining a set of plausible hyperparameters and testing every combination of these hyperparameters.

A.4 Regression

Supervised machine learning has two major subgroups, namely regression and classification. Regression is the task of predicting continuous output values given data. Below, several regression algorithms used in this dissertation are discussed.

A.4.1 K-nearest neighbors regression (KNN)

KNN is a non-parametric instance based regression algorithm [1]. Instead of learning the underlying pattern of the data, the algorithm will compare new samples to instances which are saved in the training phase. A new sample is assigned a value equal to the average of the values of its K closest neighbors determined using the euclidean distance.
More concretely, the algorithm consists of three steps:

- **Step 1. Calculate distances**: Given a new sample $\mathbf{x}_{new}$, calculate the distance to all samples in the training set using Equation A.3. N is the number of samples in the training set.

$$d(\mathbf{x}_{new}, \mathbf{x}) = \sqrt{\sum_{i=1}^{N}(x_{new,i} - x_i)^2} \tag{A.3}$$

- **Step 2. Sort the samples**: Using the distances calculated in step 1, sort the samples from small to large.
- **Step 3. Determine output value**:

  Predict the output value $\hat{y}$ using Equation A.4

$$\hat{y} = \frac{1}{K}\sum_{k=1}^{K}(y_k) \tag{A.4}$$

  Where K is a hyperparameter.

KNN allows multivariate regression, but it should be noted that when using multiple features, the features should be normalized, i.e. re-scaled so all features have the same scale. This is due to the fact that when using the Euclidean distance, features in a large range will influence the distance metric more than those with a small range.

A.4.2 Regression trees (CART)

Regression trees split the data set/feature space into several subsets/regions. A region is delimited by a certain feature and a corresponding feature value. For example, if feature one of a sample is smaller than value s, the sample belongs in region Q_1, else it belongs to region Q_2. Every sample in the data set is put into such a region (Q_m), based on the values of its features. Every region has a corresponding response c_m which is determined using Equation A.5, where ave is the average according to the notation in [1].

$$c_m = ave(y_i|\mathbf{x}_i \in Q_m) \tag{A.5}$$

To make a prediction for a new sample, a corresponding region has to be determined based on the sample's feature values. When the region is known, the corresponding response c_m is used as prediction for the new sample as is formulated in Equation A.6. In this equation, M is the total number of subspaces; $\mathbf{x}$ a new sample for which a prediction has to be made and $\mathbb{1}$ the indicator function.

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{1}(\mathbf{x} \in Q_m) \tag{A.6}$$

To construct the regions, the regression tree building algorithm has to automatically decide on the feature (f) and accompanying value (s) to split the data set/feature space (Q) on. The goal is to split Q in a manner so that when predicting for a data set, the responses match $\mathbf{y}$ as well as possible. In practice a feature f and split point s is sought that solves the mean squared error, as formulated in Equation A.7.

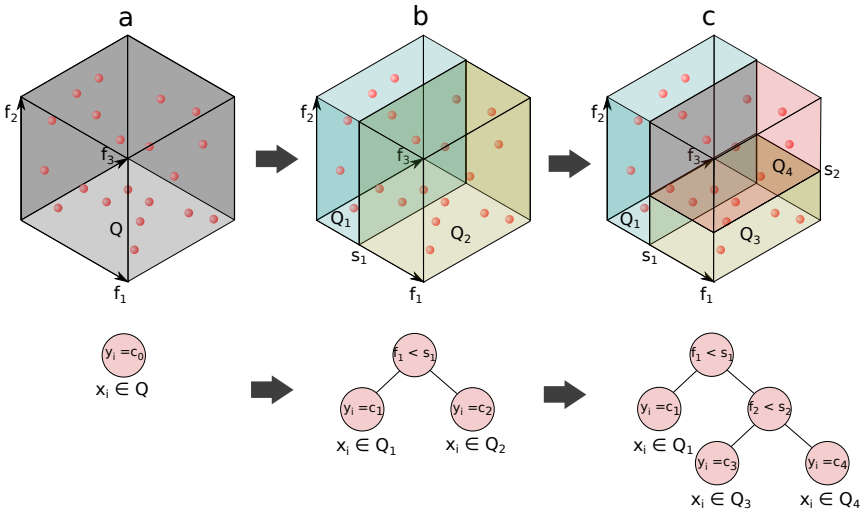


Figure A.2: At the left-hand side the data set Q is not split yet. In the middle, Q is split into Q_1 and Q_2 based on feature f_1 and value s_1. At the right-hand side, Q_2 is further split into Q_3 and Q_4 using feature f_2 and value s_2.

$$\min_{f,s}[\min_{c_1} \sum_{x_i \in Q_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in Q_2(j,s)} (y_i - c_2)^2] \tag{A.7}$$

For each possible splitting feature f, the determination of the split point s can be done very quickly by scanning through all of the unique possible options in the data. When the split is found, the data is partitioned in two subsets. This region splitting procedure is done recursively to construct the tree. Figure A.2 illustrates this procedure visually.

To determine how many splits a tree should contain, a hyperameter optimization procedure can be applied.

It should be noted that no normalization of the features is required when using tree-based approaches. This is because the range of the features do not affect the tree growing procedure.

For more information on regression trees the reader is referred to [1].

A.4.3 Random forest (RF) regression

Random forests are built upon regression trees. Regression trees consist of human interpretable "if-then" rules, built by optimally splitting the data set in subsets according to a certain feature and accompanying feature-value. In this disseration the CART algorithm [2] is used to build the regression trees. CART constructs binary trees and to find the optimal splits, the mean squared error metric is used.

After a regression tree is constructed, a new prediction based on the regression tree can be computed by traversing down the constructed tree, and subsequently taking the average value of the remaining samples in the subset.
The downside of a regression tree is that it is prone to overfit, i.e. the underlying pattern in the data will not be learned, but the data itself will be memorized, including the random noise. Such a tree will have high variance in the predictions. Variance is the variability of a model's prediction for a given data point. To reduce the high variance, ensemble methods have been proposed, such as random forests. A random forest consists of a set of regression trees grown on subsets of the original training set. These subsets are created using bootstrap aggregation (bagging). Additional to using a subset of the data, often, a subset of features is used to make sure that not all regression trees are trained using the same features.

Bootstrap aggregation or bagging: Given a training set $\mathbf{X}$ containing n samples, bootstrap aggregation generates m new training sets $\mathbf{X}_i$ each containing n' samples. These new training sets are constructed by sampling from $\mathbf{X}$ uniformly and with replacement.

In order to get a prediction from a random forest, the individual predicted values from the regression trees are averaged [1]. The goal is to have uncorrelated trees, i.e. the fluctuations in the predictions causing high variance should not be correlated to the fluctuations of other models. This because when averaging the predictions, the variance of the entire RF will be small compared to the variance of the individual models.
The most important hyperparameter to be tuned for a random forest is the amount of trees that are used. One hand, if only a few trees are used, the performance will often not be much better compared to when using a single regression tree. On the other hand, if too many trees are used, the model will become very large and difficult to work with, and possibly start to overfit.

A.4.4 Extremely randomized trees (ERT)

There are two main differences between RF and ERT. First, the construction of the regression trees in ERT is not done using different bootstrap samples, instead the entire training set is used. Second, as opposed to finding an optimal feature and feature-value to split the data set at every step, a random value is chosen for every feature. From this feature-value set an optimal feature together with its random value is used to split the data set.
Additional to the fact that ERT can outperform RF, the ERT method is also computationally cheaper than RF. This is due to the fact that for the RF method a best possible split has to be found considering every available feature and possible

value, while for the ERT method one random value is chosen for every available feature.
Similar to RFS, the number of trees is a hyperparameter for ERT, which has to be determined by the practitioner, using for example grid-search.

A.4.5 Gradient boosted regression trees

The idea behind this method is to improve upon a simple model, such as a regression tree, by creating an additional regression tree that focuses on the errors of the previous regression tree. This is done in a stage-wise procedure, meaning that additional regression trees are created, each focusing on the error of the previous regression tree. Such a procedure is called boosting.
The algorithm starts off by creating a regression tree $f(\mathbf{X})$ as described in Section A.4.2. $\hat{\mathbf{y}}$ are the predictions of this regression tree, when provided with $\mathbf{X}$. These predictions will not match $\mathbf{y}$. The differences between $\hat{\mathbf{y}}$ and $\mathbf{y}$ are called the residuals $\mathbf{r}$. Hence, $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$. If these residuals were known for new data, perfect predictions could be made ($\mathbf{y} = \hat{\mathbf{y}} + \mathbf{r}$). Hence, the very interesting idea to create a regression tree to predict these residuals, given $\mathbf{X}$. In other words, $f_2(\mathbf{X}) \approx \mathbf{r}$. This means that $\mathbf{y} \approx \hat{\mathbf{y}} + f_2(\mathbf{X})$ or rather $\hat{\mathbf{y}}_2 = f(\mathbf{X}) + f_2(\mathbf{X})$ which improves the predictions as a model has been added that can predict the error of the first model. However, improvements upon predictions can still be made by again creating a regression tree using the new residuals. This entire procedure is in fact done iteratively as is formulated in Equation A.8, where k is the amount of additional models to create, which is a hyperparameter.

$$g(\mathbf{X}) = \sum_{i=1}^{k} f_i(\mathbf{X}) \tag{A.8}$$

This procedure is called gradient boosting because the residual is in fact the negative gradient of the objective function with respect to the output of the model. By constructing a regression tree to predict the residuals, the regression tree can in fact predict an approximation of the negative gradient. Hence, the model is updated based on the negative gradient.
The gradient which is used to update the model, is scaled using a learning rate parameter to limit the influence of the gradient. Furthermore, to help the model to generalize, the regressions trees in the gradient boosted regression trees algorithm are kept shallow to prevent overfitting. Finally, at each iteration regression trees are often not trained on the entire data set, but rather a random subset of the data. This is also done to prevent overfitting.
For more information on stochastic gradient boosted regression trees the reader is referred to [1].

A.5 Classification

Contrary to regression, where the machine learning task is to predict continuous values, classification has the task to predict a discrete output given data. Next, several classification algorithms used in this dissertation are discussed.

A.5.1 Classification trees (CART)

Classification trees are built similar to regression trees. However, whereas regression trees use the mean squared error to optimally split the data on, classification trees use metrics such as the Gini impurity. To optimally split a data set, the Gini impurity has to be minimized. The Gini impurity is given by Equation A.9, where p_c is described in Equation A.10, and c is a certain class and C the total number of classes. The Gini impurity of a subset is maximal if the number of samples, in the subset, per class is the same for each class, i.e. $p_1 = p_2$. Conversely, the Gini impurity is minimal if only samples of one class is present in a subset $p_1 = 1; p_2 = 0$. To make a prediction for a new sample using a classification tree, the new sample has to be put in the right subset/region and will subsequently be assigned the value equal to the majority class in said subset/region.

$$G(Q) = \sum_{c=1}^{C} p_c(1 - p_c) \tag{A.9}$$

$$p_c = \frac{1}{N} \sum_{\mathbf{x}_i \in Q} \mathbb{1}(y_i = c) \tag{A.10}$$

For more information on classification trees the reader is referred to [1].

