Detecting the Onset of Machine Failure Using Anomaly Detection Methods

by

Mohammad Riazi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

When a machine or a component of a machine fails, corrective maintenance is performed to identify the cause of failure and decide on a repair mechanism to restore the machine to its normal working condition. However, because the machine has failed without any prior warning, a considerable amount of time for procuring and repairing the failed component is required. Since machines and their components generally degrade through time, i.e., start in a normal condition and progress to failure, and the time of failure is not known in prior, a maintenance strategy will need to be considered to minimize the downtime of the machine and the service(s) it provides. As such, time-based maintenance strategies are predominantly used for maintaining the healthy condition of machines and equipment through a regular maintenance schedule. This helps with the extension of the operational reliability of the machine, preventing potential catastrophic failure(s), and reducing time required for maintaining the equipment due to pre-planning. Nevertheless, substantial disadvantages such as expenses for hiring expert technicians, replacement of parts regardless of considerable remaining useful life, imminent failure(s) in between scheduled maintenance, and increased risk of failure due to improper diagnosis or intrusive repair mechanisms has triggered the interest of the research community in further investigating other strategies for maintaining the healthy condition of machines. Hence, prognostics and health management through condition-based maintenance is suggested as an alternative strategy. PHM is an enabling discipline consisting of technologies and methods to evaluate the reliability of a

product in its actual life cycle to determine the development of failure and mitigate system risk. Sensor systems are needed for PHM for online monitoring of an equipment. This strategy is referred to as CBM. PHM can be implemented in several ways: Physics-based, knowledge-based and data-driven-based.

We propose anomaly detection as a data-driven technique for the early detection of fault in a machine. Anomaly detection is described as the task of identifying observations with anomalous behaviour (a fault) that can cause systems to deviate from their normal operating conditions in an unacceptable way.

For the purpose of this study, a belt-driven robot arm test platform is designed. The robot arm is conditioned on the torque that is required to move the arm forward and backward, simulating a door opening and closing operation. A number of failures are simulated and data is collected. Several anomaly detection methods namely, k-NN, LOF, ABOD, HBOS, isolation forest, one-class SVM, PCA, T*-framework and deep neural network-based models including feed forward and convolutional neural network auto-encoders are tested. Data for normal condition and several simulated failures, e.g., loose belt tension, high temperature, etc. is collected. The normal operating conditions of the arm is learnt by the anomaly detection methods through training on samples of the normal only class and a threshold is obtained. The learnt model is then used on unseen test data which includes samples of normal data mixed with samples of failure modes and an anomaly score is computed per observation. The scores are then compared to a previously computed threshold which determines the normal or anomalous label. The performance of each model is evaluated using the precision, recall, F1-score, area under the receiver operating characteristic curve metrics and the average time required to train and test a model. Our results show that, the onset of failure can indeed be detected with a very high precision and recall and with an average F1-score of over 90%

for majority of the algorithms. Moreover, we further investigate feature-based anomaly detection of the torque time series data with hand-crafted descriptive statistic features and automatic features extracted from the convolutional auto-encoder. The reduction of dimensions through manual feature engineering show a positive impact both on the allocation size of data and on the performance of the models in terms of accuracy, time to train and test, and the size of the fitted models. The robot arm dataset is made available to the research community and the results of our comparative assessment of anomaly detection algorithms brings a significant potential contribution to the PHM and the CBM research field.

Preface

The design and development of the robot arm is a joint collaboration with the Department of Mechanical Engineering MECE lab. The data acquisition and cleaning scripts have been written by myself. I have also participated in the calibrating of the strain gauge installed on the belt for measuring tension.
Chapter three of this thesis: robot arm components and operation of the robot arm is co-authored with Anthony Maltais.
I am intending to publish the results of this research as a comparative assessment of outlier detection algorithms for the purpose of detecting onset of machine failure in the future.

To my family, specially my wife,
For providing me with constant support and continuous encouragement
throughout my years of study.

Acknowledgements

I would first like to thank my supervisors Professor Osmar Zaiane and Professor Micheal Lipsette for their valuable guidance and great supervision throughout this journey.

I would also like to thank Professor Lipsett's lab members, Nicolas Olmedo and Anthony Mathias for their great help with the design and development of the robot arm platform, without their support I could have not completed this project.

I am thankful to Dr. Colin Bellinger and Dr. Johannes Gunther for their constructive feedback and valuable recommendations.

I would also like to thank my friend and colleague Tomoharu Takeuchi from Mitsubishi Electric Co. for his constant support and useful feedbacks with regards to this research. Moreover, my sincere gratitude to Mitsubishi Electric Co. for providing the funding for this research and the wonderful opportunity to experience Japan.

Finally, I must express my very profound gratitude to my wife providing me with constant support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

Contents

List of Tables

List of Figures

Acronyms

ABOD:	Angle-Based Outlier Detection
AE:	Acoustic Emission
ANN:	Artificial Neural Network
AR:	Autoregressive
ARMA:	Autoregressive Moving Average
AUC:	Area Under Curve
AUROC:	Area Under Receiver Operating Characteristic Curve
BMU:	Best Matching Unit
CBM:	Condition-Based Maintenance
CNN:	Convolutional Neural Network
CNNAE:	Convolutional Neural Network Autoencoder
CSV:	Comma-Separated Value
DBSCAN:	Database
DC:	Direct Current
DOF:	Degree of Freedom
EM:	Expectation Maximization
FCM:	Fuzzy C-Mean
FFAE:	Feed Forward Autoencoder
FFT:	Fast Fourier Transform
FMMEA:	Failure Modes Mechanisms and Effects Analysis
GMM:	Gaussian Mixture Model
GPU:	Graphical Processing Unit
GVF:	General Value Function
HBOS:	Histogram Based Outlier Score
***k*-NN:**	k – Nearest Neighbours

LED:	Light Emitting Diode
LOF:	Local Outlier Factor
LRD:	Local Reachability Density
MAE:	Mean Absolute Error
MD:	Mahalanobis Distance
MF:	Machine Failure
MLE:	Maximum Likelihood Estimation
MQE:	Minimum Quantization Error
mRMR:	minimum Redundance Maximum Relevance
MSE:	Mean Square Error
OCSVM:	One-Class Support Vector Machine
PCA:	Principal Component Analysis
PHM:	Prognostics and Health Management
ROC:	Receiver Operating Characteristic
ROS:	Reference-Based Outlier Score
RUL:	Remaining Useful Life
SCADA:	Supervisory Control and Data Acquisition
SOM:	Self-Organizing Map
SVM:	Support Vector Machine

Chapter 1

Introduction

The considerable costs and risks pertaining to improper maintenance have been repeatedly reported and documented in the industry. A 2017 International Data Coporation (IDC) survey [4] highlights the fact that quality continues to be a priority for manufacturing and that manufacturers are looking for ways to improve product and or service quality. Numerous factors can contribute to the quality of a product, and not every one of these factors are under manufacturers' control. However, it is fortunate that one of the most common sources of quality problems is *faulty equipment* that has not been properly maintained [57]. Poorly functioning machines, machinery components and unreliable products or services are not good for a company's business. Therefore monitoring the condition of these machines and components such as cooling fans, bearings, turbines, gears, belts and maintaining a desirable working state becomes very crucial because maintenance is directly linked to competitiveness and profitability which in turn determine the prospects of a company.

Since the beginning of the industrial revolution, machinery components have been used extensively in many sectors. However, till today, the biggest and most important issue has been reliability. Reliability of a machine can be decomposed to three important factors:

- Cruciality of failure —Failure of component(s) could lead to a complete failure of a system.
- Frequency of failure —The rate at which a component fails plays an

important role in multitude of applications.

- Service suspension —When a system/component fails, the amount of time required to repair and maintain it may hinder service provisioning.

1.1 Background

When a machine or a component fails, corrective maintenance is performed to identify the cause of failure and decide on repair procedures required to maintain and reinitiate the machine to its normal working conditions. However, because the machine has already failed without any prior warnings, time is required for procuring and repairing of the failed component. Because machines and its components degrade through time, i.e., start at a normal healthy condition and progress to failure, and the time of failure is not known in advance, a maintenance strategy needs to be considered to minimize the downtime of a service.

Hence, time-based maintenance strategies are predominantly used for maintaining the conditions of machines and equipment. Using this method, a scheduled maintenance is developed and regular maintenance is performed. There are many advantages to using regular scheduled maintenance. Due to its regular checks, the operational reliability of the machine is extended and catastrophic failures are somewhat prevented. Second, the time required to maintain the equipment is drastically reduced because the necessary tools and resources are obtained well in advanced. However, there still remains substantial disadvantages that has triggered the interest of the research community in further investigating other possible strategies. These shortcomings include, costs pertaining to tools and experts performing regular maintenance even-though it might not be necessary at all. Another problem is an occurrence of failure in between scheduled maintenance, which results in previously mentioned limitations due to unexpected failure. Moreover, during a regular maintenance a part may end up being replaced regardless of its considerable Remaining Useful Life (RUL), which incurs additional costs and expenses. Last but not least, the risk of failure may increase during maintenance due to

improper diagnosis and or because of intrusive measures taken during repairs.

Due to the unreliability of this strategy, industries are switching to a more reliable program, namely, Prognostics and Health Management (PHM). PHM is an enabling discipline consisting of technologies and methods to evaluate the reliability of a product in its actual life cycle conditions to determine the development of failure and mitigate system risk [15]. Sensor systems are needed for PHM for online monitoring of an equipment. With online monitoring the condition of the equipment is constantly checked throughout its life cycle to identify any potential failure. This strategy is referred to as Condition-Based Maintenance (CBM). In general, PHM consists of sensing, anomaly detection, diagnostics, prognostics and decision support [59] as shown in Figure 1.1.

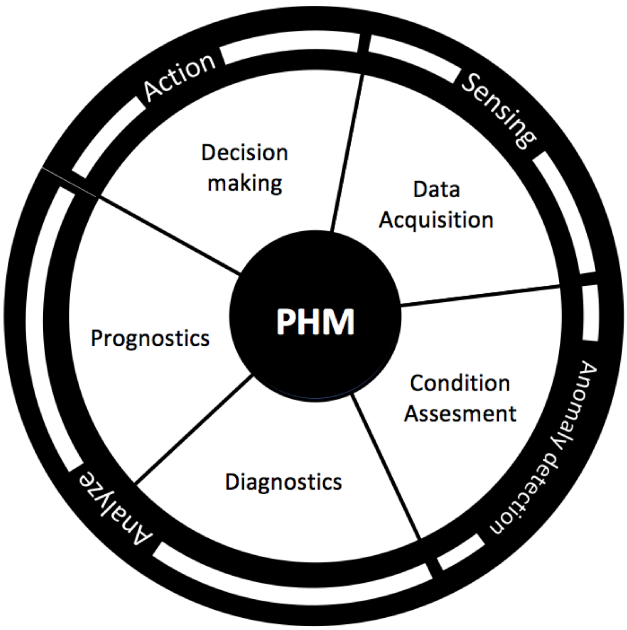

Figure 1.1: PHM framework [partially adapted from [59]]

Sensing is for collecting a history of the operational condition of a product. The main purpose of anomaly detection is to identify unexpected and unusual (anomalous) behaviour of a component via deviations from healthy state. Anomaly detection results can then be used as an advanced warning system referred to as failure precursors. It should be noted that anomalies do not necessarily indicate system failures as the change in operating and envi-

ronmental conditions may influence data from sensors and indicate anomalous behaviour. Although these warnings are not considered anomalies, however it could provide valuable information to the overall PHM system showing unexpected use of system. Diagnostics and prognostics are considered as two important aspects in PHM [36]. Diagnostics is the process of detecting a fault; recognizing whether something is wrong with the system, isolating the fault; locate the part that is faulty, and finally identifying the nature of the fault. Prognostics on the other hand deals with predicting and estimating the RUL of the product, which most of the time requires additional information not usually provided by sensors, such as environmental factors, past and future operating profiles and maintenance history. Ultimately the goal of a complete PHM system is to make intelligent decisions about the system health and to arrive at strategic and business case decisions [59].

PHM can be implemented in several ways: physics-based models, knowledge-based models and data-driven models. Physics-based models are usually based on mathematical models. Health index and what influences the health state of physical components are derived by mathematical differential equations by domain experts. Similar to the physics-based methods, knowledge-based methods also take experts' knowledge into consideration, however this time physical behaviour is not formulated by mathematical model. Knowledge-based systems try to formalize the extensive knowledge of the domain expert.

During recent years, data-driven models also known as data mining methods or machine learning methods for machine health monitoring are becoming more attractive due to the substantial development of sensors and computing platforms. These approaches are considered more generic than physics-based and knowledge-based models. Data-driven methods use historical data to automatically learn a model of system behaviour. Features that encompass the health state of a machine are extracted using different procedures such as noise removal, standardization and transformation. These features are then used by a variety of machine learning methods to decide on the health state of the machine. Based on the availability of data, different learning techniques such

as supervised, semi-supervised, unsupervised or reinforcement learning could be used to achieve the desired result. We will briefly discuss the different PHM approaches with an emphasis on data-driven-based methods in Chapter 2 of this manuscript.

As mentioned previously, fault detection is a major task across many applications [2] including quality control, finance, security and medical. In an industrial or manufacturing settings, an equipment or product fault is an abnormal behaviour that may lead to the total failure of a system. Often, when a fault initiates, the machine can still function as it was programmed to do so until the fault progresses to a certain degree. In general, there is a time gap between the appearance of a fault and its ultimate failure. PHM can be used to reduce the probability of failure by monitoring the stages of failure throughout its development and prevent fault from becoming a failure in an efficient manner.

In data-driven PHM, fault monitoring can be obtained using anomaly detection techniques via learning from historical data [12]. The task of anomaly or novelty detection can be described as the detection of differing test data with regards to the training data during the learning phase. In other word, when a system is faulty, the monitored data no longer follows the normal behaviour and thus it is considered as anomalous. The methods are usually employed in situations where large amounts of "normal" condition sample data is available and data describing the "abnormal" condition are insufficient or not conveniently accessible.

1.2 Motivation

When machines are regularly used, there are variety of conditions that can change in the system that would result in a non-optimal performance. Many faults in a mechanical system are very obvious and can be found in routine maintenance inspections, however, as mentioned previously, regular time-based maintenance is not economically feasible nor operationally convenient as it provokes service inaccessibility. Moreover, there are some faults that occur

very subtly and can be difficult to detect. These faults may not completely interrupt the operability of the machine, but they can cause the system to operate in non-optimal ways, potentially accelerating the fatigue of various components or increasing the risk of a catastrophic failure. Therefore, it is important that all faults — including the ones that are not apparent but may have substantial impact on the system — be monitored and managed.

The major challenge with most industrial systems is that they come in many variations. Moreover, the relationship between the many parts and components can sometimes be very complex, which makes understanding the governing laws very difficult and increases the space for possible abnormal modes. In addition, industrial machines are purposely designed to operate for very long periods of time, therefore, data collected on abnormalities can be rare and may not be known a priori, hence precluding the use of supervised approaches for fault detection. In general, acquiring a set of labeled anomalous data which cover every possible type of abnormal behaviour is a lot more difficult than getting labels for normal behaviour.

1.3 Thesis statement

We propose a possible solution to the above mentioned dilemma where lack of abnormal data is a major issue, and that is using semi-supervised anomaly detection methods for detecting the onset of failure. Based on the availability of data, detecting anomalous behaviour can operate in one of three categories: supervised anomaly detection, which assumes availability of a labeled normal and abnormal class(s) for training; semi-supervised anomaly detection, which assume training data has labeled samples for only the normal or positive class, and unsupervised anomaly detection which do not require any labeled training dataset. The proposed technique in general involves the learning of the normal profile by several anomaly detection algorithms through training on various samples of the normal only class and using the learnt model on test data and outputting some kind of an anomaly score. This score, which may or may not be probabilistic is then compared to a pre-defined threshold to mark the

decision on the test data as being normal or anomalous.

Generally, there is a time period between the advent of a failure (fault) and the occurrence of an actual failure. If the fault is detected in an early stage (onset) before leading to complete failure of a machine, one can reduce the failure rate and perform optimized maintenance. In this research, we are interested in detecting the early stage of a failure or the onset of failure using semisupervised anomaly detection. Using this approach, it is assumed that test data that have a higher score than the set threshold learned by the anomaly detection model is behaving different than that of the normal condition and we can conclude that this may be the start or the onset of a future failure.

1.4 Thesis objectives

Unfortunately, there are no freely published datasets available on machine degradation through time. The purpose of this study is two fold; design and development of a robot-arm platform that can produce data at different operational states and to apply and compare anomaly detection techniques on data collected via the platform to assess the practicality of these algorithms for detecting the onset of failure in machines.

The objectives of the present study can be summarized as:

- Design and development of a robot-arm platform which can simulate faults that can occur during regular operation of the arm.
- Assessing and evaluating the application of machine learning techniques, specifically anomaly/outlier detection methods for detecting the onset of failure for a belt-driven single degree of freedom mechanical system - the robot arm.

Torque signals collected from the robot-arm platform is given as input to a number of different anomaly detection techniques and an anomaly score is assigned to each test signal. As a first step some classical anomaly detection techniques were used with the available data. We then investigated the use of deep learning methods for the detection of anomalies to further improve

the results from the initial tests using classical methods. Some state-of-the-art techniques using deep learning, such as various auto-encoders (feedforward and convolutional neural networks) were compared. The results of our experiment do not show a significant improvement in terms of F1-score, however deep learning methods designed as auto-encoders do provide an advantage over the classical methods, and that is the possibility of dimension reduction and stacking of layers for creating a model that is able to handle more sophisticated input signals. Moreover, the reduced dimensions provides a means of automated feature extraction which could be used with any anomaly detection method or used in an end-to-end structure to detect anomalous behaviour.

1.5 Thesis contributions

- In this study, a single degree of freedom mechanical system, referred to as robot-arm platform was built and developed. The robot-arm is capable of generating normal operational torque data as well as eventual torque degradation signals.

- The robot arm data is made freely accessible to the research community where the lack of such data is substantially evident.

- Multiple anomaly detection techniques were applied and tested on the data collected from the robot arm platform and their applicability for detecting onset of failure were assessed and confirmed.

- Multiple variations of auto-encoders were also implemented and used with the robot-arm data to assess the applicability and evaluate the outcome of using deep learning for detecting the onset of failure. Comparable results was achieved with the additional advantage of dimensionality reduction.

1.6 Manuscript organization

In this chapter we have discussed and elaborated the motivation behind this research study and proposed anomaly detection as a possible solution to the challenging task of detecting the onset of failure in machines. The remainder of this manuscript is organized as follows: Chapter 2 provides a general overview of the PHM methods and reviews some of the past and current approaches to detecting anomalous behaviour in industrial machines and acquaint the reader with various existing anomaly detection techniques and justify the use of some of these techniques for the case of robot arm experiment. In Chapter 3, the experimental design including the robot arm platform, its main components and data acquisition is discussed. The experiments and results of applying several anomaly detection techniques is presented in Chapter 4. We then discuss the results in a more detail in Chapter 5 and conclude the research followed by future directions.

Chapter 2

Literature Review

An equipment or product's health state can be described as the level of degradation or divergence from its expected normal operating conditions. Therefore, to acquire the PHM of a product or a system, it is critical to identify any deviation from the nominal healthy behaviour and detect the onset of a potential failure. As mentioned previously in Chapter 1, PHM can be implemented using different approaches: physics-based, knowledge-based and data-driven-based. In the following sections, we describe these approaches with more emphasis on the data-driven approaches specially different categories of anomaly detection techniques and how they are used for machine fault detection.

2.1 Physics-based models

Physics-based models are usually based on mathematical models [33]. Health index and what influences the health state of physical components are derived by mathematical differential equations. These equations incorporate the knowledge of hardware including properties of material and structure of machinery and operational (torque, duty cycles, etc.) and environmental (temperature, pressure, humidity, etc.) loads which can predict the reliability and remaining useful life of the components. This method has its advantages and disadvantages. An example of using physics-based modelling for fault detection is demonstrated in a work by Weber et al., [82] for fault diagnosis and fault tolerant control in wind turbines using speed sensors. In another study by Walter et al., [7], two models of gearboxes for two mechanical systems, namely

the belt conveyor and the bucket wheel excavator is constructed. The authors claimed that the models allow better understanding of the phenomena taking place in the actual operating conditions of machines. The biggest advantage is that if the model is well designed it is an interpretable and reliable method. The greatest downside however, is the need for domain experts who have a very high understanding of the system to build the model. The need for expert knowledge dictates high cost and limits the usage of the model as it is designed for a specific component [11]. In addition, the accuracy of these models is influenced by many environmental factors which would be complex to monitor. Moreover, many of these physics-based models cannot cope or update with the online measured data which limits their effectiveness and flexibility.

2.2 Knowledge-based models

Similar to the physics-based methods, knowledge-based methods also take experts' knowledge into consideration, however this time physical behaviour is not formulated by mathematical model. Knowledge-based systems try to formalize the extensive knowledge of the domain expert. For example, in expert systems, rules describe the state of a system. These rules are usually represented in the form of `|if...then|` statements. Because there are various situations and conditions that could occur in a real-world system, not all rules and conditions could be implemented. Even if this was possible, not all knowledge from experts could be easily converted to rule-based system. Therefor knowledge-based models cannot be used alone to detect failures [76]. However, these models can be combined with data-driven models that could achieve better results or an optimal solution. For example, in [85], a fault detection method that combines data-driven techniques with association rules to detect failure in house-keeping data in spacecraft systems is proposed.

2.3 Data-driven-based models

During the recent years, data-driven models also known as data mining methods or machine learning methods for machine health monitoring are becoming

more attractive due to the substantial development of sensors and computing platforms. This approach uses historical data to automatically learn a model of system behaviour, which are considered more generic than physical and knowledge-based models.

One of the most commonly used data-driven method for machine fault detection is outlier or anomaly detection. Chandola et al., defines an anomaly or fault as an anomalous behaviour that causes a system to deviate from its normal operating conditions or states in an unacceptable way. Anomaly detection has been applied in many fields such as network intrusion detection, fraud detection, sensor network fault detection, medical diagnosis and many others which are thoroughly reviewed in [13].

In PHM , anomaly detection becomes a very important task because anomalies in data translate to significant information about a product's health state. Based on availability of data, anomaly detection using machine learning techniques can be categorized into supervised, when both normal and abnormal cases are available, unsupervised where no labeled data is available, and semisupervised anomaly detection when only one class (normal) label is available.

2.3.1 Supervised anomaly detection

Supervised learning technique is the common technique when historic labeled data is available. These labeled data are past sensor recordings and machine log files. In the particular case of PHM, labels are system health state indicators that determine the health status of a system. Through some suitable machine learning technique these data are examined and a model is learned. As new data arrive, it is fed into the model and a health state is predicted. Commonly used supervised learning techniques include but is not limited to Support Vector Machines (SVM), k-Nearest Neighbour (k-NN), Artificial Neural Networks (ANN), tree-based methods and regression models.

2.3.2 Unsupervised anomaly detection

In some cases, historic data is available, however there are no labeled data or target values associated with the input values. Hence unsupervised anomaly detection methods may be used. The objective of such algorithms is to find patterns in the historic data that could be grouped together in the form of clusters. Another goal of the unsupervised learning approach is to determine the distribution of data in an input space. Looking at the results of the unsupervised methods may reveal some information about the state of machine and failure detection.

2.3.3 Semisupervised anomaly detection

On the other hand, in most cases ample data for a healthy condition system is available but labeled failure data are rare if not available at all. In this scenario semi-supervised anomaly detection methods are used. These methods try to learn the normal profile of a machine using the available normal-only training dataset. At test time, samples that do not conform to that of the normal profile are flagged anomalous. The results can be either in the form of a binary outcome or an anomaly score. This is also known as one-class classification [54].

2.4 Review of anomaly detection methods

There exists numerous anomaly detection methods. We have roughly grouped some of these methods into statistical algorithms, clustering-based, nearest neighbour-based, classification-based, spectral-based, space subsampling-based and deep learning methods. Each category and related algorithms with its applications (if available) will be discussed further in individual sections.

2.4.1 Statistical-based methods

Statistical-based methods assume that the data follows a specific distribution, so the model is created from a specified probability distribution [18]. Subsequently, the simplest approach for detecting anomalies in data would be

flagging data points that deviate from common statistical properties of a distribution, including the mean, median, mode and quantiles. For example, one could define an anomaly based on a certain standard deviation away from the mean. The advantage of these models is that they output a probability as a measure of outlierness.

There are parametric [19] and non-parametric models [20] that can be used to define a probability distribution. The major assumption with parametric methods is that the normal observations in a training dataset can be presented by parameters θ of specific statistical distribution. A commonly used distribution is the Gaussian distribution for which the parameters of the distribution can be determined using the training dataset through Maximum Likelihood Estimators (MLE). To determine if a test observation is anomalous, the inverse probability distribution function of the distribution with learned parameters is used as the decision function. For more complex distributions, the data may be modelled using a mixture of models like Gaussian Mixture Models (GMMs) [51]. Non-parametric methods do not assume a fixed structure for the dataset and are flexible to fit a complex dataset as required. A histogram-based model is an example of a basic non-parametric statistical method. The histogram is modelled using the normal training dataset and to check for a sample test abnormality, a distance measure to the normal is used [13]. The advantages and disadvantages of statistical anomaly detection methods is given in Table 2.1.