A.5.2 Neural networks (NN)

A NN is a model that consists of layers, where each layer contains a number of nodes/neurons. Each node computes a weighted linear combination of the input, followed by an non-linear activation function. The output of a layer is calculated as in Equation A.11, where $\mathbf{x}_{n-1}$ is the input to layer n, $\mathbf{W}_n$ a matrix of weights, $\mathbf{b}_n$ a vector of biases and f the activation function. The activation function can be for example a sigmoid or tanh. However for the final layer, the softmax activation function is used for multiclass classification problems where the classes are mutually exclusive. The softmax function provides a probability distribution of the output classes. The softmax function can be seen in Equation A.12, where $\mathbf{z}$ is the pre-activation as can be seen in Equation A.13, which has k elements, corresponding to the number of classes.

$$\mathbf{x}_n = f(\mathbf{W}_n \mathbf{x}_{n-1} + \mathbf{b}_n) \tag{A.11}$$

$$f(\mathbf{z})_j = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{A.12}$$

$$\mathbf{z}_n = \mathbf{W}_n \mathbf{x}_{n-1} + \mathbf{b}_n \tag{A.13}$$

Generally, a neural network consists of three layers, i.e.: one input layer, a hidden layer and an output layer. The number of layers and nodes per layer are hyper-parameters that have to be determined. When no hidden layer is used, the neural network is only capable of representing linearly separable functions. When one or more are used, more complex functions can be learned. How many layers are required, depends on the complexity of the task. Regarding the number of nodes, on the one hand when few nodes are used there is a chance that the neural network will not have the capacity to learn a good decision boundary. On the other hand, when too many nodes are used, the NN will memorize the data.
In order to evaluate if a NN is performing well, a loss function is required. When using the softmax activation function in the last layer of the network, the cross-entropy loss function can be used as in Equation A.14, where q represents the estimated class probabilities of the respective classes; k determined by the softmax function and p the true distribution wherein all probability mass is on the correct class (i.e. p = [0,...1,...,0] contains a single 1 at the location of the correct class). Hence, the eventual loss –also known as the softmax loss– when combining the softmax function and the true distribution can be seen in Equation A.15. In Figure A.3 an example of the softmax activation in combination with the cross-entropy loss can be seen for sample $\mathbf{x}_i$.

$$H(p, q) = -\sum_k p(k) \log q(k) \tag{A.14}$$

$$l = -\log\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right) \tag{A.15}$$

This loss function indicates how well a NN is performing. When the output of the loss function is high, the NN is not performing well. Conversely, if the loss value is low, the network is performing well. Hence, a NN that is performing optimally will have a very low loss value. To train a NN, the loss function has to be minimized by adapting the weights in every layer. Updating the weights is done using gradient descent. The update rules for gradient descent can be seen in Equation A.16, where η is a hyperparameter called the learning rate and $\nabla_{\mathbf{W}} l$ the gradient of the loss function with respect to the weights. The purpose of gradient descent is to update the weights so that the value of the loss function becomes smaller. The gradient of the loss function provides the information in what direction the weights should be updated in order to minimize the prediction error. The update rules of gradient descent are executed iteratively during many epochs (τ), each time with

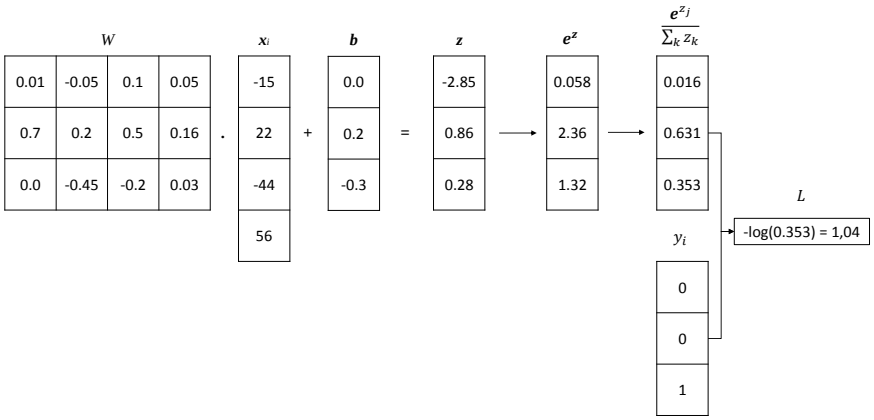


Figure A.3: Example of the softmax activation function and the cross-entropy loss, i.e. softmax loss.

the goal of lowering the loss value. To calculate the gradient of the loss ($\nabla_{\mathbf{W}} l$) the backpropagation algorithms is used [3].For more information on backpropagation, the reader is referred to [4], for an in depth explanation.

$$\mathbf{W}^{(\tau)} \leftarrow \mathbf{W}^{(\tau-1)} - \eta \nabla_{\mathbf{W}} l \tag{A.16}$$

A.5.3 Logistic regression

Logistic regression is a binary linear classifier, where a logistic sigmoid is applied to a linear function of the feature vector ($\mathbf{x}$) restricting the output between 0 and 1 (see Equation A.17). Logistic regression does not output the class of the sample, but rather provides the probability that the given sample belongs to a certain class. To optimize the model's parameters, the maximum likelihood approach is used. The maximum likelihood estimate of a parameter is the value that maximizes the probability of the observed data. Several algorithms can be used to do this, however, in practice the Newton-Raphson approach is often used [5].

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + exp(-\mathbf{w}^T \mathbf{x})} \tag{A.17}$$

A.6 Evaluation metrics

In order to judge the performance of machine learning algorithms, several metrics can be calculated.

A.6.1 Regression

Within this dissertation, two metrics to evaluate regression tasks are used.

Mean absolute error (MAE) (Equation A.18) is an error metric, which means that the lower the score the better.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{A.18}$$

Root mean squared error (RMSE) (Equation A.19) is also an error metric, hence, the lower the score the better. Additionally, compared to the MAE, RMSE amplifies large deviations resulting in larger errors.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{A.19}$$

A.6.2 Classification

Within this dissertation classification tasks are evaluated using the accuracy metric, but also using confusion matrices. The data sets used are balanced hence accuracy enables the most intuitive comparison.

The **accuracy** metric can be seen in Equation A.20 with $|TP|$ being the amount of true positive classifications; $|TN|$, the amount of true negative classifications; $|FP|$, the amount of false positive classifications, e.g. a false alarm, and $|FN|$, the amount of false negative classifications, e.g. missed faults. In essence, the goal is to maximize the true negative and true positive classifications and minimize the false positive classifications and the false negative classifications. However, often there will be a trade-off between minimizing the false positive classifications and the false negative classifications, so a decision has to be made, e.g. allow more false alarms but detect all real faults, or miss some real faults and have almost no false alarms.
It should be noted that the accuracy score is not a good metric to use when the data set is imbalanced because, if the model always predicts the label of the majority class, the accuracy will be good although the model did not learn anything. Hence, in case of imbalance data sets, other metrics should be considered such as the Cohen Kappa statistic, F1-score or the area under the receiver operator characteristic curve.

$$accuracy = \frac{|TP| + |TN|}{|TP| + |FP| + |FN| + |TN|} \tag{A.20}$$

Confusion matrices are tables displaying the classification results of a model. How a confusion matrix works can be seen in Figure A.4.

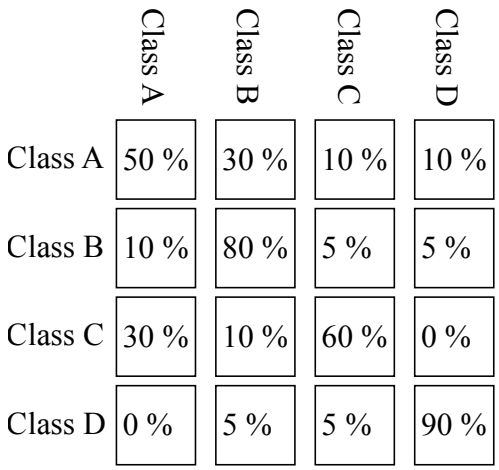
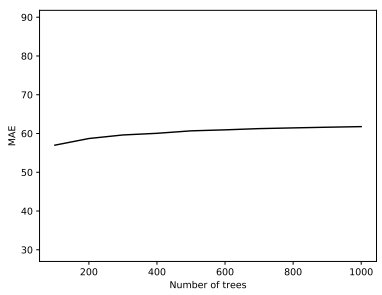


Figure A.4: Illustration of a confusion matrix. From this confusion matrix, it can, for example be seen that of all the samples belonging to class A, 50 % are classified as class A; 30 % as class B; 10 % as class C and 10 % as class D.

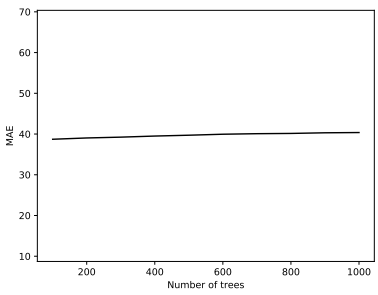
Appendix B

Hyperparameters

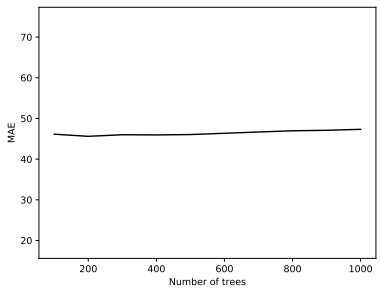
In this chapter, graphs for the hyperparameters for the multivariate machine learning models discussed in Chapter 2 can be consulted.