Advantages	Disadvantages
• If the underlying data distribution is correctly identified (assumed), these methods present a statistically proven solution for anomaly detection. • The resulting anomaly score is associated with a confidence interval. • Depending on the robustness of the distribution to anomalies within data, these methods can be used in an unsupervised fashion.	• Parametric statistical methods assume that the data is from a specific distribution, which is normally not the case in a real world setting with high dimensional data. • Require a large sample of training data to estimate model parameters

Table 2.1: Advantages and disadvantages of statistical anomaly detection methods

2.4.1.1 Gaussian Mixture Models (GMMs)

GMMs are able to estimate the probability density of a normal training dataset (target class) assuming they are from different normal distributions [66]. Because in real life, many datasets follow Gaussian distribution, the method tries to model the dataset as a mixture of several Gaussian distributions. The parameters of the model can be estimated using several methods such as MLE or Expectation-Maximization (EM).

Zorriassatine et al., [91] use GMMs in condition monitoring of multivariate milling processes for pattern recognition. In their study, the condition of several healthy milling process is monitored and signals are collected for each machining process. Various models for the underlying healthy probability density function of the machining process at given combinations of machining parameters, such as depth of cut, speed and feed rate is computed. The centre of each Gaussian distribution was calculated and initialized using a k-means clustering algorithm and all model parameters were computed using EM. The GMM with the smallest training error was then used as the anomaly detector. The threshold for detecting anomalies was set to be the minimum log-likelihood of the training data.

2.4.1.2 Regression models

Anomaly detection using regression is considered a parametric statistical technique where the data is explained by a regression model. At its most basic setting, the technique involves two steps: a regression model is fit onto the data, and in a second step, for every test sample, the residual of the test and the results from the model determine the anomaly score. A major downside with the regression-based anomaly detection is that they are prone to outliers in training data [13]. Therefore some variants of the regression method have been proposed by researchers such as robust regression [67][8] for fitting the data.

Aboul-Yazeed et al., [1], proposed an auto regressive time series model to predict future failure rate of a Haematology medical equipment using historical data. The equipment was observed for three years for collecting failure history data. Using this data, time between two failures (failure time rate) was calculated. The failure data rate is considered as stationary, therefore an Autoregressive (AR) model was used for the analysis. They compared the predicted failure rate with actual failure rate and were able to show an accurate result with less than 0.1% Mean Square Error (MSE). The authors further showed that their model can predict failures within a two-day time frame.

Zhao et al., [89] used the Autoregressive Moving Average (ARMA) method proposed by [9] for forecasting future failure events based on past data. To handle the non-stationary nature of the data, the Machine Failure (MF) attribute is created: MF = Down Time / (Down Time + Productive Time) x 100%. Subsequently the residual time series was computed by moving average method which resulted in a stationary time series ready for input to the ARMA model. The model was further tested on a real-world semiconductor manufacturing data. A total of 32 MF values were observed and forecasted. The comparison of the results with the actual MF values using Mean Absolute Error (MAE) was less than 2.48%.

2.4.1.3 Histogram-based Outlier Score (HBOS)

The histogram-based outlier score [27] is a very simple non-parametric statistical algorithm. This method assumes independence amongst features which makes it much faster than multivariate approaches having less precision as the downside. The way this algorithm works:

1. For every feature or dimension d, a univariate histogram is constructed, where the height of each single bin represents a density estimation. Two different methods can be used:

   - Static bin-width histograms
   - Dynamic bin-width histograms
     For anomaly detection tasks the dynamic width mode is recommended, because the density estimation is more robust against large outlier values [27].

2. Number of k needs to be set. An often rule of thumb is setting k to the square root of the number of samples.

3. Histograms are then normalized such that the maximum height is 1.0 to ensure an equal weight for each feature when determining the outlier score.

4. Finally, the HBOS of every instance p is calculated using the corresponding height of bins where the instance is located:

$$HBOS(p) = \sum_{i=0}^{d} log(\frac{1}{hist_i(p)})$$

2.4.2 Clustering-based methods

In clustering-based anomaly detection, the assumption is that data points that are similar belong to a similar group. This is determined by the distance to the cluster centroid. Anomaly score is then calculated by setting a threshold

for the size of a cluster or the distance to the cluster centroid; if cluster has data points less than the value of threshold they are marked as anomalies, or if the data point's distance to the centre of the cluster exceeds the set threshold it is flagged as anomalous. Clustering is primarily thought as an unsupervised technique however semi-supervised clustering techniques have also been studied. A category of clustering algorithms use a clustering method to first cluster the data and then for each sample in a cluster the distance to its nearest cluster centroid is computed as an anomaly indicator. Self-Organizing Maps (SOM), k-means are examples of such methods. Clustering-based anomaly detection can be helpful in scenarios where there exists multiple varying operating conditions and each condition needs to be clustered and used as a separate reference.

Table 2.2 lists the advantages and disadvantages of using clustering-based methods for anomaly detection.

Advantages	**Disadvantages**
• Can operate in an unsupervised fashion, without requiring any labeled data. • Capable of being used in incremental models, i.e., new data points. can be given and tested for anomaly. • Fast test time results due to testing against small number of identified clusters.	• Performance of these methods is very much dependent on the clustering algorithm and how successful they can identify the structure of normal data. • Some clustering methods force every data point to be part of a cluster. Hence, clustering methods that assume anomalies do not belong to clusters consider these as normal. • Some methods only work well if anomalies do not form smaller clusters amongst themselves.

Table 2.2: Advantages and disadvantages of clustering anomaly detection methods

2.4.2.1 k-Means clustering

Perhaps one of the most popular and simplest unsupervised clustering algorithm for structured data is the k-means algorithm [50]. It works by choosing k random cluster centres and computes the distances between each point in the training set and the cluster centres. The Euclidean distance is a common choice, however other distance metrics such as the Mahalanobis distance could be used. Based on the computed distances, it then identifies points that are closest to a cluster centre and assigns it to the cluster. The cluster centroids are then recalculated using the mean of points within a cluster. The algorithm converges when the cluster centres do not move from one iteration to the next.

So, in other word, the k-means is trying to minimize total intra-cluster variance or the squared error function as follows:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{N} \left\| x_i^{(j)} - c_j \right\|^2$$

where x_i is the i^{th} data instance, i = 1,2,...,N, N is the number of data points in the given training dataset and c_j is the centroid of cluster j. There exists many modifications of the k-means clustering, some popular methods are the fuzzy versions of the k-means such as Fuzzy C-Means (FCM) [63]. Basically FCM allows a data point to belong to more than a single cluster.

Zhang and Kusiak [88] implemented an anomaly detection method using k-means clustering for wind turbines. The authors used the data for Supervisory Control and Data Acquisition (SCADA) system at the normal operating state of wind turbines and fit the k-means clustering algorithm onto the dataset. The model is then used to detect anomalous behaviour of incoming data from the SCADA system. Wang et al., used the k-means clustering algorithm for rolling bearing elements fault detection [79]. Clifton et al., employed the k-means clustering algorithm to condition monitoring of aero space gas-turbine engines [18]. Anomalous data are identified based on the number of standard deviations that a test point is from its closest cluster centre, relative to the distribution of all clusters. Fuzzy c-means clustering was used in a work by Baraldi et al., as an unsupervised clustering method to detect abnormal be-

haviours of process equipment [6]. In a different study, the authors evaluated the effectiveness of fuzzy based clustering on 148 shutdown transients of a nuclear power plant turbine [5].

2.4.2.2 Self-Organizing Maps (SOMs)

Kohonen Self Organizing maps (SOM) [43] provide a means to represent multi-dimensional data in a compressed and a much lower-dimensional space (usually one or two dimensions). In addition, the SOM is able to store information in a way that the topological relationships within the training set is preserved. What is noteworthy about SOMs is their ability to learn to classify data in an unsupervised fashion. SOM tries to cluster data samples by grouping similar data together. Basically, a SOM is made from several artificial neurons, each with their own weight vector usually the same dimension as the dimension of the input data. These neurons gradually adapt to the intrinsic shape of the dataset and are grouped based on the similarity of their weight vectors. A SOM learns the underlying shape of the dataset through an iterative process summarized in the following steps:

1. Determine the size of the SOM and randomly position (random initial weight) the neurons in the data space. There are no basis for determining the size of SOM. Hence, it has been determined empirically depending on the number of samples in training set [75] and as a rule of thumb it is:

$$M \approx 5\sqrt{N}$$

   where N is the number of samples in our dataset and M is the approximate number of neurons.

2. Choose a random vector from the set of training data.

3. Find the neuron that is most close to the chosen data point. This neuron is called the Best Matching Unit (BMU). A measure of similarity can be the Euclidean distance.

$$BMU = \underset{ij}{\operatorname{argmin}} \left\| x^{(t)} - w_{ij}^{(t)} \right\|$$

Where, t indicates iteration steps, x is sample data point and w is the weight vector of neuron with i and j indicating the number of rows and columns of the map respectively.

4. Move the BMU closer to that data point. The distance moved by the BMU is determined by a learning rate, which decreases after each iteration.

5. Move the BMU's neighbours closer to that data point as well, with farther away neighbours moving less. Neighbours are identified using a radius around the BMU, and the value for this radius decreases after each iteration.

6. Update the learning rate and BMU radius, before repeating Steps 2 to 5.

7. Convergence is reached once positions of neurons are stable.

Once the SOM is trained on the training dataset (normal healthy samples), BMUs are used as a decision function for determining if a test sample is normal or anomalous. The metric used as anomaly score is the Minimum Quantization Error (MQE), which is the Euclidean distance between a test sample x_{test} and its nearest w_{BMU_k}:

$$MQE = \min_k \|x_{test} - w_{BMU_k}\|$$

The higher the MQE value the more anomalous the test sample.

Du et al., used the SCADA data from wind turbine with a combination of features derived based on domain knowledge to build a normal behaviour model of the system based on SOM [22]. The SOM projects higher-dimensional SCADA data into a two-dimension-map. Afterwards, the Euclidean distance-based indicator for system level anomalies is defined and a filter is created to screen out suspicious data points based on quantile function.

2.4.3 Nearest neighbour-based methods

The main assumption with these techniques is that normal data samples appear in neighbourhoods that seem to be dense, while anomalies are far from their closest neighbours. Nearest neighbour methods can be generally grouped into distance-based and density-based methods. Both approaches require a similarity or a distance measure in order to make the decision on the degree of abnormality of a data instance. Table 2.3 summarizes the advantages and disadvantages of the nearest neighbour-based anomaly detection methods.

Advantages	**Disadvantages**
• Can operate in an unsupervised fashion, without any assumption of the generative distribution of data. • Suitable distance measures can be selected to cope with the characteristics of data.	• If normal instances do not have enough close neighbours, or if anomalies have enough close neighbours, they are highly likely to false alarms in terms of anomaly detection. • If normal test samples are different from that of training false positive rate will be high. • The distance measure highly affects the performance. Hence, choosing the correct distance measure becomes challenging in complex data.

Table 2.3: Advantages and disadvantages of Nearest Neighbour-based anomaly detection methods

2.4.3.1 Distance-based methods

These anomaly detection methods are based on the k-nearest neighbours algorithm. Anomalies are those points with large k-nearest neighbour distances. Below, two of the more popular distance-based methods are explained.

Mahalanobis Distance (MD) is a widely used method for anomaly detection. A general implementation of the MD algorithm can be seen in Figure 2.1. Basically, training dataset (normal data) is normalized and the mean and standard deviations are kept to transform test time observations. Subsequent to this, a covariance matrix is calculated to obtain the MD values. The

covariance matrix can be calculated using several methods such as inverse matrix method or Gram-Schmidt orthogonalization method. The matrix is also used at test time to calculate the MDs of test observations. As a final step, a decision rule that could be based on any distribution such as the Gamma, Weibull or Cox-Box transformation is used to determine the threshold at which a sample is considered anomalous or normal.

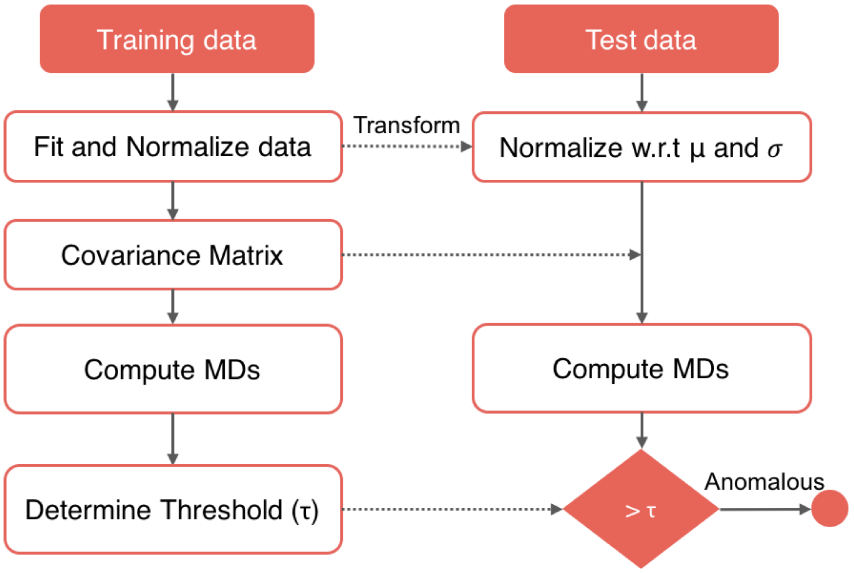

Figure 2.1: MD-based anomaly detection algorithm [partially adapted from [59]]

In their first study, Jin et al., [37] propose a health index using MD, to determine the health condition of cooling fan and induction motor from vibration signal. Features were extracted from the vibration signals using a Taguchi system. Anomaly detection is done by comparing the MDs under normal and abnormal conditions. Because MD is not normally distributed, Box-Cox transformation is used to achieve a Gaussian like distribution so the properties of the distribution could be used to define corresponding MDs with different health conditions. They used experimental data from cooling fans to validate their work and show that the early stages of faults can be detected successfully.

In another study, Jin et al., [38] suggested MD-based health index using vibration signals from healthy bearing and training of an autoregressive model.

Residuals from the monitored vibrations signals were calculated and wavelet features were derived from these residuals. Finally, MD from signal-based features was computed. The health indexes undergo a Gaussian transformation using the Box-Cox transformation so a threshold of 3 standard deviations away from the mean can bet set as an anomaly flag. When a monitored signal crosses the threshold the bearing wear life is started. The proposed method was tested on PRONOSTIA dataset [56] with simulated faulty and normal data.

A MD-based anomaly detection was used in [23] to detect early anomalous behaviour of Light-Emitting Diodes (LEDs). The MDs were calculated using LED operation data (e.g. temperature, input current and voltage).

Wang et al., [80] identified a set of features associated with potential failures of hard disk drives using Failure Modes, Mechanisms and Effects Analysis (FMMEA), then used minimum Redundance Maximum Relevance(mRMR) method to reduce the dimensionality of features and calculated MD values for determining the anomalies in the disk drive. In a different research they used the MD values and a Box-Cox transformation to determine the anomalies of hard disk drives [81].

The k**-nearest neighbour** is an unsupervised anomaly detection method that is mainly used for detecting global anomalies in a dataset. The way this algorithm works is that it first tries to find the k-nearest neighbour for every record in the dataset [65]. Then an anomaly score based on these nearest neighbours is calculated either by measuring the distance to the k^{th} nearest neighbour or averaging the distances to all of the k nearest neighbours. In practice the k-NN is usually used instead of the k^{th}-NN because setting the threshold is somewhat difficult to figure out [28]. Basically, k-NN assumes outliers are far from instances or put in another way, they have empty neighbourhood. The calculation method used in k-NN is as follows:

1. Find the k-nearest-neighbours of a sample point

2. Calculate an anomaly score using found neighbours either the distance

to the k^{th} nearest neighbour (single one) or the average distance to all of the k-nearest-neighbours.

The selection of parameter k plays a crucial role in the results. If it is given a very low value, the reliability of the density estimation for the instances may be dubious and the model may overfit. On the other hand, if it is a large value, density estimation may be too coarse and the model may underfit. An acceptable range for k is usually $10 < k < 50$.

2.4.3.2 Density-based methods

With density-based anomaly detection, the assumption is that normal data points are usually around a dense neighbourhood and abnormal data are scattered and in remote distance. To determine if a data point is anomalous, the number of data points within a local region is compared against the set threshold and if this number is below the threshold it is marked anomalous.

Local Outlier Factor (LOF) algorithm is the most popular density-based local anomaly detection method; it is the first to introduce the concept of local anomalies. LOF is based on the notion that outliers have low density with respect to their k neighbourhood [10].To calculate the anomaly score, LOF uses 3 steps:

1. Find the k-nearest neighbours (N_k) for each record.
2. Calculate the Local Reachability Density (LRD) for each record using the k-nearest neighbours.

$$LRD_k(x) = 1/\left(\frac{\sum_{o \in N_k(x)} d_k(x, o)}{|N_k(x)|}\right)$$

3. Compute the LOF score by comparing the LRD of a record to its k neighbours LRD value.

$$LOF(x) = \frac{\sum_{o \in N_k(x)} \frac{LRD_k(o)}{LRD_k(x)}}{|N_k(x)|}$$

In other words, the LOF is the ratio of local densities. So normal data will be assigned a score around 1.0 because its local density is as big as the densities of its neighbours, but anomalies get a higher score because they tend to have a low local density resulting in a larger score.

Like in k-NN, the value of k is crucial for this algorithm. Besides trying out different values for k, the authors of the algorithm suggest to use an ensemble strategy for computing the LOF.

Reference-based outlier detection - A bottleneck of distance and density-based outliers is that a nearest-neighbour search is required for each of the data points, resulting in a quadratic number of pairwise distance evaluations. Hence, a new method that uses the relative degree of density with respect to a fixed set of reference points for approximating the degree of density defined in terms of nearest neighbours of a data point is proposed by Pei et al., [62]. The outliers are ranked based on the outlier score assigned to each data point. The following steps are taken to determine outliers in a dataset:

1. For each reference point $p \in P$, sort the original dataset X in the one-dimensional space $X_p = d(x_i, p), 1 \leq i \leq n$, i.e., data points in X are ordered according to the distances to p.

2. For each data point $x \in X$, find the k reference-based nearest neighbours and compute the average neighbourhood density $D(x, k, p)$;

3. Set $D^P(x, k)$ of each point x to be the minimum of $D(x, k, p_r)$ w.r.t. P and compute the Reference-based Outlier Score (ROS):

$$ROS(x) = 1 - \frac{D^P(x, k)}{\max_{1 \leq i \leq n} D^P(x_i, k)}$$

The algorithm requires choosing a set of reference points that are not necessarily points from the dataset. The authors have tried a variety of different

shaped 2D synthetic data with small and big sample sizes and were able to detect anomalies that were challenging to find for the compared methods. Moreover, because the algorithm does not require calculating distances to every point, its time complexity is $O(Rnlogn)$, where R is the number of reference points and n is the dataset size. Hence, with regards to the above mentioned properties and empirical experiments performed by the authors, it is proposed that the algorithm is effective, efficient and very scalable in detecting outliers in large datasets.

2.4.3.3 Angle-based Outlier Detection (ABOD)

Existing approaches in anomaly detection are based on an assessment of distances (sometimes indirectly by assuming certain distributions) in the full-dimensional Euclidean data space. In high-dimensional data, these approaches are bound to deteriorate due to the notorious "curse of dimensionality". Hence, Kreigel et al., [44] propose ABOD.

The ABOD computes, for each point, the angles to all other pairs of points, and uses a weighted variance of these angles as the measure of outlierness [44]. Most purely distance-based methods are limited to applications with lower dimensions due to the "curse of dimensionality", however ABOD alleviates this limitation which makes it a useful in scenarios with higher dimensional feature space. In Figure 2.2 , the intuition behind the algorithm is shown. Point P is considered an outlier because the variance of the angles between each pair point is substantially smaller than that of the inlier points Q & R. Thus the datapoint with smaller variance of angle is considered an outlier. To increase the precision of the method the distance between the points is also considered so that nearby points which may have smaller variance in angle but smaller distance are not considered outliers. So to formulate this, the angle-based outlier factor is the variance over the angles between the difference vectors of $\overrightarrow{A}$ to all pairs of points in D with a weighted coefficient of distance of the points:

$$ABOF(\overrightarrow{A}) = VAR_{\overrightarrow{B},\overrightarrow{C}\epsilon D}\left(\frac{\langle \overline{AB}, \overline{AC} \rangle}{\left\|\overline{AB}\right\|^2 . \left\|\overline{AC}\right\|^2}\right)$$

where:

$$\text{dot product of AB: } \langle \overline{AB}, \overline{AC} \rangle$$

$$\text{distance between A and B, A and C: } \overline{AB}, \overline{AC}$$

$$\text{Cosine divided by distance: } \frac{\langle \overline{AB}, \overline{AC} \rangle}{\left\| \overline{AB} \right\|^2 . \left\| \overline{AC} \right\|^2}$$

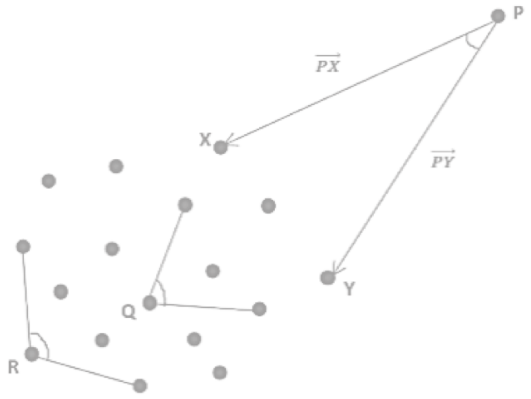

Figure 2.2: Angle-based outlier detection [adapted from [44]]

To calculate the angle-based outlier factor of an instance, the variance of all possible cosine distance is taken. The lower the variance the more outlier the point.

The fast angle-based outlier detection (FastABOD) was designed as a fast variant of ABOD, where only those pairs of points that are among the k-NNs are considered for calculating the angles (the strongest weights in the variance) and thus enhances the computation speed.

2.4.4 Classification-based methods

Classification is the task of classifying test samples using a learned classifier by training on labeled dataset. Based on the number of labels in training, the learnt model labels test samples into two or more classes. Anomaly detection based on a classifier comprises of two steps [13]: The first step, which is the training phase, a classifier is learned using available labeled training data. In a second step, the test instances are classified as normal or abnormal using

the classifier trained in initial step. The general assumption made by classifier based detectors is the ability of learning and differentiating the normal and abnormal classes from the given feature space. In general, classification-based anomaly detection can be divided into one-class (normal labels only) and multi-class (multiple classes) classification depending on the availability of labels. Below the One-Class Support Vector Machine (OCSVM) and the neural network method are explained. Table 2.4 lists the advantages and disadvantages of classification-based methods.

Advantages	**Disadvantages**
• Classification-based methods, particularly multi-class methods, can be used to distinguish between samples belonging to different normal classes. This can be useful with systems that have multiple normal operating conditions. • Test time computation is very fast due to the pre-trained model.	• Multi-class methods require accurate labeling of normal classes, which are often hard to acquire. • Some classification-based methods assign a class to samples. This may be a disadvantage as it does not provide a tangible anomaly score.

Table 2.4: Advantages and disadvantages of classification based anomaly detection methods

2.4.4.1 One-Class Support Vector Machine (OCSVM)

The OCSVM is a one-class classifier where a model is trained on a single class dataset and then the SVM classifies a sample test data to either belonging to a learned class or not. In a case of anomaly detection, OCSVM is trained on the normal dataset and then each test record is given a score by a normalized distance to the determined decision boundary [49]. The general idea is that the anomalies contribute less to this decision boundary compared to the normal instances. A number of parameters need to be configured, e.g.,:

- Kernel type
- ν: Represents the lower bound on the number of support vectors and the upper bound on the number of outliers

- ϵ: Tolerance of termination criterion

Different variations of SVMs e.g. one-class SVM and least squares SVM have been used in [19][84] for anomaly detection in spacecraft and aviation and electronics systems.

2.4.4.2 Neural network methods

Neural networks have been applied to anomaly detection methods in both one-class and multi-class configurations. At its simplest form, a multi-class anomaly detection first trains a neural networks on the normal training dataset to learn the different normal classes. In a second step, the test samples are fed to the network. If the network recognizes the input then it is considered as normal, otherwise the sample is rejected and is identified as anomaly [13]. Variations of the basic neural network technique has been proposed that use a slightly different network structure. Replicator neural networks have been used for one-class anomaly detection [30] [83]. Basically, a multi layer neural network with the same number of input and output neurons that conform to the dimensionality of the input is constructed. The model is trained using three hidden layers to compress the data. At test time, the data is reconstructed using the compressed representation. The error between the original data and the reconstructed output is taken as an anomaly score. The auto-associative neural network also known as auto-encoder was used by Diaz and Hollmen [21], for detecting outliers in vibration and current data in a mechanical asynchronous motor.