(a)

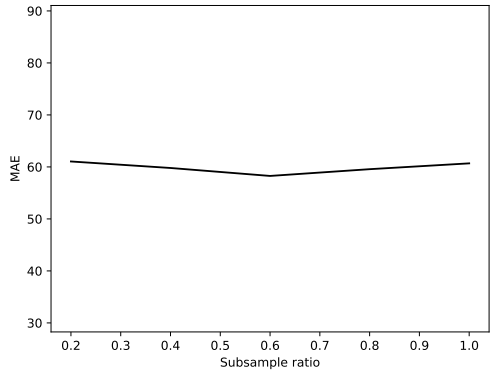


(b)

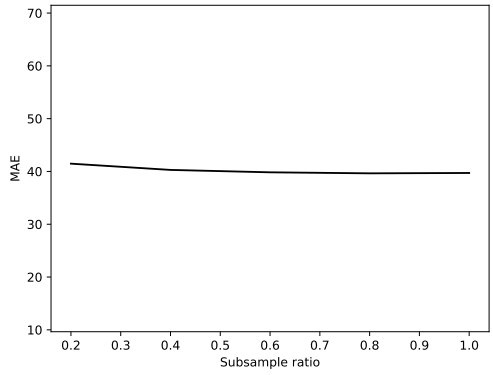


(c)

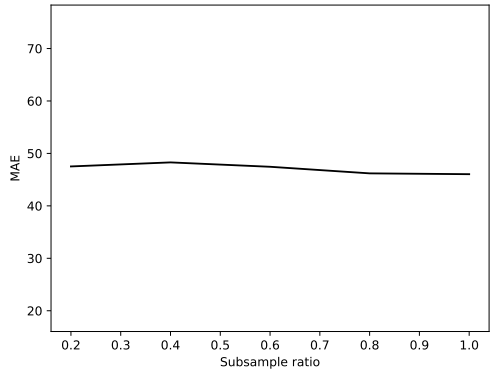
Figure B.1: Effects of modifying the number of trees for the gradient boosted regression trees model for turbine 1,2 and 3 respectively.



(a)

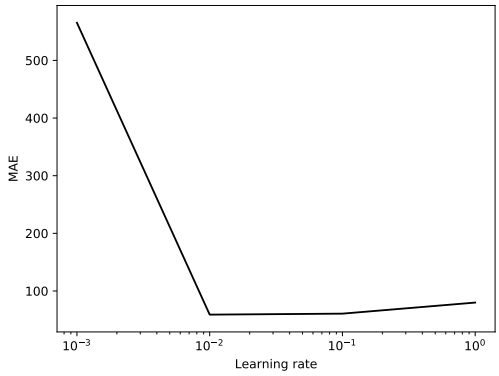


(b)

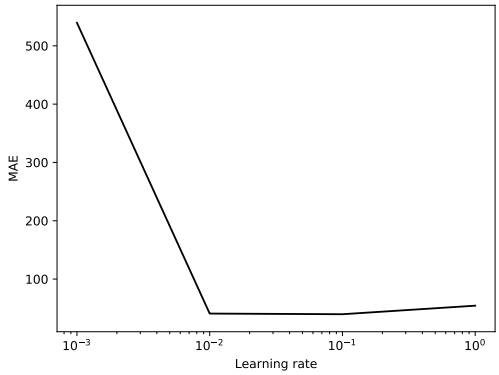


(c)

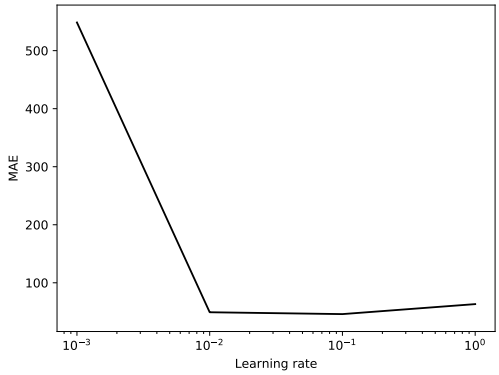
Figure B.2: Effects of modifying the subsample ratio for the gradient boosted regression trees model for turbine 1,2 and 3 respectively.



(a)

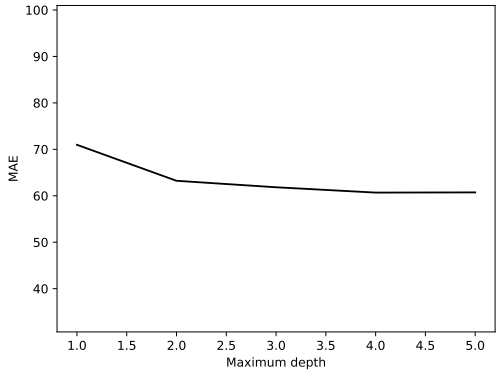


(b)

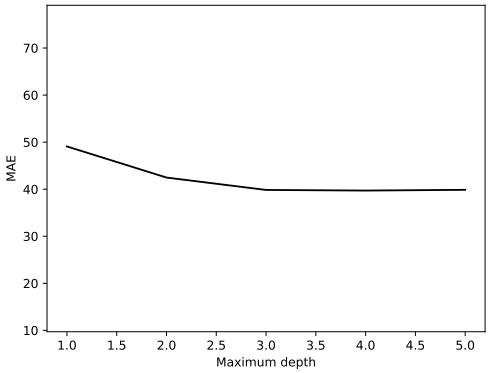


(c)

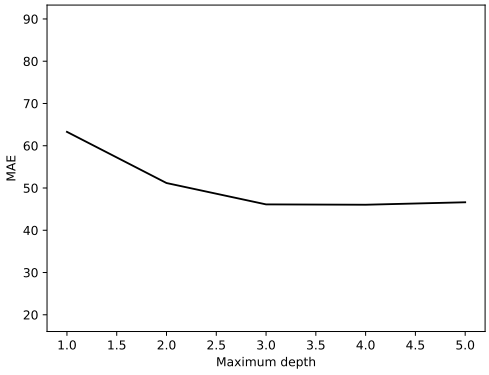
Figure B.3: Effects of modifying the learning rate for the gradient boosted regression trees model for turbine 1,2 and 3 respectively.



(a)

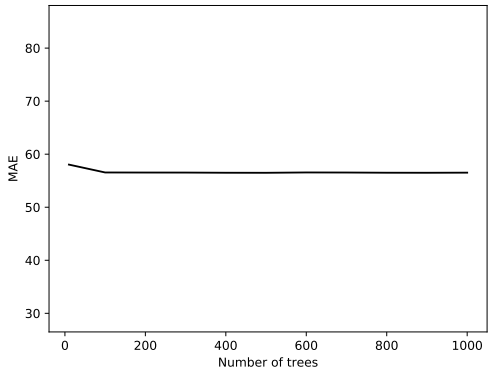


(b)

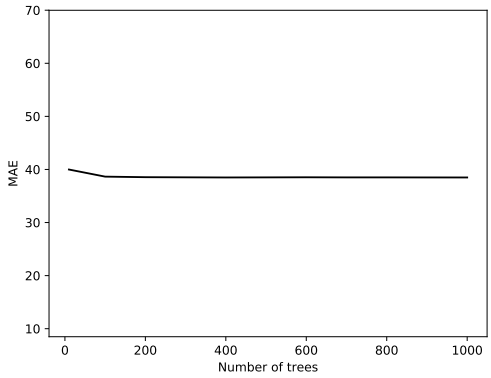


(c)

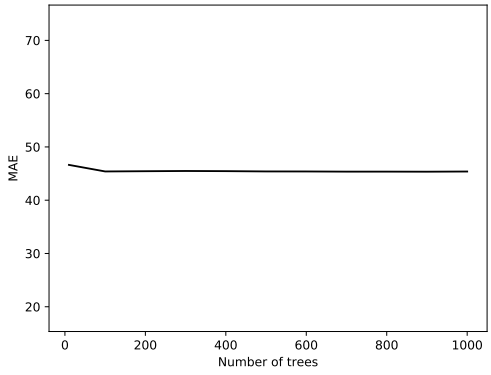
Figure B.4: Effects of modifying the maximum depth for the gradient boosted regression trees model for turbine 1,2 and 3 respectively.



(a)

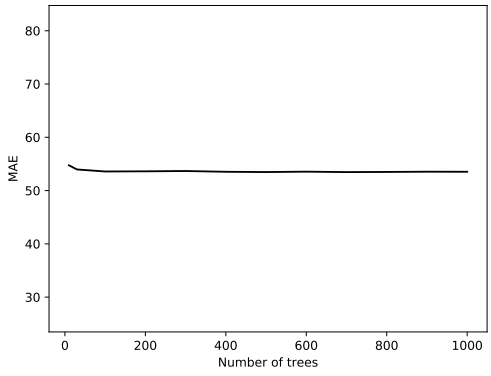


(b)

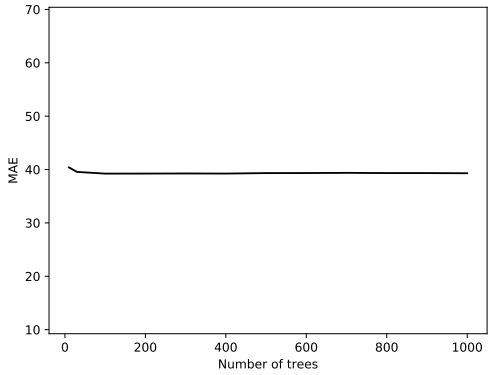


(c)

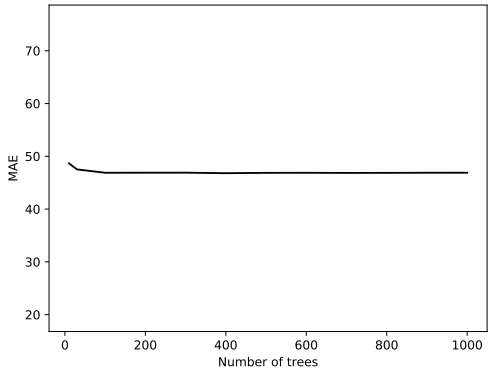
Figure B.5: Effects of modifying the number of trees for the random forest model for turbine 1,2 and 3 respectively.



(a)



(b)



(c)

Figure B.6: Effects of modifying the number of trees for the extremely randomized trees model for turbine 1,2 and 3 respectively.

Appendix C

Towards intelligent lubrication control: infrared thermal imaging for oil level prediction in bearings

C.1 Introduction

Industry consumes 33 % of the worlds available energy [117]. The majority of this energy goes to electromechanical drive systems. Hence, reducing the energy usage of these machines can have a great impact on the global energy usage. To reduce the energy usage of drive systems, the individual components have to be optimized such as rollling element bearings (REB). Bearings are one of the most common components in drive systems. These bearings suffer from energy losses as for example in cars up to 30 % of the energy can be consumed by bearings [118]. Generally, the total losses of a bearing can be divided into two parts. The main part is the mechanical friction. It results for example from contact forces between rolling elements and the runway. The second part is caused by displacement of lubricant, called churning and drag losses. These lubricant related losses are important because they depend on the operating conditions, and can be as high as the mechanical losses! [119]. Some bearing manufacturers already provide energy-efficient bearings [120]. However, the lubrication is often not optimized for energy-efficiency. The selection of bearings follows from a procedure where operation conditions such as operational speed, nominal load and working environment are taken into account. Depending on the bearing and the usage, an appropriate lubrication system is chosen. However, during the operational period of a machine the parameters of the lubrication are often not monitored actively or adjusted. Nevertheless, as the energy losses in a bearing are dependent on the lubrication, adjusting these parameters in real-time can potentially result in a significant amount of saved energy. Recently, it was determined that rotational speed,

oil viscosity and especially the oil level affect the drag and churning losses in a bearing [119]. Hence, an intelligent lubrication system should be able to to adapt these properties in real-time to optimize the energy efficiency. Nevertheless, determining the oil level, which is the most important factor, in a bearing is difficult as it is not possible to visually determine this, and no sensors exist for this purpose. We showed that it is possible to detect grease inadequacy in bearings using infrared thermal imaging (see Chapter 3). Generally, we showed that lubrication related conditions can be identified using a data-driven approach and infrared thermal imaging. Therefore, in this chapter we investigate the use of infrared thermal imaging in order to determine the oil-level in bearings using a data-driven approach, as a step towards intelligent lubrication control. To do this, both a feature engineering as in Chapter 3, and a feature learning approach as in Chapter 4 is used. In the next section the set-up and data set are discussed. Next, the description of the methodology for the oil-level detection using infrared thermal imaging is given. Subsequently, the results are discussed and finally in a conclusion is given.