2.4.5 Spectral/Subspace-based methods

Spectral or subspace-based methods try to extract features that best describe the variability of the training data [13]. These methods assume that the normal data can be presented in a lower dimension subspace where normal data is distinguished from abnormal data. Principle component analysis is considered as a subspace-based approach to anomaly detection.

2.4.5.1 Principle Component Analysis (PCA)

PCA reveals inner structure of the data and explains variance in the data. It looks for correlation between features and determines the combination of values that best captures the differences in outcomes. This combination of values is then reduced into a more compact feature space called the principle components. For the task of anomaly detection, for each new input, its projection on the eigenvectors is calculated along with a normalized reconstruction error. The normalized score is used as anomaly score. The higher the score the more anomalous the sample is [70].

Plante et al., [64] applied PCA for identifying healthy, unbalanced and misalignments in motors. PCA is used as a multi-class classifier to group known FFT vibration patterns based on their associated trends. When unseen and unknown patterns are given to the model, it may be classified as anomaly. The dataset used in the study are vibration signals from a laboratory machinery fault simulator. Data was recorded using four accelerometers (two per two bearings). The vibration signals are processed using Fast Fourier Transform (FFT) and frequency domain features are extracted. The experiment is designed to determine severity trend for each of the fault types (healthy, unbalanced and misalignment in motors). The score and loading plots for all four PCA per accelerometer is computed and plotted. Each point on the score plot represents a single frequency value and on the loading plot, the points show one of previously named conditions of the equipment. The loading plot showed a clear separation of the conditions. Unknown instances were seen to belong to the correct healthy or failure classes. From the load plot, the data points that were further away from the healthy class could be assumed to having a more severe fault. The authors were able to show that PCA provided a good separation of clusters, however, capturing the fault severity trends still remain a challenge.

Stellman et al. [72] used PCA for spectroscopic data to monitor the condition of a lubricant in helicopter rotary gearboxes. PCA was used in a work by Allgood and Upadhyaya [3], where certain descriptive statistics were given

to PCA for DC motor diagnostics and prognostics. Qingbo et al., [31] also used PCA on statistical features of frequency-domain signals from an internal-combustion engine sound and automobile gearbox vibration analysis. The most representative principle components of the statistical features were then used to classify machine fault patterns. Some systems have a non-linear behaviour and so PCA may not be a suitable technique.

2.4.6 Space-subsampling methods

2.4.6.1 T*-framework

Foss et al., [24] propose a new approach to outlier detection in an arbitrary number of dimensions, based on rankings obtained by investigating low-dimensional subspaces. The proposed algorithm for subspaces of fixed low dimensionality $1 << k << d$ and accumulate the outlier scores over all k-dimensional spaces that result in an outlier ranking for the original d-dimension. The T*-framework is used to divide the d-dimensional space into k-dimensions. Intuitively, the algorithm combines the outlier scores and how frequent a point is considered in lower dimensions. The top ranked points are considered outliers. The steps involved in the T*-framework is as follows:

1. Compute outlier score for each sample in D:

   - Set parameter $k << d$
   - Let $S_1 \subseteq \{1, ..., d\}, ..., S_z \subseteq \{1, ..., d\}$ be all subsets of $\{1, ..., d\}$ of size k for all $i \in \{1, ...z\}$
   - For every point $x \in D$ compute an outlier score
     $score(x) = \sum_{i=1}^{z} outlier(x, D_{S_i})$
     where $outlier(.)$ can be any outlier detection method.

2. Rank the points in D with respect to their outlier scores:

   - Sort the outliers according in ascending order
   - Return the top N queried outliers

The authors have conducted several experiments and conclude that a value of $k = 2$ is sufficient to outperform the standard outlier detection methods while not compromising for efficiency.

2.4.6.2 Isolation forest outlier detection

Isolation forest [47] is an efficient way of performing outlier detection in high-dimensional datasets. The algorithm isolates data points by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Because a tree structure can be served as recursive partitioning, the number of splitting could be considered as the degree of normality or abnormality. Data instances with shorter paths in partitioning are considered more anomalous. Therefore, when a an ensemble of random trees collectively produce shorter path lengths for an observation, they are very likely to be anomalies. Another distinction of isolation forest compared to the previously mentioned algorithms is that it does not utilize distance or density measure as a means of detecting anomalies and so this reduces computation time in comparison to the distance and density-based methods. According to Liu et al., [47], the algorithm has a linear time complexity with very low memory requirement and is very scalable in case of high dimensional data.

2.4.7 Deep learning-based methods

Deep learning is a descendant of machine learning that tries to model high level representations hidden in a dataset and classify or predict patterns by stacking multiple layers of neurons or processing modules in a hierarchical structure. In recent years, deep learning has found its way in many areas and has been successfully applied to tasks such as computer vision, speech recognition, natural language processing, bioinformatics and fraud detection [90]. Some of the factors that has made deep learning a popular approach for many applications include, increase in computing power, availability of big data [40], and the fast growing research in neural network methods. With regards to machine health monitoring systems, considering big machinery data

collected from sensors and the complexity in the data, deep learning and its power in learning hidden representation from such data seems like the right fit. The conventional data-driven methods for machine health monitoring follow typical steps: manual feature engineering, extracting and selecting the right features, and model training. Designing and selecting good features manually could be very challenging and in most cases, it requires expert knowledge and a great amount of tuning. However, using deep learning, the above three phases, feature engineering, selection and extraction and model training can be jointly performed and optimized in an end-to-end architecture. The intuition behind this is the use of many single layer operations that can be regarded as a non-linear transformation from input to output. Each layer learns a new representation of the input data and then by stacking multiple layers, complex patterns are learned. This eliminates expert knowledge and designing hand crafted features, thus the models can be applied to machine health monitoring in a very general way [90].

Deep learning methods come with advantages and disadvantages. A list of some of these strength and weaknesses is given in Table 2.5.

Advantages	**Disadvantages**
• Can bypass manual feature engineering by extracting abstract features automatically.	• Require many samples to train the model in order to produce great results.
• They can be used as an end-to-end model. Given raw input and depending on the last layer, classification or regression analysis is obtained.	• No standard architecture.
	• Many parameters need to be tuned.
	• Long training time depending on the number of parameters.
• Model is very fast at test time because of pre-training.	• Features are abstract and hard to infer.

Table 2.5: Advantages and disadvantages of deep learning-based anomaly detection methods

Deep learning models have many variations: Auto-encoders [60], Deep belief network [61], Deep Boltzmann machines [62], Convolutional neural networks [63], and Recurrent neural networks [64]. In the next sections we will review some of the most commonly used deep learning anomaly detection

methods.

2.4.7.1 Auto-encoder

Auto-encoder is basically a neural network for which the input is the same as the output. It compresses the input into a latent-space representation known as the bottleneck layer and then uses this layer to reconstruct the input (Figure 2.3).

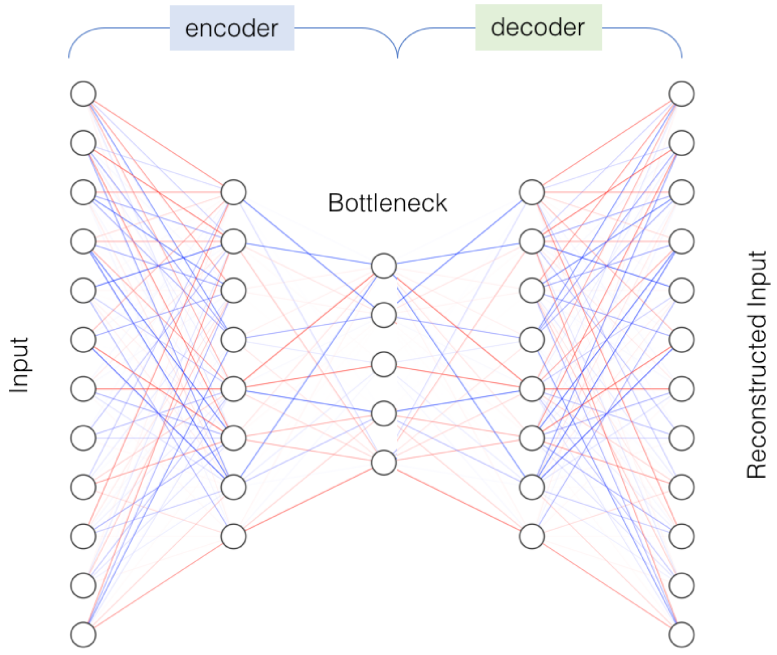

Figure 2.3: An auto-encoder with 3 hidden layers

A simple feedforward auto-encoder consists of an input layer, one or more hidden layer(s) and an output layer that has the same amount of neurons as the input layer. The auto-encoder includes an encoder and a decoder and steps involved in training a single hidden layer auto-encoder as an anomaly detector is as follows:

1. Given the input in the form of $\mathbf{x}\epsilon\mathbb{R}^d$
2. Create a mapping or a latent space representation (encoder): $\mathbf{z}\epsilon\mathbb{R}^k$ using a deterministic application: $\mathbf{z} = \sigma(\mathrm{W}x + \mathrm{b})$

Where σ is an element-wise activation function and popular choices include the *softmax*, *tanh* and *ReLu*. z is called the (bottleneck) latent space representation. W is the weight matrix, and b is the bias vector.

3. Decode the encoded representation to reconstruct ẋ the original input using:

$$\acute{x} = \acute{\sigma}(\acute{\mathrm{W}}\mathrm{z} + \mathrm{b})$$

4. Train the encoder, decoder using backpropagation to minimize the reconstruction error:

$$\mathcal{L}(\mathrm{x}, \acute{\mathrm{x}}) = \|\mathrm{x} - \acute{\mathrm{x}}\|^2$$

5. Use the mean squared error between the original data and the reconstructed output as anomaly score

A deep auto-encoder is achieved by stacking layers to form a deeper structure. By reducing the number of units in the hidden layer, we expect features that better represent the data will be extracted. Additionally, by stacking layers we are able to apply dimensionality reduction in a hierarchical manner, which results in a more abstract feature in the deeper hidden layers and ultimately better reconstruction of the data.

A number of hyper-parameters can be tuned to design the best architecture for the specific task in hand e.g., number of neurons per layer, number of hidden layers, use of regularization per layer, activation function, optimizer, and batch size just to mention a few.

An anomaly detection method using extreme learning machines was developed by Janakiraman and Nielson [34] to detect anomalies in aviation data.

In a research by Miranda et al., [52] an auto-associative neural network is designed to diagnose incipient faults in power transformers based on the results of dissolved gas analysis. The authors mention that the presence of dissolved gases in the oil from a power transformer is a good indicator that

could be used to monitor the condition of the equipment. There are six fault cases in total and a typical normal profile data is also available. A total of 7 auto-encoders is trained and linked in a competitive parallel arrangement. At test time, the unseen sample is given to every auto-encoder model and the output reconstruction error is recorded. The auto-encoder with the smallest value of reconstruction error reveals the class the sample belongs to. The results give a 100% success rate for the purpose of this research study.

2.4.7.2 Denoising auto-encoder

A denoising auto-encoder is a type of auto-encoder that is robust to noise. Basically, instead of directly giving the input data to the encoder, noise is added to the input and fed into the encoder. The denoising auto-encoder tries to learn a latent space at the bottleneck layer which is robust to noise. During training, the network computes a loss between the noisy output of the decoder and the ground truth (original non-noisy data) and tries to minimize the loss between the reconstruction and the original sample.

Yan and Yu [86] used a stacked denoising auto-encoder for extracting features for gas turbine combustors. An extreme learning machine was then used with the extracted features for anomaly detection.

2.4.7.3 Convolutional auto-encoder

Convolutional Neural Networks (CNNs) have been used extensively in image processing where inputs are mostly 2D data. CNN tries to learn abstract features and identify patterns within a dataset. As the number of layers increase, more complex patterns are learned. Kernels in convolutional layers, convolve with multiple local filters and generate invariant local features. Usually convolutional layers are followed by pooling layers that extract the most significant features with a fixed length over a sliding windows of raw data.

A 1D-CNN is very effective when we are interested in features from shorter fixed-length segments of the overall dataset and where the location of the feature within a segment is not of high relevance. Hence, this makes 1D-CNN very suitable in analyzing time sequences or any fixed-length data such

as sensor data. Any CNN architecture whether 1D, 2D or 3D, share the same concept and use the same approach, the only difference is the shape or dimensionality of the data that is fed into the network and how the filter slides over the data.