C.2 Data and set-up

The used set-up can be seen in Figure C.1. The used bearing is a cylindrical roller bearing (NUP212) from FAG. A re-circulatory lubrication system was used with Mobil SHC 630 oil. The hydrostatic pad and pneumatic muscle are used to apply a controlled radial force on the outer-raceway of the bearing. The controlled variables which were changed in between test-runs are the rotational velocity of the inner ring, lubricant flow rate and lubricant temperature. As thermal camera the Flir A320 was used[1]. The distance between the set-up and the camera was 70 cm. An example of an infrared thermal image can be seen in Figure C.2. The main body of the bearing cover is made out of stainless steel. However, at the left-hand side of the cover a small plexiglass window was added to monitor the oil level. This was done because the method proposed in this chapter is a data-driven method and requires ground truth data, i.e. labeled data, to learn a model on. However, as will be made clear in Section C.3, the plexiglass cover part is removed from the thermal image before the feature extraction step.
In total 30 runs were done using this set-up. It was made sure that the temperature reached steady-state, i.e. the temperatures did not rise any further, before the recording started. Every recording contains 300 frames. To make sure that the oil level can be detected in different operational conditions, several parameters of the set-up were changed between runs. The load was set to 5000 N for all these tests. The rotation speed was changed between 200, 500, 1000 and 1500 rpm. The oil

[1] http://www.flir.com/uploadedFiles/Thermography_APAC/Products/Product_Literture/090722%20A320%20datasheet_eng.pdf

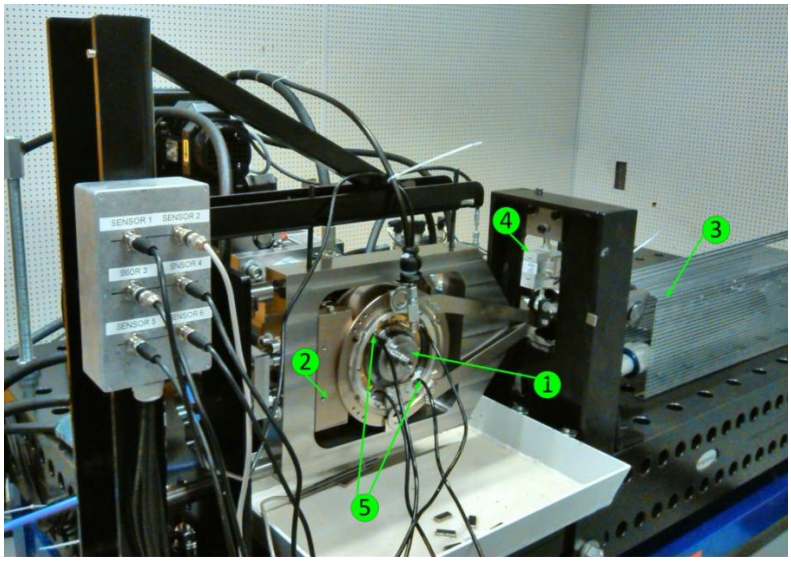


Figure C.1: Image of the used set-up. 1. bearing, 2. hydrostatic pad to apply radial load on the bearing, 3. pneumatic muscle for loading the bearing, 4. force cell for friction torque, 5. temperature measurements.

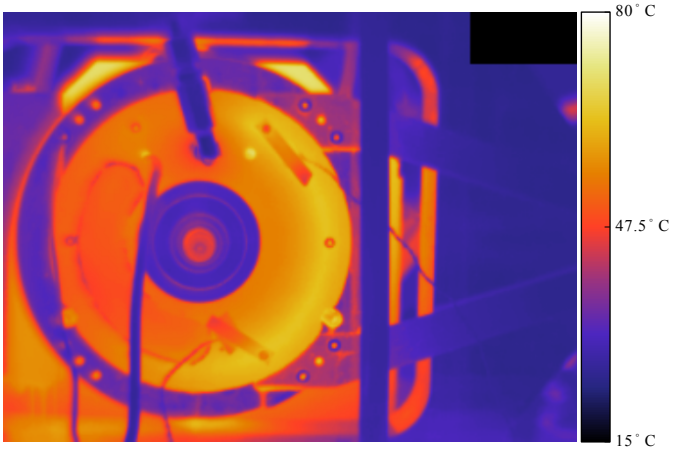


Figure C.2: Example of a thermal infrared image of the bearing.

flow rate was changed between 30, 35, 50, 100, 200, 250, 300, 350, 500, 750, and 1000 CC/min. The oil temperature was set to 40, 50 and 60 °C . Finally, the room temperature was kept constant at 23 °C.
During the test runs, the height of the oil was measured and ranged between 25 and 120 mm. The main goal of this research is to investigate if it is feasible to determine the height of the oil-level using infrared thermal imaging. As we want to investigate the feasibility, the prediction is coarse-grained. Hence, we want to be able to predict if the bearing is full of oil or not. A full bearing suffers the greatest from churning and drag losses. Hence, for each measurement a label "full" or "not full" was assigned according to Equation C.1 with a threshold level set to 100 mm. From the 30 samples 9 were labeled as full and 21 as not full.

$$y_i = \begin{cases} Full & \text{if } \textit{oil level} >= 100 \text{ mm} \\ \textit{Not full} & \text{otherwise} \end{cases} \tag{C.1}$$

C.3 Methodology: Feature engineering

Per IRT image several features have to be extracted in order to be able to determine what the oil level in the bearing is. To do this, several image processing steps are done as depicted in Figure C.3. The first step in this process is image registration.

Figure C.3: The image processing and machine learning pipeline.

C.3.1 Image registration

In between the measurements, the IRT camera moved. Due this movements, translations and rotations are induced in the data set. To enable robustness against these transformations, all the frames need to be aligned to a common reference image (i.e. image registration). By applying image registration, the REB housing will be in the same position in every frame of every recording. Image registration is done using the following steps:

1. A random reference frame from a random recording is extracted. In the end, every frame should be aligned to this reference frame.

2. From the reference frame, but also from every first frame from every recording, key points are extracted. The key points are corners and are determined using the FAST algorithm [83].

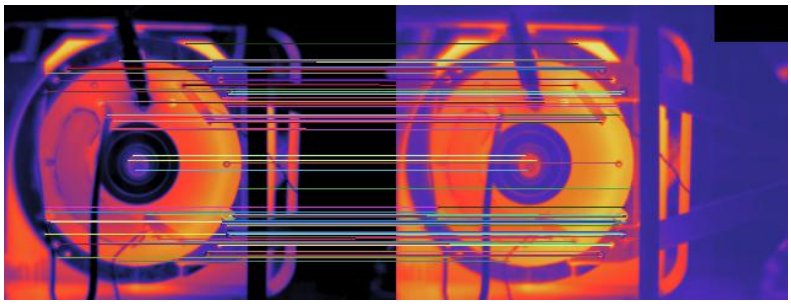


Figure C.4: Matching points between the reference frame and the frame that has to be aligned.

3. Image patches are extracted around every key point for which ORB feature descriptors are calculated [84].

4. Matching key points extracted from the reference frame and a frame that needs to be aligned are selected. This is done by calculating the hamming distance which allows for very fast matching computation compared to for example the euclidean distance [85].

5. Using the matching key points from the reference frame and a frame that needs to be aligned, the affine transformation is determined using the RANSAC algorithm. The affine transformation describes how an image should be translated, rotated and scaled so that it will match the geometry of the reference frame. This transformation is calculated for every recording and applied to every frame, resulting in recordings wherein the housing is approximately in the same place. An example of matching keypoints can be seen in Figure C.4.

C.3.2 Region of interest extraction

The goal in the ROI extraction step is to segment the right side of the bearing housing as the plexiglass part cannot be used for feature extraction as plexiglass is not used in industrial machines. The plexiglass part, in this set-up, is used to visually monitor the oil-level and create ground truth labels for the different test runs. Furthermore, only the cover should be extracted and the background should be removed as the cover is closest to the bearing. To do this ROI segmentation, several steps are required. First, the shaft has to be detected. As the shaft is a circle in the image, this is done using the circle Hough transform [121]. The result of this can be seen in Figure C.5a. Using this circle a binary matrix, i.e. a mask, is created according to Equation(C.2) where $\mathbf{M}_1$ is the binary matrix and i and j its

indices, i_{1_c} and j_{1_c} are the coordinates of the center of the circle and r_1 its radius. In this binary mask all pixels within the circle will be equal to 0 and the pixels outside the circle equal to 1. Afterwards this mask is applied on the IRT frames: $I = I \odot \mathbf{M}_1$.

$$M_{1_{ij}} = \begin{cases} 0, & \text{if } (i - i_{1_c})^2 + (j - j_{1_c})^2 < r_1^2 \\ 1, & \text{otherwise.} \end{cases} \tag{C.2}$$

Now that the center is removed, in the next step the circle Hough transform is applied again. This time to detect the outer-raceway of the bearing. The result can be seen in Figure C.5b. The goal is now to remove the background of the image. To do this, again a mask is created as in Equation C.3. This time, the pixels within the circle are equal to 1 and those outside the circle equal to 0. Afterwards, the mask is applied on the IRT frames: $I = I \odot \mathbf{M}_2$. This operation results in an isolated ROI as can be seen in Figure C.5c. From this ROI only the right-hand side is kept as this side made out of stainless steel. The final ROI can be seen in Figure C.5d.

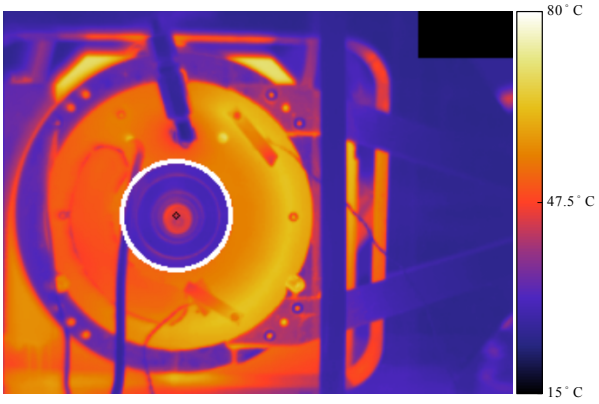
$$M_{2_{ij}} = \begin{cases} 0, & \text{if } (i - i_{2_c})^2 + (j - j_{2_c})^2 > r_2^2 \\ 1, & \text{otherwise.} \end{cases} \tag{C.3}$$

C.3.3 Feature extraction

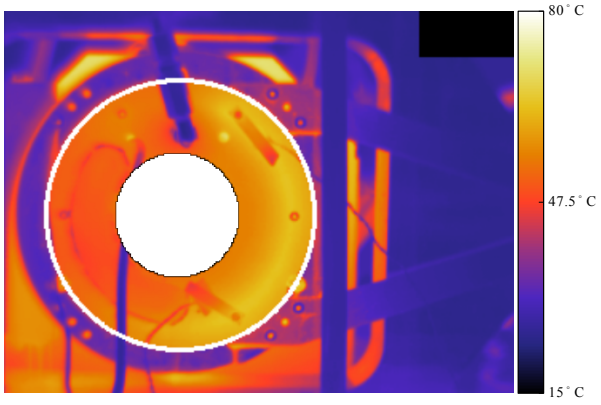
To evaluate the feature proposed in Chapter 3 the Gini Coefficient, standard deviation and the M_{20} are extracted and tested.
To minimize the influence of noise and outliers in the feature extraction step, windowing is applied. A window contains 30 frames of IR video and overlaps its neighbouring windows by 15 frames, resulting in 19 windows per video sequence. Per window all the features are averaged. This means that in the end there are 30 videos $\times$ 19 windows = 570 samples.

C.3.4 Machine learning

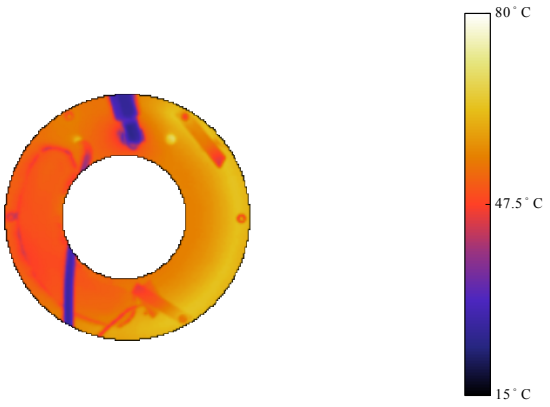
In an intelligent lubrication control system the oil level will have to be updated automatically to optimize the energy efficiency and the lifetime of the machine. This also means that the level of the oil will have to be determined automatically. Hence, an algorithm has to interpret the extracted features from the infrared thermal video and make a decision. The classification problem at hand is a binary classification problem and when plotting the data, it can be seen that a non-linear classifier is required as no straight decision boundary can separate the two classes, i.e. full and not full. As in Chapter 3, a random forest classifier is used.



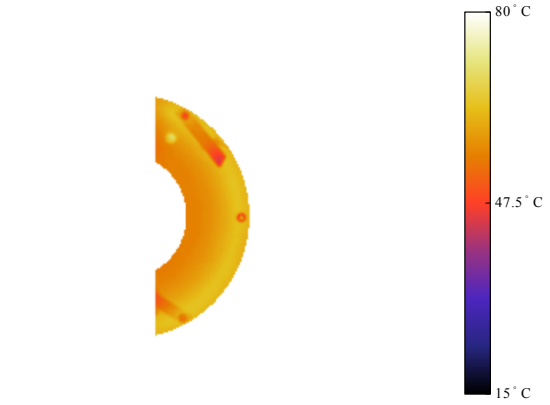
(a)



(b)



(c)



(d)

Figure C.5: Region of interest extraction procedure. First, the shaft is removed from the image (a). Afterwards the outer-raceway is detected (b). Finally, the background is removed (c) and the left part of the image is removed (d).

C.4 Methodology: feature learning

Additional to the feature engineering approach, the feature learning approach of Chapter 4 is also tested in this use case. The same preprocessing steps as described above are used. However, also rescaling and normalization as in Chapter 4 are done. As in Chapter 4, transfer learning is applied by re-using a VGG network and removing the last layer. The whole network is trained using infrared thermal images.

C.5 Results

C.5.1 Evaluation

To quantify the classification performance, four error measurements are used: accuracy, weighted precision, weighted recall and f1-score for which the equations can be seen in equations C.4–C.7, with $|\text{TP}_k|$ being the amount of true positive classifications for class k; $|\text{TN}_k|$ the amount of true negative classifications; $|\text{FP}_k|$ the amount of false positive classifications for class k, e.g. a false alarm, and $|\text{FN}_k|$ the amount of false negative classifications for class k, e.g. miss. Accuracy is chosen as it easy to interpret, however it does not always reflect the classifier's performance when the data set is imbalanced, i.e. when there are many more samples of one class in the data set than the others. As here the data set is imbalanced, weighted precision and recall are chosen to deal with this problem. For example, if the classifier doesn't learn anything and predicts that every sample in our data set has the label of "not-full", then the accuracy would be 70 %, which a fairly good accuracy. However, the classifier didn't learn anything, so the accuracy does not reflect the situation well. Weighted precision, recall and f1-score on the other hand would respectively be 49 %; 70 % and 57.66 % which better reflect the performance of the classifier and help to interpret the classifier's performance.

$$accuracy = \frac{|TP| + |TN|}{|TP| + |FP| + |FN| + |TN|} \tag{C.4}$$

$$precision = \sum_{k \in K} w_k \frac{|TP_k|}{|TP_k| + |FP_k|} \tag{C.5}$$

$$recall = \sum_{k \in K} w_k \frac{|TP_k|}{|TP_k| + |FN_k|} \tag{C.6}$$

$$f1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{C.7}$$

To make sure that the classification algorithm generalizes, the data set is split into the training and test set according to per-sample-cross validation. This means that

Algorithm and parameters	Precision	Recall	F1-score	Accuracy
RDF (amount of decision trees = 100)	86.67 %	88.13 %	86.67 %	87 %
CNN	86.67 %	88.13 %	86.67 %	87 %

Table C.1: Results of the three classification algorithms.

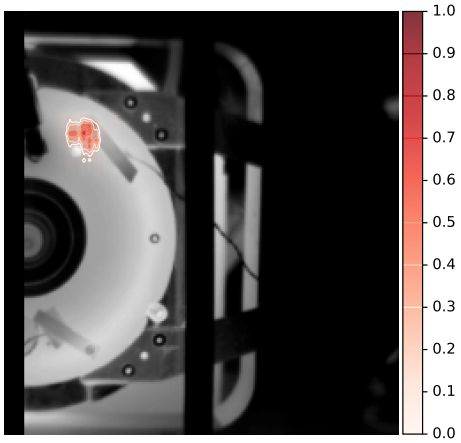
the features extracted from one recording are put in the test set and the rest is put into the training set. This is done several times so that all the samples are in the test set only once, while the remaining sequences served as training set. The results are discussed next.

C.5.2 Performance results

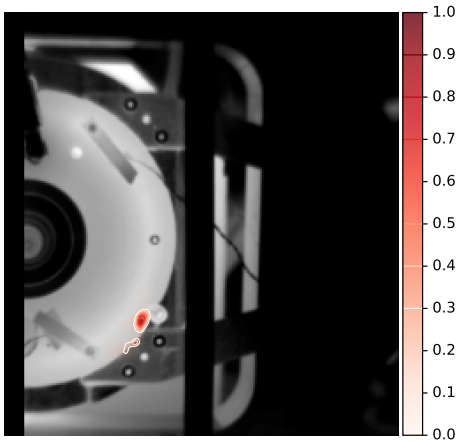
The results for the feature engineering approach and the feature learning approach are presented in Table C.1. As can be seen, the two methods perform the same. The approaches do not achieve a perfect score which is due to the fact that when the machine is rotating at 1500 rpm, the oil becomes turbulent and no exact oil level remains. Hence, miss-classification happens.

C.5.3 Feature-learning insights

As in Chapter 4, insights into the decision of the CNNs can be extracted which can be seen in Figure C.6a for a bearing full of oil and in Figure C.6b, for a bearing not full of oil. These figures illustrate that specific areas in the images are important for the specific conditions. If the bearing is full of oil, the top of the bearing is very important for the CNN's decision, which is to be expected as this part will be much warmer than when the bearing is not full. Conversely, when the bearing is not full of oil, the bottom part of the bearing is important. This is also to be expected as this part of the image will be less warm compared to when the bearing is full of oil.
Tests were done using features inspired by the regions indicated by the CNN. Two features were created. The first is the difference between the temperature at the top and the bottom of the bearing housing and the second is the difference between the maximum and minimum temperature in the IRT image. When a data set is provided containing these features to a random forest classifier and accuracy, precision, recall and f1-score of 80 % is achieved. This indicates that the information and insights extracted from the CNN are valuable to gain knowledge into the underlying physical phenomena.



(a)



(b)

Figure C.6: Regions that influence the CNNs output for a bearing full of oil (a) and not full of oil (b).

C.6 Conclusion

In this appendix a first step towards intelligent lubrication control with energy efficiency as focus is described. The goal of this research is to automatically deter-

mine the oil level in a rolling element bearing as the oil level influences the energy efficiency of a bearing the most. To detect this oil level a data-driven approach is used employing infrared thermal imaging. Both feature engineering and feature learning are used for this purpose. A data set was created containing infrared thermal videos of a rotating set-up during different conditions. The conditions were created by varying several parameters of the set-up between test runs such as rotation speed, oil temperature and flow rate. The feature engineering based approach and the feature learning based approach perform the same. Hence, it can be concluded that even under different rotation speeds, initial oil temperatures and flow rates the oil level can be determined. However, at high rotational speeds this becomes difficult due to turbulent oil. Finally it is also shown that the insights into the convolutional neural networks can help to design features which are more closely linked to the underlying physical phenomena.

Appendix D

Imbalance displacement

As a result of imbalance, the set-up will vibrate. This vibration has a certain displacement that needs to be measurable by the camera. Hence, in this appendix this displacement is calculated for the three imbalance conditions in data set two. Next, the area size that is captured in each pixel is calculated. Finally, the displacement is converted into pixels.

The following parameters are known of the set-up.