A 1D CNN can be formulated as follows:

Input Assuming the input data is sequential $\mathbf{x} = [\mathbf{x_1}, ..., \mathbf{x_T}]$ where T is the length of the sequence and $\mathbf{x}_i \in \mathbb{R}^d$ at each time step.

Convolution This layer defines a filter or the feature detector. The filter size can be seen as the required number of features a network needs to learn in a specific layer. A filter vector (or kernel) $\mathbf{u_l^j}$ is multiplied by an input vector and the result of this dot product is the convolution operation formulated as follows:

$$c_j^{l+1}(i) = \varphi(\mathbf{u_l^j} * \mathbf{x^l(i)} + b_j^l) \tag{2.1}$$

where $\mathbf{u}_j^l$ is the filter vector, b_j^l is the bias term and φ is the non-linear activation function. The $\mathbf{x^l(i)}$ is the i^{th} subsequence in layer l. The output of the equation 2.1 is a feature vector which is the result of convolving the filter vector $\mathbf{u}$ with the defined input vector through sliding the filter from start to end of the input sequence. The number of resulting feature vectors is dependent to the number of filters defined.

Max-pooling In order to minimize the number of parameters for the model, a pooling layer is usually defined after the convolutional layer. The pooling layer reduces the length of the feature map through its parameter - pool size. The *max* operation takes the maximum over the pool size values from the feature map c.

By adding alternating convolutional and pooling layer the network learns more complex features, which ultimately represents the latent space and the encoded representation. This compressed latent representation is then used in subsequent deconvolution and upsampling layers as a decoder to reconstruct

the original input. Just like the original auto-encoder, the mean squared error is used as the loss function and an optimizer is used to minimize the error between the original input (data sequence) and the reconstructed output. The higher the error the more anomalous the sequence.

In their article, Janssens et al., [35] propose a feature learning model for condition monitoring based on convolutional neural networks. The model is given the raw amplitudes of the frequency spectrum of the vibration data. Two accelerometers are placed perpendicular to one another. The model comprises of a convolutional layer with width 64 and height of 2 (corresponding to the signals from the accelerometers), followed by a fully-connected layer with 200 units. The results show that the automated feature engineering outperforms (93.61%) the classical manual feature engineering method and a random forest classifier (87.25%).

In a work by Ince et al., [32] a simplified 1D CNN architecture is used. The authors claim that using a 1D CNN can efficiently compute hundreds of back-propagation iterations. The 1D architecture can merge feature extraction and classification into a single model and the computational complexity of the method is drastically lower which makes it a suitable method for real-time detection of faults. The model is trained offline in a supervised fashion and subsequently used on current signals coming directly from a motor.

2.5 Conclusion

In this chapter we reviewed a variety of anomaly detection methods. It is observed that no single anomaly detection method is necessarily the right fit for a specific domain. Hence, in a real world setting, a number of different methods should be applied and tested. Furthermore, depending on the application, one can select algorithms based on the robustness it provides. Some algorithms require parameter tuning (e.g., LOF, k-NN, ROF, ABOD), some are very fast but not as accurate (e.g, HBOS), others require a big amount of data to train and need more computation resources (e.g. Neural network). Based on the review of literature and the observations made, we have decided to

try a number of above mentioned algorithms for our robot arm experiment. The following section describes the robot arm platform followed by Chapter 4, where results of using aforementioned algorithms on the robot arm data is presented.

Chapter 3

Experimental Setup

A belt-driven, reconfigurable robot manipulator has been customized for this investigation. A single Degree of Freedom (DOF) mechanical system was developed to simulate faults that can occur during regular operation of machinery. The robot arm system was designed to assess the effect of mass, friction and belt tension on motion. Multiple reproducible experiments could repeatedly be run on it with similar results.

3.1 Robot arm development

The arm was initially built with five DOF. A general schematic of the earlier proposed platform is presented in Figure 3.1 . However, we were interested in detecting faults and assessing the techniques incrementally on a simpler machine and gradually move onto a more complex system. Hence the arm was dismantled and setup with a single DOF shown in Figure 3.2. The robot arm is designed to move in a back and forth manner and following a predefined trajectory. Even though this seems like a simple machine, there exists many real life examples such as opening and closing gates in elevator doors that require fault monitoring due to constant use and importance of uptime for providing service. Thus, we can translate the arm's back and forth movement to a door opening and closing mechanism.

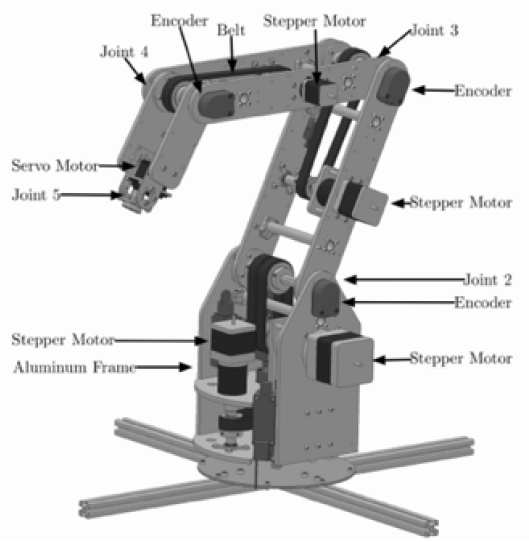

Figure 3.1: Initially proposed 5 DOF robot arm

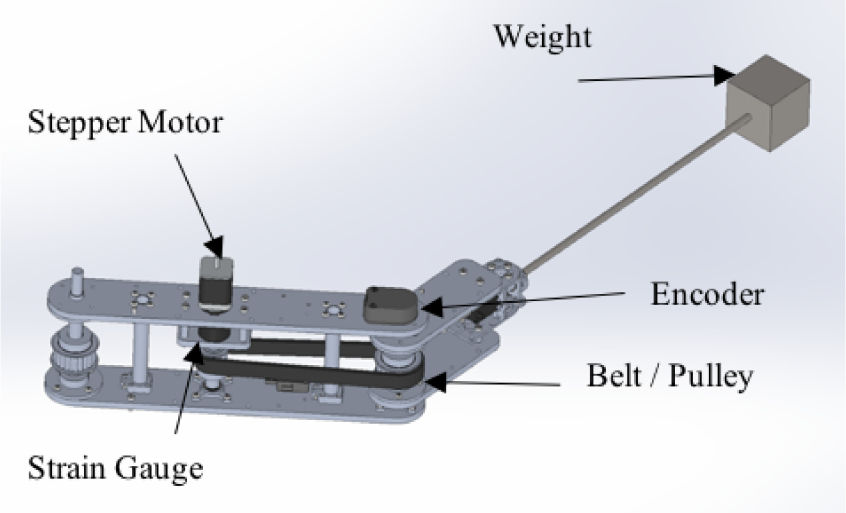

Figure 3.2: Final single DOF robot arm

3.2 Robot arm components

The robot arm consists of many components that help with its operation, data acquisition and reproducibility.

Microcontroller : The system is controlled by a programmed teensy 3.2 microcontroller to actuate the motor and read the measurements from the instrumentation.

Stepper motor : The robot arm uses a brushless stepper motor to drive the

belt through a programmed range of motion (typically between 19 and 126 degrees). The motor is controlled by a Leadshine DM542 Microstep Drive.

Adjustable tensioner : The robot arm was fitted with an adjustable tensioning device mounted with an idler pulley which allows the tension in the belt to be modified for various experiments. The tensioner has four bolts that fasten it to the main body of the arm and the heights of the bolts can be changed to adjust the height of the idler pulley, increasing or decreasing the tension in the belt.

Initialization switch : There is a switch attached to the robot arm that is activated by having a screw push it on the arm's first cycle. The switch tells the microcontroller that it has reached the starting point and tells the motor to begin the looping script that causes the robot arm to oscillate. See the operation procedure below for more information.

Belt-pulley system : The robot arm transmits power from the motor to the arm link via a timing belt and pulley system. The pulley on the shaft connected to the motor and the pulley on the adjustable tensioner are 5GT 16-tooth pulleys and the pulley on the joint is a 5GT 24-tooth pulley.

Encoders : The robot arm is fitted with two US Digital E2 optical encoders to measure the angles of the arm as it moves through its range of motion.

Strain gauges : There are two full-bridge strain gauge configurations installed on the arm. One set is located on the shaft connected to the motor to measure the instantaneous torque outputs of the motor as the arm moves through its range of motion. The second configuration is located on a piece of aluminum that is connected to the belt. These strain gauges are used to measure the instantaneous tension in the belt, which is useful for resetting the belt tension back to its normal operating value. The strain gauges are each connected to individually calibrated amplifiers and are then read by the micro-controller.

Thermo couple : Thermo couple is installed on the robot arm to measure the ambient temperature while the robot arm is in operation. By measuring temperature, we are able to simulate temperature related faults and provide a means of reproducing the condition at normal or faulty operation.

Weights : Weights are used on the robot arm to increase the overall inertia of the system to decrease unnecessary vibrations and oscillations of the arm. The weights are able to be added or removed as necessary. For most experiments, 5.5 kg weights are placed on the arm.

Steel plate : A steel plate is used as the track for the robot arm. The wheel that the arm rolls on is made of aluminum and the steel plate minimizes the depth of the scratches made by the wheel as the robot arm moves through its range of motion.

Seismic mass : The robot arm is fastened to a large 3-ton platform to minimize any external vibrations made by walking around the system or momentum that is created when the robot arm moves.

A picture of the complete robot arm platform with labeled components is demonstrated in Figure 3.3.

3.3 Operation of the robot arm

The robot arm operates as follows:

1. The Teensy Micro-controller is plugged into a computer via USB to provide it with power and it begins to output sensor data.

2. When the Teensy is first plugged in, the arm moves forward until a protruding screw hits the initialization switch.

3. When the initialization switch is activated, the microcontroller tells the motor to rotate backwards for 5 seconds.

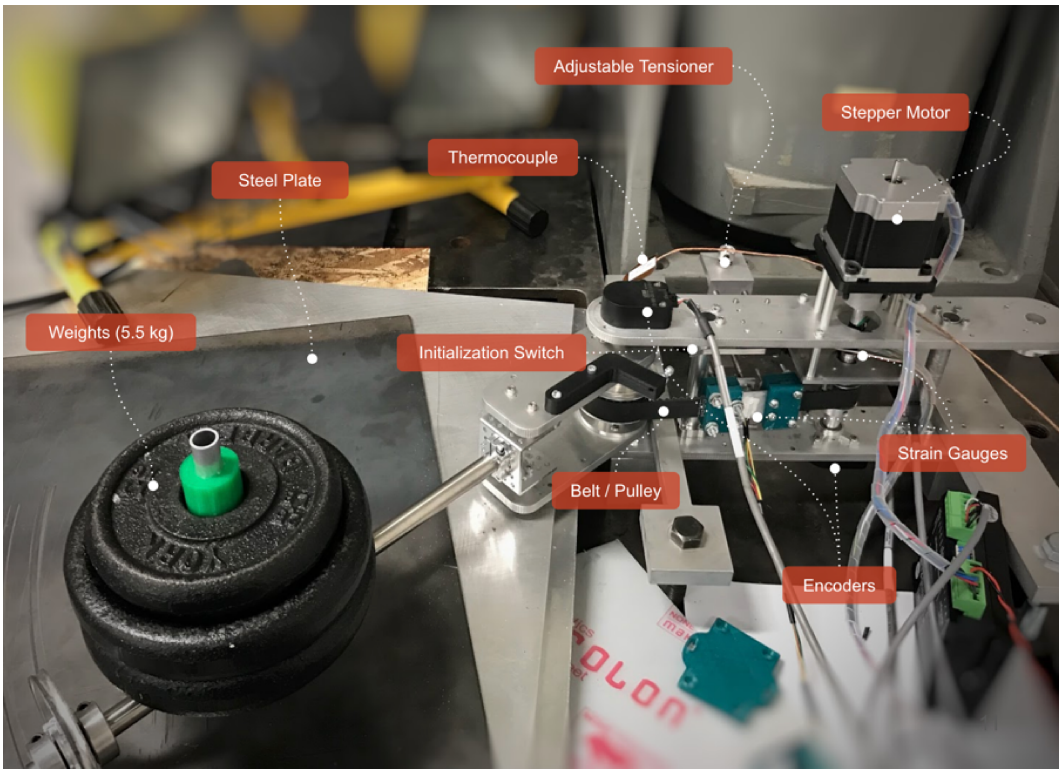

Figure 3.3: Complete robot arm platform with labeled components

4. After 5 seconds, the arm stops for 0.1 seconds and then moves forward another 5 seconds.

5. The switch is not pressed again. The robot arm stops just short of the switch, stops for 0.1 seconds, and once again moves backwards for 5 seconds.