- Mass of the shaft: 1.618 kg
- Mass of one disks: 2.317 kg
- Total mass (m) = 6.252 kg
- Rotation frequency (ω): 25 Hz
- First eigenfrequency of the set-up (ω_n): 17.8 hz. This was determined using an impact test.
- Radius at which the screws were attached to the disk (r) = 5.4 x 10^{-2} m
- Mass of screw 1: 4.1 x 10^{-3} kg
- Mass of screw 2: 9.3 x 10^{-3} kg
- Mass of screw 3: 13.0 x 10^{-3} kg

The displacements are calculated according to Equation D.1. The right-hand side of the denominator can be ignored due to the fact that there is no, or no significant, dampening, hence ζ will be zero or very close to zero.

$$q_0 = \frac{\frac{m_r r}{m}\frac{\omega^2}{\omega_n^2}}{\sqrt{(1-\frac{\omega^2}{\omega_n^2})^2 + 4\zeta^2\frac{\omega^2}{\omega_n^2}}} \tag{D.1}$$

The resulting displacements are:

- Displacement by screw 1: 73.15 µm
- Displacement by screw 2: 165.9 µm
- Displacement by screw 2: 231.9 µm

Next, the area size that is captured by a single pixel, given the properties of the camera, is determined.

Two properties are required:

- Vertical field-of-view (v): 19°
- Distance from the camera to the bearing housing (d): 380 mm

Using Equation D.2, it is determined that at a distance of 380 mm, the camera can capture 130.84 mm vertically. By dividing this vertical dimension by the amount of vertical pixels ($\frac{130.84 \text{ mm}}{480 \text{ pixels}}$) it is determined that 1 pixel = 272.59 µm.

$$\text{Vertical dimension} = d\tan(v) \tag{D.2}$$

Finally, the displacements are converted into pixels (Equations D.3, D.4, D.5). Note that displacements are multiplied by two as displacements are both in the upwards and downwards direction.

$$s_1 = 2\,\frac{73.15 \text{ µm}}{272.59\ \frac{\text{µm}}{\text{pixel}}} = 0.54 \text{ pixels} \tag{D.3}$$

$$s_2 = 2\,\frac{165.9 \text{ µm}}{272.59\ \frac{\text{µm}}{\text{pixel}}} = 1.22 \text{ pixels} \tag{D.4}$$

$$s_3 = 2\,\frac{231.9 \text{ µm}}{272.59\ \frac{\text{µm}}{\text{pixel}}} = 1.70 \text{ pixels} \tag{D.5}$$

In general, there is a relationship between the weight causing the imbalance and the displacement in the image [1]. This also means that there is a limit to the amount of imbalance that can be distinguished using this technique, as amounts of imbalance that are almost the same will not be distinguishable from one another. Furthermore,

[1] Note that several properties of the set-up and camera have to be taken into account.

due to noise, inherent to a thermal camera, very low levels of imbalance will not be measurable with great certainty.
It should also be noted that due to this relationship it should also be possible to calculate the weight of imbalance to a certain degree, given the measured displacement. However, this requires further investigation.

Appendix E

M20 properties

Theorem 1. *The moment of light will be large (i.e. closer to zero) when there are two distributions of significant size, and low if there is only one distribution of significant size and one or more distributions of non-significant size. Significant size means that it is taken into account in the nominator of Equation 3.8 due to the fact that it contains points that are a part of the 20 % of the highest points.*

Proof. Given two Laplace distributions [1] (Equations (E.1) and (E.2)), and a distance function (equation (E.3)). The distance function is dependent on the location of the highest peak. λ and θ represent the scale of the distributions and μ and η the means.

$$f(x) = \begin{cases} \frac{\lambda}{2} e^{-\lambda(x-\mu)}, & \text{if } x > \mu \\ \frac{\lambda}{2} e^{-\lambda(\mu-x)}, & x < \mu \end{cases} \tag{E.1}$$

$$f(x) = \begin{cases} \frac{\theta}{2} e^{-\theta(x-\eta)}, & \text{if } x > \eta \\ \frac{\theta}{2} e^{-\theta(\eta-x)}, & x < \eta \end{cases} \tag{E.2}$$

$$g(x) = (x-z)^2 \text{ where } z = \begin{cases} \mu, & \text{if } \theta < \lambda \\ \eta, & \lambda < \theta \end{cases} \tag{E.3}$$

The $M20$ can be written as in equation (E.4), wherein $\mu - d$,$\mu + d$,$\eta - t$ and $\eta + t$ are used to define the borders in the distributions so that 20 % of the highest points are used to calculate the nominator. These boundaries are illustrated in Figure E.1.

$$M20 = f(x, \lambda, \theta, \eta, \mu, d, t) = \frac{\int_{\mu-d}^{\mu+d} \frac{\lambda}{2} e^{-\lambda|x-\mu|}(x-\mu)^2 dx + \int_{\eta-t}^{\eta+t} \frac{\theta}{2} e^{-\theta|x-\eta|}(x-\eta)^2 dx}{\int_{-\infty}^{\infty} \frac{\lambda}{2} e^{-\lambda|x-\mu|}(x-\mu)^2 dx + \int_{-\infty}^{\infty} \frac{\theta}{2} e^{-\theta|x-\eta|}(x-\eta)^2 dx} \tag{E.4}$$

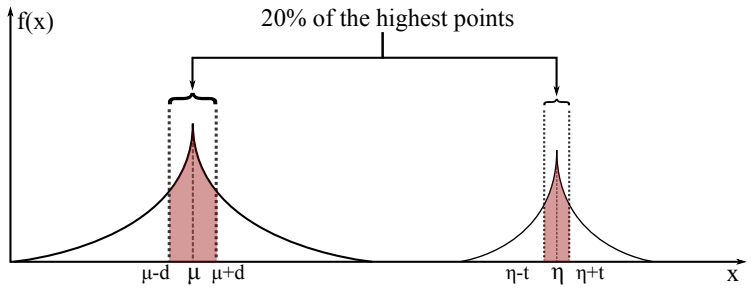


Figure E.1: Laplace distributions with boundaries.

Both μ and λ are kept fixed as varying one distribution is sufficient to illustrate the behavior of the M_{20} when there are two distributions.
The borders, which dependent d and t, have to be determined, which is done using the optimization problem in Equation (E.5).

$$\max_{d,t} 0.2\left(\int_{\mu-d}^{\mu+d} \frac{\lambda}{2} e^{-\lambda|x-\mu|}(x-\mu)^2 dx + \int_{\eta-t}^{\eta+t} \frac{\theta}{2} e^{-\theta|x-\eta|}(x-\eta)^2 dx\right) = \int_{-\infty}^{\infty} \frac{\lambda}{2} e^{-\lambda|x-\mu|}(x-\mu)^2 dx + \int_{-\infty}^{\infty} \frac{\theta}{2} e^{-\theta|x-\eta|}(x-\eta)^2 dx \quad \text{(E.5)}$$

θ is varied between 0.01 and 0.25 and η between 1000 and 1500 and used in Equation E.4 together with the results of the optimization problem in Equation (E.5), so that the $M20$, now a function of x, θ and eta, can be plotted.
Figure E.2, illustrates that when increasing the height of the second distribution the $M20$ will rise. In fact, the $M20$ will suddenly rise when the second distribution achieves a critical mass, i.e. will be used in the nominator of the $M20$.
figurename E.3, illustrates that the location of the second hotspot does not affect the $M20$ a lot. When the second distribution does not contain a critical mass, the $M20$ will decrease the larger the distance between the two distributions. This is due to the fact that the denominator of $M20$ will increase faster than the nominator when increasing the distance between the distributions.
Only when the second distribution achieves critical mass will the distance not influence the $M20$ as can be observed in figurename E.3.

□

[1]Laplace distribution is chosen as it is parameterized by its shape and location

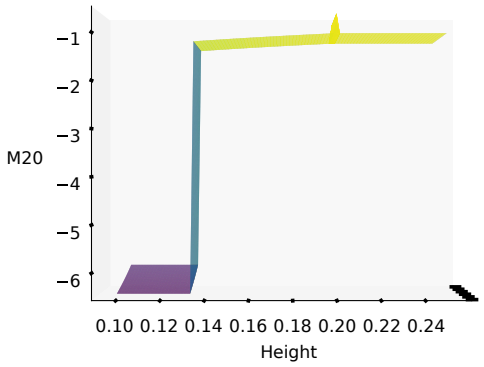


Figure E.2: y-axis: $M20$; x-axis: θ.

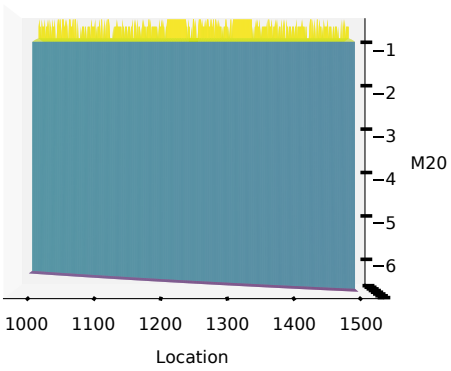


Figure E.3: y-axis: $M20$; x-axis: η.

References

[1] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[3] Y. LeCun, Y. Bengio, and G. Hinton. *Deep learning*. Nature, 521(7553):436–444, 2015.

[4] Michael A Nielsen. *Neural networks and deep learning*. URL: http://neuralnetworksanddeeplearning. com/.(visited: 01.11. 2014), 2015.

[5] Christopher M Bishop. *Pattern recognition*. Machine Learning, 128:1–58, 2006.

[6] EWEA. *Wind energy scenarios for 2020; A report by the European Wind Energy Association*, 2014.

[7] iea wind. *2015 annual report*. `https://www.ieawind.org/annual_reports_PDF/2015/2015%20IEA%20Wind%20AR_small.pdf`, 2016.

[8] World-Energy-Council. *World Energy Resources: Wind*. `https://www.worldenergy.org/wp-content/uploads/2013/10/WER_2013_10_Wind.pdf`, 2013.

[9] EWEA. *The European offshore wind industry - key trends and statistics 2015*. `http://www.ewea.org/fileadmin/files/library/publications/statistics/EWEA-European-Offshore-Statistics-2015.pdf`, 2016.

[10] International Renewable Energy Agency. *Renewable energy technologies: cost analysis series*. `https://www.irena.org/documentdownloads/publications/re_technologies_cost_analysis-wind_power.pdf`, 2012.

[11] European Commision. *Subsidies and costs of EU energy*. https://ec.europa.eu/energy/sites/ener/files/documents/ECOFYS%202014%20Subsidies%20and%20costs%20of%20EU%20energy_11_Nov.pdf, 2014.

[12] National Renewable Energy Laboratory. *Report on Wind Turbine Subsystem Reliability - A Suvery of Various Databases*. http://www.nrel.gov/docs/fy13osti/59111.pdf, 2013.

[13] C. Radu. *The Most Common Causes of Bearing Failure and the Importance of Bearing Lubrication*. RKB Technical Review-February, 2010.

[14] W. A. Smith and R. B. Randall. *Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study*. Mechanical Systems and Signal Processing, 64 – 65:100 – 131, December 2015.

[15] W. M Hannon. *Rolling-Element Bearing Heat TransferPart I: Analytic Model*. Journal of Tribology, 137(3):031102, 2015.

[16] O. Janssens, R. Schulz, V. Slavkovikj, K. Stockman, M. Loccufier, R. Van de Walle, and S. Van Hoecke. *Thermal Image Based Fault Diagnosis for Rotating Machinery*. Infrared Physics & Technology, 73:78 – 87, 2015.

[17] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, Van de Walle R., and S. Van Hoecke. *Convolutional Neural Network Based Fault Detection for Rotating Machinery*. Journal of Sound and Vibration, 377:331 – 345, 2016.

[18] G. M. Masters. *Wind Power Systems*, pages 307–383. John Wiley & Sons, Inc., 2005.

[19] Z. Hameed, Y.S. Hong, Y.M. Cho, S.H. Ahn, and C.K. Song. *Condition monitoring and fault detection of wind turbines and related algorithms: A review*. Renewable and Sustainable Energy Reviews, 13(1):1–39, 2009.

[20] J. Tamura. *Calculation Method of Losses and Efficiency of Wind Generators*, pages 25–51. Springer London, London, 2012.

[21] JG Slootweg, SWH De Haan, H Polinder, and WL Kling. *General model for representing variable speed wind turbines in power system dynamics simulations*. IEEE Transactions on power systems, 18(1):144–151, 2003.

[22] M. Singh and S. Santoso. *Dynamic models for wind turbines and wind power plants*. National Renewable Energy Laboratory, 2011.

[23] R. J. Barthelmie, S. T. Frandsen, M. N. Nielsen, S. C. Pryor, P.-E. Rethore, and H. E. Jrgensen. *Modelling and measurements of power losses and turbulence intensity in wind turbine wakes at Middelgrunden offshore wind farm*. Wind Energy, 10(6):517–528, 2007.

[24] Noppe N. , Wout W., and C. Devriendt. *Reliable empirical analysis of effects of turbulent air in an operating wind farm based on unreliable SCADA-data*. In European Academy of Wind Energy, 2015.

[25] C. Biyun, W. Suifeng, Z. Yongjun, and H. Ping. *Wind power prediction model considering smoothing effects*. In 2013 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), pages 1–4, Dec 2013.

[26] N. Charhouni, A. Arbaoui, and M. Sallaou. *Analysis of wake impact on wind farm performance using two analytical models*. In 2014 International Renewable and Sustainable Energy Conference (IRSEC), pages 323–327. IEEE, 2014.

[27] B. Sanderse, S. Pijl, and B. Koren. *Review of computational fluid dynamics for wind turbine wake aerodynamics*. Wind Energy, 14(7):799–819, 2011.

[28] P. M. O. Gebraad, F. W. Teeuwisse, J. W. van Wingerden, P. A. Fleming, S. D. Ruben, J. R. Marden, and L. Y. Pao. *Wind plant power optimization through yaw control using a parametric model for wake effectsa CFD simulation study*. Wind Energy, 19(1):95–114, 2016.

[29] G. Crasto, F. Castellani, A. R. Gravdahl, and E. Piccioni. *Offshore wind power prediction through CFD simulation and the actuator disc model*. EWEA ANNUAL EVENT, 2011.

[30] M. Etemaddar, M. O. L. Hansen, and T. Moan. *Wind turbine aerodynamic response under atmospheric icing conditions*. Wind Energy, 17(2):241–265, 2014.

[31] K. Park, T. Asim, and R. Mishra. *Computational fluid dynamics based fault simulations of a vertical axis wind turbines*. In Journal of Physics: Conference Series, volume 364, page 012138. IOP Publishing, 2012.

[32] A. Clifton and R. Wagner. *Accounting for the effect of turbulence on wind turbine power curves*. In Journal of Physics: Conference Series, volume 524, pages 1–11. IOP Publishing, 2014.

[33] J. S Oakland. *Statistical process control*. Routledge, 2007.

[34] S. J. Qin. *Survey on data-driven industrial process monitoring and diagnosis*. Annual Reviews in Control, 36(2):220–234, 2012.

[35] S. Yin, S. X. Ding, X. Xie, and H. Luo. *A Review on Basic Data-Driven Approaches for Industrial Process Monitoring*. IEEE Transactions on Industrial Electronics, 61(11):6418–6428, 2014.

[36] A. Kusiak, H. Zheng, and Z. Song. *On-line monitoring of power curves*. Renewable Energy, 34(6):1487–1493, 2009.

[37] M. Lydia, S. Suresh Kumar, a. Immanuel Selvakumar, and G. Edwin Prem Kumar. *A comprehensive review on wind turbine power curve modeling techniques*. Renewable and Sustainable Energy Reviews, 30:452–460, 2014.

[38] M. Lydia, S. Selvakumar, A.and Kumar, and G. Kumar. *Advanced Algorithms for Wind Turbine Power Curve Modeling*. IEEE Transactions on Sustainable Energy, 4(3):827–835, 2013.

[39] A. Kusiak and A. Verma. *Monitoring Wind Farms With Performance Curves*. IEEE Transactions on Sustainable Energy, 4(1):192–199, January 2013.

[40] M. Schlechtingen, I. F. Santos, and S. Achiche. *Using Data-Mining Approaches for Wind Turbine Power Curve Monitoring: A Comparative Study*. IEEE Transactions on Sustainable Energy, 4(3):671–679, July 2013.

[41] M. Schlechtingen, I. F. Santos, and S. Achiche. *Wind turbine condition monitoring based on SCADA data using normal behavior models. Part 1: System description*. Applied Soft Computing, 13(1):259–270, 2013.

[42] B. Choi. *Statistical Analysis, Modeling, and Algorithms for Pharmaceutical and Cancer Systems*. PhD thesis, University of South Florida, 2014.

[43] International Electrotechnical Commission. *Wind Turbines Part 12-2: Power performance of electricity producing wind turbines based on nacelle anemometry*. Technical report, International Electrotechnical Commission, 2008.

[44] L. Breiman. *Bagging predictors*, 1996.

[45] P. Geurts, D. Ernst, and L. Wehenkel. *Extremely randomized trees*. Machine learning, 63(1):3–42, 2006.

[46] J. Friedman. *Stochastic gradient boosting*. Computational Statistics & Data Analysis, 38(4):367–378, 2002.

[47] M. D. G. Dukes and J. P. Palutikof. *Estimation of Extreme Wind Speeds with Very Long Return Periods*. Journal of Applied Meteorology, 34(9):1950–1961, 1995.

[48] J. Krause, A. Perer, and K. Ng. *Interacting with Predictions: Visual Inspection of Black-box Machine Learning Models*. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pages 5686–5697. ACM, 2016.

[49] K. Fischer, F. Besnard, and L. Bertling. *Reliability-Centered Maintenance for Wind Turbines Based on Statistical Analysis and Practical Experience*. Energy Conversion, IEEE Transactions on, 27(1):184 – 195, 2012.

[50] J. Lacey. *An Overview of Bearing Vibration Analysis*. Maintenance & Asset Management, 23(6):32–42, 2008.

[51] B. Graney and K. Starry. *Rolling element bearing analysis*. Materials Evaluation, 70(1):78 – 85, 2012.

[52] R.B.W. Heng and M.J.M. Nor. *Statistical analysis of sound and vibration signals for monitoring rolling element bearing condition*. Applied Acoustics, 53(1–3):211 – 226, 1998.

[53] I. El-Thalji and E. Jantunen. *A summary of fault modelling and predictive health monitoring of rolling element bearings*. Mechanical Systems and Signal Processing, 6061:252 – 272, 2015.

[54] H. Ohta, Y. Nakajima, S. Kato, and H. Tajimi. *Vibration and Acoustic Emission Measurements Evaluating the Separation of the Balls and Raceways With Lubricating Film in a Linear Bearing Under Grease Lubrication*. Journal of Tribology, 135(4), 2013.

[55] P. Bokoski, J. Petrovi, B. Musizza, and Jurii". *Detection of lubrication starved bearings in electrical motors by means of vibration analysis*. Tribology International, 43(9):1683 – 1692, 2010.

[56] D. Koulocheris, A. Stathis, T. Costopoulos, and G. Gyparakis. *Comparative Study of the Impact of Corundum Particle Contaminants Size on Wear and Fatigue Life of Grease Lubricated Ball Bearings*. Modern Mechanical Engineering, 2013.

[57] P. Tchakoua, R. Wamkeue, M. Ouhrouche, F. Slaoui-Hasnaoui, T. Tameghe, and G. Ekemb. *Wind turbine condition monitoring: State-of-the-art review, new trends, and future challenges*. Energies, 7(4):2595–2630, 2014.

[58] S. Bagavathiappan, B. B. Lahiri, T. Saravanan, J. Philip, and T. Jayakumar. *Infrared thermography for condition monitoring – A review*. Infrared Physics & Technology, 60(0):35 – 55, 2013.

[59] D. Kateris, D. Moshou, X. Pantazi, I. Gravalos, N. Sawalhi, and S. Loutridis. *A machine learning approach for the condition monitoring of rotating machinery*. Journal of Mechanical Science and Technology, 28(1):61 – 71, 2014.

[60] B. A. Jaouher, F. Nader, S. Lotfi, C. Brigitte, and F. Farhat. *Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals*. Applied Acoustics, 89:16 – 27, 2015.

[61] P.K. Kankar, S. C. Sharma, and S. P. Harsha. *Fault diagnosis of ball bearings using machine learning methods*. Expert Systems with Applications, 38(3):1876 – 1886, 2011.

[62] M. Monte, F. Verbelen, and B. Vervisch. *Detection of Coupling Misalignment by Extended Orbits*. In Experimental Techniques, Rotating Machinery, and Acoustics, Volume 8, pages 243 – 250. Springer, 2015.

[63] A. Widodo, D. Satrijo, T. Prahasto, G. Lim, and B. Choi. *Confirmation of Thermal Images and Vibration Signals for Intelligent Machine Fault Diagnostics*. International Journal of Rotating Machinery, pages 1–10, 2012.

[64] H. Fandino-Toro, O. Cardona-Morales, J. Garcia Alvarez, and G. Castellanos-Dominguez. *Bearing Fault Identification using Watershed-Based Thresholding Method*. In Giorgio Dalpiaz, Riccardo Rubini, Gianluca D'Elia, Marco Cocconcelli, Fakher Chaari, Radoslaw Zimroz, Walter Bartelmus, and Mohamed Haddar, editors, Advances in Condition Monitoring of Machinery in Non-Stationary Operations, Lecture Notes in Mechanical Engineering, pages 137–147, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[65] G. Lim, Y. Ali, and B. Yang. *The Fault Diagnosis and Monitoring of Rotating Machines by Thermography*. In J. Mathew, L. Ma, A. Tan, M. Weijnen, and J. Lee, editors, Engineering Asset Management and Infrastructure Sustainability, pages 557–565. Springer London, 2012.