6. This cycle will then repeat for a programmed number of cycles.

3.3.1 Data acquisition

The robot arm is armed with four sensors: two full-bridge strain gauge configurations and two encoders. For the purpose of these experiments, only the encoder on the motor output shaft is used. The data acquisition system outputs 7 values through the USB ports connected to a laptop. A python script shown in Figure 3.4 reads the outputs and saves it into a CSV file format. These measurements are:

Timestamp This value counts which sample the dataset is on. The sampling rate for these experiments is 100 hz.

Cycle number The cycle number counts which cycle the robot arm is at. One cycle is defined as one pass forward and one pass backward. It takes approximately 10.2 seconds per cycle.

Cycle mode The cycle mode is outputted as either a "0" or a "1". These indicate whether the robot arm is moving forward or backwards.

Motor shaft angle (°) The angle of the motor shaft that is measured by the encoder is outputted. After pressing the initialization switch, normal range of motion of the robot arm moves between 19 and 126 degrees.

Motor output torque (Nm) The output torque of the motor is measured via the calibrated strain gauge configuration on the shaft.

Belt tension (N) The tension of the belt is measured via a calibrated strain gauge that is installed on a piece of aluminum that holds the belt together.

Temperature (°C) The ambient temperature of the robot arm measured via the thermo couple installed over the robot arm.

```python
import serial
import time

last_temp_message = "0"

filename = input('Enter a filename: ') or 'data.csv'
log_file = open(filename,"w",newline='\n')
# COM4: Temperature
temp_ser = serial.Serial("COM4")
# COM5: Torque, etc.
ard_ser = serial.Serial("COM5")
print(ard_ser.name)
ard_ser.reset_input_buffer()
ard_ser.readline()

count = 0

while True:
    count = count + 1
    if temp_ser.in_waiting:
        last_temp_message =  temp_ser.readline().decode("utf-8").rstrip()
        print(last_temp_message)

    sample = ard_ser.readline().decode("utf-8").rstrip()
    print(sample + ',' + last_temp_message)

    log_file.write(str(time.time()) + ',' + sample + ',' + last_temp_message + '\r\n')

    if count == 1000 :
        count = 0
        log_file.flush()

log_file.close()
```

Figure 3.4: Python script used to read and save outputs from the data acquisition system.

3.3.2 Sample signal output

A single sample of torque signal is shown in Figure 3.5 below. The waveform represents a single cycle. A cycle is one round of arm going forward and backward which can be seen as door opening and closing.

To demonstrate the stability of the system, normal samples are aggregated and overlaid on top of each other. As shown in Figure 3.6, the signals are following a consistent pattern. However, at some points in time, there are signals that don't follow the normal pattern and show some deviations which could be because of bad sensor reading or a sudden change in the behaviour of the robot arm due to random condition.

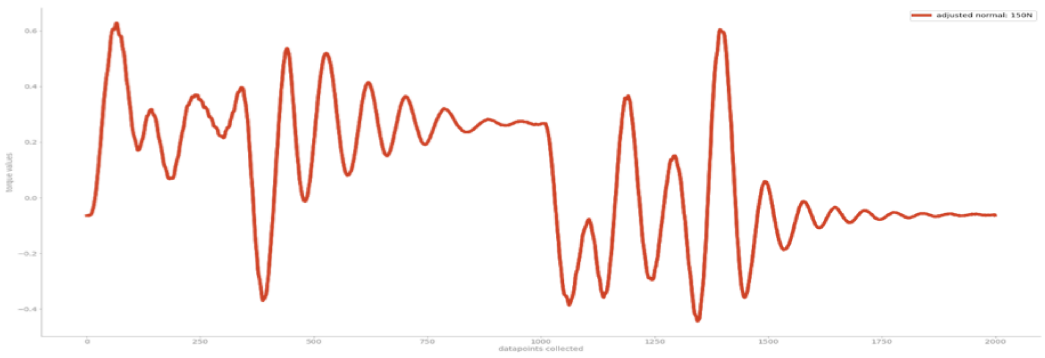

Figure 3.5: A normal torque signal sample

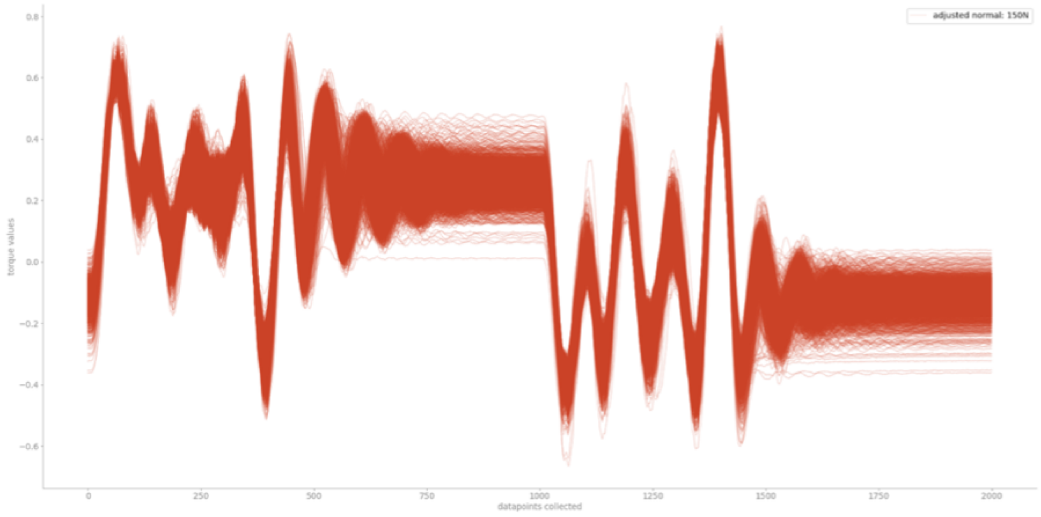

Figure 3.6: Normal samples overlaid, showing consistent pattern with some abrupt changes

3.3.3 Failure types and data collection

The failure types for the robot arm were generated having in mind the types of failures that could occur in a real world setting with systems that have doors opening and closing. These failures mostly have direct effects on the belt tension. In order to simulate faults that are reproducible, some additional sensors have been installed on the robot arm. For example, the strain gauge installed on the belt measures the tension and the thermocouple measures the ambient temperature around the robot arm (see Figure 3.7(a) and (b)).

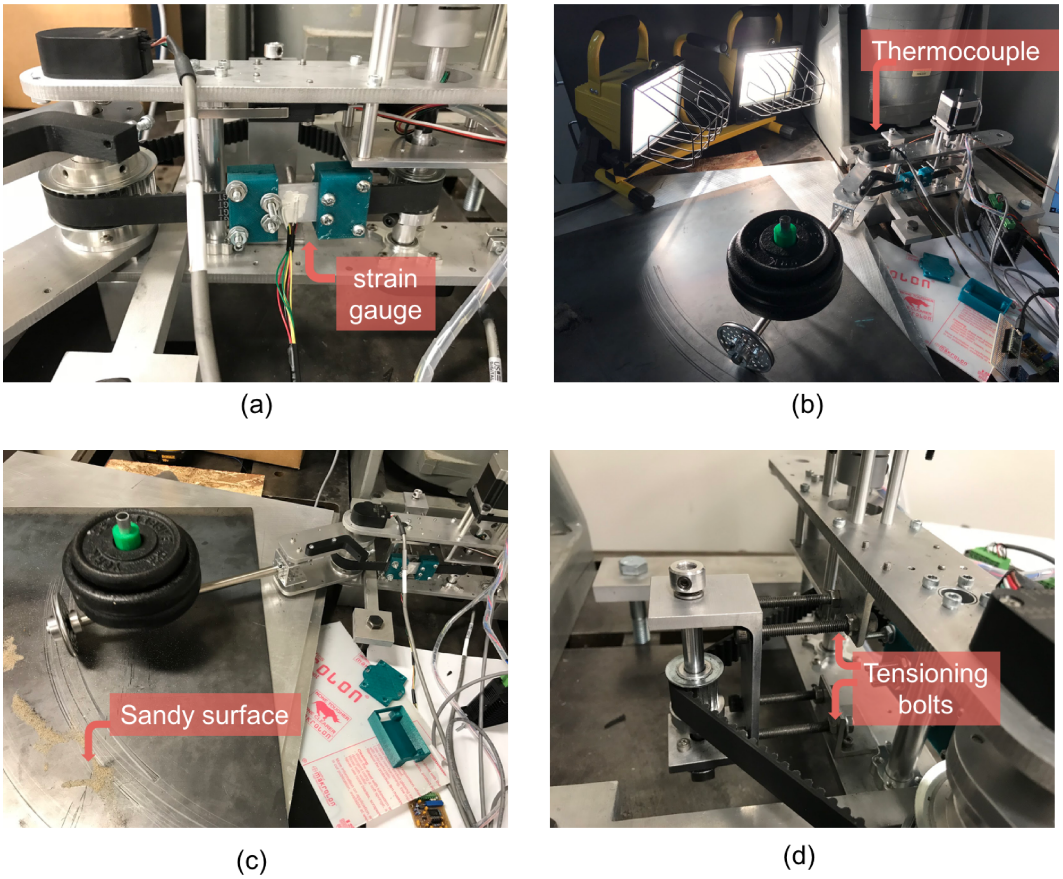

Figure 3.7: Sensors installed on the belt and the arm and simulated faults. (a) Strain gauge for measuring the tension on the belt. (b) Thermocouple for measuring ambient temperature around robot arm with halogen bulbs for generating heat to see the effect of temperature on the belt tension. (c) Friction using sandy surface. (d) Loose and tight belt tension faults using loosening and tightening of bolts and adjusting the height of idler pulley.

In total we simulated five different fault modes including:

- Loose belt: Two levels of looseness. This was achieved by loosening the bolts on the tensioner and adjusting the height of idler pulley. Loose level 1 (l1) at 120N and level 2 (l2) at 100N (see Figure 3.7(d)).
- Tight belt: Same procedure as loose belt, but this time the belt was tensioned at 168N.
- Friction: Friction was simulated using sand scattered on the metal surface (see Figure 3.7(c)).
- Change in temperature: Halogen bulbs were used to generate heat around the robot arm (see Figure 3.7(b)). The temperature around the arm was increased to approximately 40 °C.

Table 3.8 summarizes the data collected from the robot arm platform. To have a better understanding of the tensions, Figure 3.9 shows the trend of tension over time for the normal state and the interval at which each fault falls considering the tension value. Moreover, to show how close and subtle the faults are in relation with the normal profile, overlaid torque signals for normal and fault conditions is shown in Figure 3.9.

	TEMP	TENSION	NO. SAMPLES
Normal	~23.5	~150	7193 (22hrs)
Loose l1	~23.5	~120	182
Loose l2	~23.5	~99-100	180
Tight	~23.5	~168	194
High Temperature	~40-42	~166-169	210
Sand (friction)	~23.5	~150	183