[66] V. Tran, B. Yang, F. Gu, and A. Ball. *Thermal image enhancement using bi-dimensional empirical mode decomposition in combination with relevance vector machine for rotating machinery fault diagnosis*. Mechanical Systems and Signal Processing, 38(2):601–614, 2013.

[67] A. Younus and B. Yang. *Wavelet co-efficient of thermal image analysis for machine fault diagnosis*. In International Conference on Mechanical Engineering, pages 1–7, 2009.

[68] A. Younus and B. Yang. *Intelligent fault diagnosis of rotating machinery using infrared thermal image*. Expert Systems with Applications, 39:2082–2091, 2012.

[69] G. Lim, D. Bae, and J. Kim. *Fault diagnosis of rotating machine by thermography method on suport vector machine*. Journal of Mechanical Science and Technology, 28(8):2948–2952, 2014.

[70] A. Nembhard, J. Sinha, A. Pinkerton, and K. Elbhbah. *Combined vibration and thermal analysis for the condition monitoring of rotating machinery*. Structural Health Monitoring, 13(3):281–295, 2014.

[71] S. Verstockt, S. Van Hoecke, P. De Potter, P. Lambert, C. Hollemeersch, B. Sette, B. Merci, and R. Van de Walle. *Multi-modal time-of-flight based fire detection*. MULTIMEDIA TOOLS AND APPLICATIONS, 69(2):313–338, 2014.

[72] S. Van Hoecke, R. Verborgh, D. Van Deursen, and R. Van de Walle. *SAMuS: service-oriented architecture for multisensor surveillance in smart homes*. SCIENTIFIC WORLD JOURNAL, pages 0–9, 2014.

[73] SKF. *Spherical roller bearings*. Online, October 2009.

[74] Schaeffler. *FAG split plummer block housings of series SNV*. Online, 2015.

[75] R Schulz, S Verstockt, J Vermeiren, M Loccufier, K Stockman, and S Van Hoecke. *Thermal Imaging for Monitoring Rolling Element Bearings*. In Quantitative InfraRed Thermography, pages 1–10, 2014.

[76] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. *Overview of the H. 264/AVC video coding standard*. IEEE Transactions on Circuits and Systems for Video Technology, 13(7):560–576, 2003.

[77] E. Bechhoefer and M. Praneet. *Bearing Envelope Analysis Window Selection*. In Proceedings of the Annual Conference of the Prognostics and Health Management Society, pages 1–7, 2009.

[78] R. Lyons. *Understanding digital signal processing*. Pearson Education, 2010.

[79] S. W. Smith. *The scientist and engineer's guide to digital signal processing*. 1997.

[80] S. Tretter. *Communication System Design Using DSP Algorithms*. Springer Science & Business Media, 2008.

[81] R. Caruana and A. Niculescu-Mizil. *An empirical comparison of supervised learning algorithms*. In Proceedings of the 23rd international conference on Machine learning, pages 161–168. ACM, 2006.

[82] N. Otsu. *A Threshold Selection Method from Gray-Level Histograms*. Systems, Man and Cybernetics, IEEE Transactions on, 9(1):62–66, 1979.

[83] E. Rosten and T. Drummond. *Machine learning for high-speed corner detection*. In European conference on computer vision, pages 430–443. Springer, 2006.

[84] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. *ORB: An Efficient Alternative to SIFT or SURF*. In Proceedings of the 2011 International Conference on Computer Vision, pages 2564–2571, 2011.

[85] F. Selka, V. Agnus, S. Nicolau, A. Bessaid, L. Soler, J. Marescaux, and M. Diana. *Biomedical Image Registration: 6th International Workshop, WBIR 2014, London, UK, July 7-8, 2014. Proceedings*, chapter Fluorescence-Based Enhanced Reality for Colorectal Endoscopic Surgery, pages 114–123. Springer International Publishing, 2014.

[86] J. M. Lotz, J. Primack, and P. Madau. *A New Nonparametric Approach to Galaxy Morphological Classification*. Astronomical Journal, 128:163–182, 2004.

[87] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[88] A. Krizhevsky, I. Sutskever, and G. Hinton. *Imagenet classification with deep convolutional neural networks*. In Advances in neural information processing systems, pages 1097 – 1105, 2012.

[89] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[90] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. In International Conference on Learning Representations, 2015.

[91] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[92] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. *How transferable are features in deep neural networks?* In Advances in neural information processing systems, pages 3320–3328, 2014.

[93] Ali Sharif R., H. Azizpour, J. Sullivan, and S. Carlsson. *CNN features off-the-shelf: an astounding baseline for recognition*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 806–813, 2014.

[94] W. Zhang, R. Li, T. Zeng, Q. Sun, S. Kumar, J. Ye, and S. Ji. *Deep model based transfer and multi-task learning for biological image analysis*. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1475–1484. ACM, 2015.

[95] F. Hu, G. Xia, J. Hu, and L. Zhang. *Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery*. Remote Sensing, 7(11):14680–14707, 2015.

[96] B. Li, M.-Y. Chow, Y. Tipsuwan, and J. C. Hung. *Neural-network-based motor rolling bearing fault diagnosis*. IEEE transactions on industrial electronics, 47(5):1060–1069, 2000.

[97] Z. Chen, C. Li, and R. Sanchez. *Gearbox Fault Identification and Classification with Convolutional Neural Networks*. Shock and Vibration, 2015:10, 2015.

[98] M. Amar, I. Gondal, and C. Wilson. *Vibration spectrum imaging: A novel bearing fault classification approach*. IEEE Transactions on Industrial Electronics, 62(1):494–502, 2015.

[99] N.K. Verma, V.K. Gupta, M. Sharma, and R.K. Sevakula. *Intelligent condition based monitoring of rotating machines using sparse auto-encoders*. In IEEE Conference on Prognostics and Health Management (PHM), pages 1 – 7, 2013.

[100] V. Slavkovikj, S. Verstockt, W. De Neve, S. Van Hoecke, and R. Van de Walle. *Hyperspectral Image Classification with Convolutional Neural Networks*. In Proceedings of the 23rd Annual ACM Conference on Multimedia Conference, MM '15, pages 1159–1162, 2015.

[101] I. Sutskever, J. Martens, G. E. Dahl, and G. Hinton. *On the importance of initialization and momentum in deep learning*. In Proceedings of the 30th International Conference on Machine Learning (ICML-13), volume 28, pages 1139–1147, 2013.

[102] M. D. Zeiler and R. Fergus. *Visualizing and Understanding Convolutional Networks*. In Computer Vision ECCV 2014, Lecture Notes in Computer Science, pages 818–833. Springer International Publishing, 2014.

[103] P. Cambron, C. Masson, A. Tahan, and F. Pelletier. *Control chart monitoring of wind turbine generators using the statistical inertia of a wind farm average*. Renewable Energy, pages –, 2016.

[104] M. Hermann, T. Pentek, and B. Otto. *Design Principles for Industrie 4.0 Scenarios*. In 2016 49th Hawaii International Conference on System Sciences (HICSS), pages 3928–3937. IEEE, 2016.

[105] S. Van Hoecke, R. Verborgh, D. Van Deursen, and R. Van de Walle. *SAMuS: service-oriented architecture for multisensor surveillance in smart homes*. SCIENTIFIC WORLD JOURNAL, pages 1–9, 2014.

[106] L. Da Xu, W. He, and S. Li. *Internet of things in industries: A survey*. IEEE Transactions on Industrial Informatics, 10(4):2233–2243, 2014.

[107] W. He, G. Yan, and L. Da Xu. *Developing vehicular data cloud services in the IoT environment*. IEEE Transactions on Industrial Informatics, 10(2):1587–1595, 2014.

[108] B. Xu, Li Da X., H. Cai, C. Xie, J. Hu, and F. Bu. *Ubiquitous data accessing method in IoT-based information system for emergency medical services*. IEEE Transactions on Industrial Informatics, 10(2):1578–1586, 2014.

[109] D. Hoang, T. Nguyen, and A M. Tjoa. *Dashboard by-example: A hypergraph-based approach to on-demand data warehousing systems*. In 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 1853–1858. IEEE, 2012.

[110] M. Leida, D. Xiaofeng, P. Taylor, and B. Majeed. *Toward automatic generation of SPARQL result set visualizations: A use case in service monitoring*. In e-Business (ICE-B), 2011 Proceedings of the International Conference on, pages 1–6. IEEE, 2011.

[111] J. Gitanjali, M. Kuriakose, and R. Kuruba. *Ontology and Hyper Graph Based Dashboards in Data Warehousing Systems*. Asian Journal of Information Technology, 13(8):412–415, 2014.

[112] S. Mazumdar, A. Varga, V. Lanfranchi, D. Petrelli, and F. Ciravegna. *A knowledge dashboard for manufacturing industries*. In Extended Semantic Web Conference, pages 112–124. Springer, 2011.

[113] M. Lanthaler and C. Gütl. *On using JSON-LD to create evolvable RESTful services*. In Proceedings of the Third International Workshop on RESTful Design, pages 25–32. ACM, 2012.

[114] R. Verborgh, T. Steiner, R. Van de Walle, and J. Gabarro. *The missing links: how the description format RESTdesc applies the linked data vision to connect hypermedia APIs*. In Proceedings of the Workshop on Linked APIs for the Semantic Web, page 8. Ghent University, Department of Electronics and information systems, 2012.

[115] R. Verborgh and J. De Roo. *Drawing Conclusions from Linked Data on the Web: The EYE Reasoner.* IEEE Software, 32(3), 2015.

[116] *Web Design Issues; What the Semantic Web can represent.* `http://www.w3.org/DesignIssues/RDFnot.html`. Accessed: 2016-06-28.

[117] *Key World Energy Statistics 2015*. `https://www.iea.org/publications/freepublications/publication/KeyWorld_Statistics_2015.pdf`. Accessed: 2016-02-15.

[118] K Holmberg, P. Andersson, and A. Erdemir. *Global energy consumption due to friction in passenger cars*. Tribology International, 47:221 – 234, 2012.

[119] L. Liebrecht, X. Si, B. Sauer, and H. Schwarze. *Investigation of Drag and Churning Losses on Tapered Roller Bearings*. Strojniki vestnik - Journal of Mechanical Engineering, 61(6), 2015.

[120] *SKF Energy Efficient (E2) bearings*. `http://www.skf.com/group/industry-solutions/electric-motors/industrial-electric-motors-and-generators/applications/general-purpose-motors/energy-efficient-e2-bearings.html`. Accessed: 2016-02-15.

[121] D. H. Ballard. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*. chapter Generalizing the Hough Transform to Detect Arbitrary Shapes, pages 714–725. Morgan Kaufmann Publishers Inc., 1987.