Figure 3.8: Summary of data collected from the robot arm platform

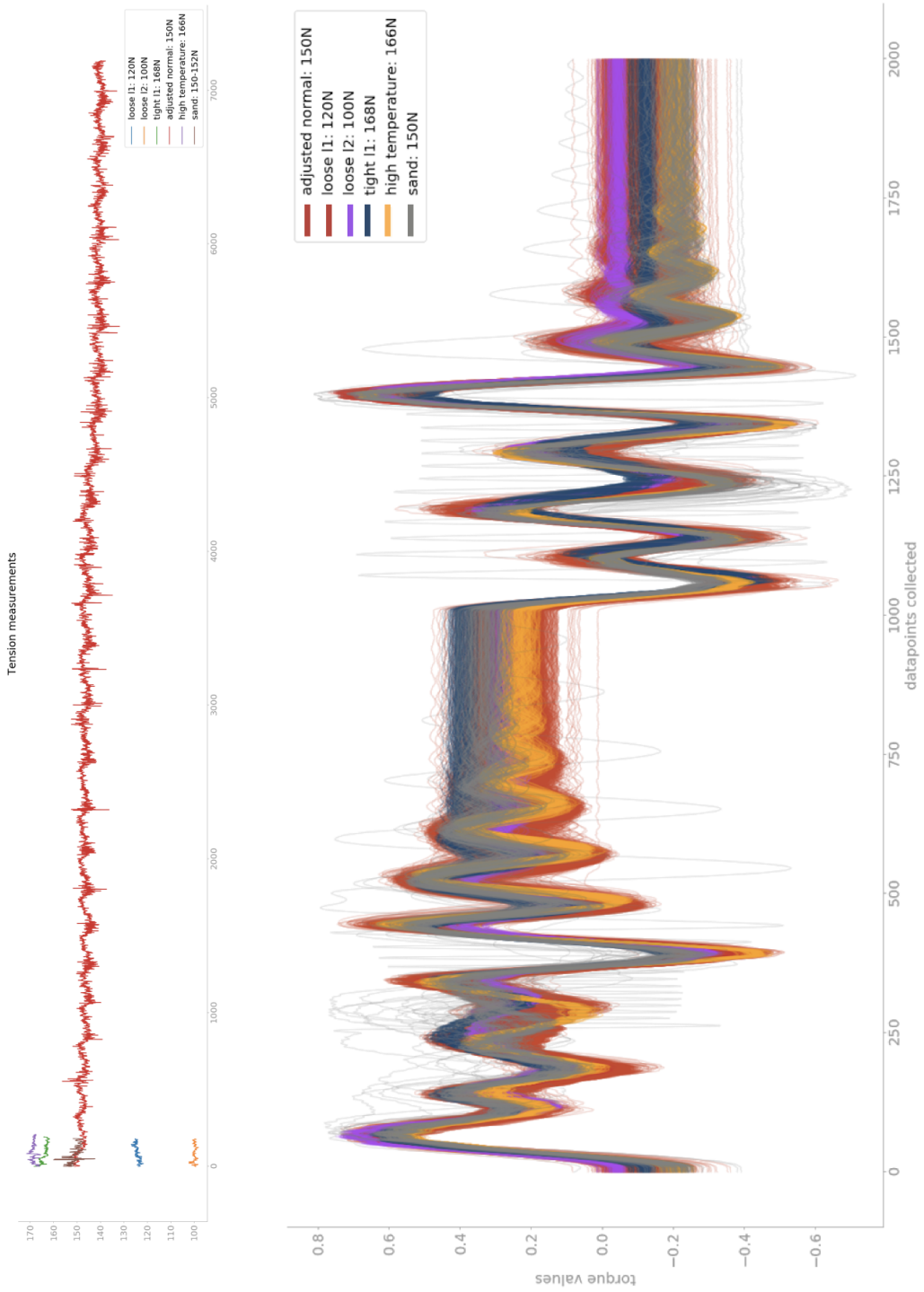

Figure 3.9: **Top**: Normal vs faulty tension signals demonstrated through time —**Bottom**: Overlaid torque signals for normal and faulty data

Chapter 4

Experiments and Results

Reviewing the literature on the applications of anomaly detection techniques for prognostics and health management reveal that, there exist variety of anomaly detection methods with their own advantages and disadvantages and that there is no *one technique fits all* situation. An anomaly detection method may perform well on a particular dataset with specific properties, while the same method applied to a different scenario may perform very poorly. Hence, trying out a number of different detection methods for a particular case study and evaluating their performance using a consistent metric can provide a fair assessment for selecting the most fit algorithm(s). In this section, we will describe the necessary steps required to perform such a selection on the robot arm data and elaborate the process in detail.

4.1 Data pre-processing

As for all data-driven techniques, "garbage in, garbage out" also apply for data-driven PHM methods. In most cases, real-world data contain many errors, are inconsistent or incomplete. Therefore, some kind of data pre-processing is necessary for resolving such issues and preparing the data for anomaly detection. In PHM, data pre-processing usually includes the following steps: data cleansing, normalization and feature engineering (feature extraction, feature selection and feature learning).

4.1.1 Data cleaning

In the case of robot arm data, the torque data is collected through a data acquisition system which is guaranteed to output reliable measurements in a structured format. The torque is sampled at 200Hz and each observation should contain 2000 sampled data points per cycle. However, at some points inconsistent number of samples are collected. Using a script, such discrepancies are identified and removed as a first step to the data pre-processing pipeline. As a result of data cleaning, the samples collected are saved in a tabular CSV file with each row representing a single observation with 2000 sampled data points. The data is now ready for further processing.

4.1.2 Data normalization

Data normalization or scaling is a preprocessing method that is usually employed before feature selection and classification. A complex PHM system may be fed very high dimensional data. If each dimension is not normalized to a similar scale, the output of the machine learning algorithms may be biased towards some of larger scaled features in the dataset.

A majority of the outlier detection algorithms use distance (Euclidean) as a measure of similarity and assume normality of input data. Some examples of algorithms where feature scaling matters include k-nearest neighbours with an Euclidean distance measure, principal component analysis extracts features with maximum variance, hence variance is higher for high magnitude features which skews PCA towards the high magnitude features, support vector machines, neural networks or any algorithm that uses gradient descent/ascent-based optimization, scaling helps in speeding up the process and performing the gradient quickly on smaller ranges.

Hence feature normalization is required to approximately level the ranges of the features and enforce approximately the same effect in the computation of similarity [68]. The choice of appropriate normalization technique and the normalization range is an important consideration since applying the wrong method could change the structure of data and affect the outcome of an anal-

ysis. There is no universally accepted rule for normalizing datasets and so the choice of normalization method is generally left to the delicacy of the user [58]. There are many techniques for normalization. Two popular techniques widely used in different application fields are Min-Max and z-score normalization. The Min-Max normalization scales the values of a feature X in a dataset using its minimum and maximum values. The Min-Max scaler is able to convert a value x of the feature X to $\hat{x}$ in the range[low, high]. The formula for calculating the Min-Max value for a feature is as follows:

$$\hat{x} = low + \frac{(high - low)(x - X_{min})}{X_{max} - X_{min}}$$

A different approach to normalization is the z-score standardization. Applying the z-score standardization rescales the features to have the properties of a standard normal distribution with mean $\mu = 0$ and a standard deviation $\sigma = 1$. The z-score is computed as follows:

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

For the purpose of the robot arm data the Min-Max scaler and the z-score standardization from the popular python library scikit-learn [61] was used to transform the training data. The transformation is subsequently applied to the test dataset at the time of applying the detectors on unseen test dataset.

4.1.3 Feature engineering

Feature engineering defines the process of creating features for machine learning algorithms using the domain knowledge around the dataset. Feature engineering is a critical step in machine learning and is both difficult and expensive [60]. In general, feature engineering consists of feature construction, feature selection & extraction and feature learning.

As mentioned previously, the dataset used in this study is comprised of sampled torque data in the form of univariate time series. In the following sections, we explain the process of feature construction and selection of the

robot arm time series data and will come back to the topic of feature learning in the following sections when deep learning methods are applied.

Feature-based representation of time series

Feature-based representation of time series have been used across many disciplines and are usually applied in cases where longer time series corresponding to streams of data are collected. e.g. medical or speech recordings [25]. A very simple example of representing a time series is using its mean and variance characteristics and transforming any time series object into a more compact vector that encapsulate these two properties. Because our study can be seen as a classification task, i.e., classifying normal vs abnormal classes (Figure 4.1), we can take advantage of some properties of time series and construct features that can represent the longer torque time series into a more compact feature vector. Constructing these features are usually selected manually and are subjective for a given dataset and there is not yet an standard procedure for this process. Hence comprehensive methodological research is required to determine whether alternative feature-based representation of time series could be simpler and/or outperform a manually-selected representation.

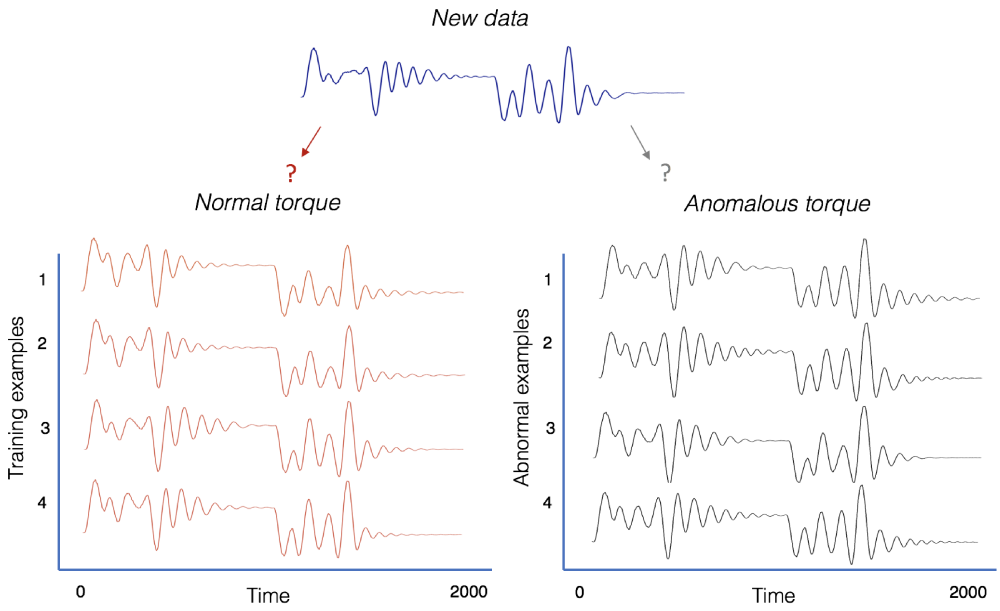

Figure 4.1: Time series classification of normal vs abnormal

Some examples of feature-based time series representation include the work of Deng et al., where measures of mean, spread and trend in local time-series intervals were used to classify different types of time series [20]. Another example is from Morchen where features were derived using the wavelet and Fourier transforms of various time-series datasets for a classification task [53]. As a first attempt to constructing features for the torque time series dataset (2000 data points per sample), we have followed the work of [55] and used the first order statistics along other descriptive measures to extract a total of 11 features including mean(μ), variance(σ), skewness, kurtosis, crest-factor, min, max, peak to peak, median, root mean square and standard deviation, which reduces the data representing a single observation approximately 200 times. Table 4.1 lists the constructed features and their related formula.

In a second attempt, additional features were extracted using the "Time Series Feature extraction based on scalable hypothesis tests" abbreviated as tsfresh python package [16]. The package contains numerous feature extraction methods and a robust feature selection algorithm. These features describe basic characteristics of a time series such as the ones we have manually extracted (mean, max, min etc.) and some more complex features such as fast Fourier transform and continuous wavelet coefficients and many more. Because the tsfresh is capable of extracting 100s of features, to avoid irrelevant features, it uses a built-in filtering procedure to evaluate the explaining power and importance of features and retrieve most relevant properties for a time series. The details of the algorithm is discussed in [17]. Using the tsfresh, a total of 714 features were extracted, which was further pruned to 114 using the built-in feature selection method. Table 4.2 summarizes the manual features and the tsfresh features extracted from the torque time series dataset.

S. no	Statistical indicator	Formula	Remark
1	RMS	$\sqrt{\frac{\sum_{n=1}^{N}(x(n)-\mu)^2}{N}}$	Normalized second statistical moment of signal.
2	Kurtosis	$\frac{\sum_{n=1}^{N}(x(n)-\mu)^4}{N\sigma^4}$	Normalized fourth statistical moment of signal. Shows the measure of the impulsive nature of signal.
3	Skewness	$\frac{\sum_{n=1}^{N}(x(n)-\mu)^3}{N\sigma^3}$	Measures the presence or lack of symmetry.
4	Maximum	$\max[x]$	Finds the highest point in a set of values.
5	Minimum	$\min[x]$	Finds the minimum point in a set of values.
6	Crest Factor	$\frac{PeakValue}{RMS}$	The ratio of peak level to RMS level. It shows the presence of high amplitude peaks in signal.
7	Mean	$\frac{\sum_{n=1}^{N}(x(n))}{N}$	Average of all the amplitudes of digitized points sampled.
8	Variance	$\frac{\sum_{n=1}^{N}(x(n)-\mu)^2}{N}$	Shows the spread of the amplitude of the values from its mean.
9	STD	$\sqrt{\frac{\sum_{n=1}^{N}(x(n)-\mu)^2}{N}}$	Simular to description for Variance
10	Peak to Peak	$\max[x] - \min[x]$	Measures the distance between the maximum amplitude and the minimum amplitude of signal.
11	Median	Sort values, pick the value in middle.	The value separating the higher half from the lower half of a signal. The middle value of signal.

RMS: root mean square; x(n): amplitude of the n^{th} digitized point in the time domain; N: number of points in time domain; μ: mean of the N points; σ: standard deviation.

Table 4.1: Manual feature construction using the descriptive statistics of torque time-seris

Method	Features
Manual (11 features)	First order moments and other descriptive statistics: • mean, variance, skewness, kurtosis, • median, standard deviation, • peak-to-peak, crest factor, max , min
tsfresh package (114 features)	Numerous basic and complex properties of time series, some examples include: • mean, max, variance, etc. • Fast Fourier, Continuous wavelet coeffiecients, • absolute energy, approximate entropy, etc.

Table 4.2: Manual and automated (tsfresh) feature construction

4.1.4 Feature selection

Feature selection is the process of selecting a subset of discriminating features that best describe a data sample and helps in filtering out irrelevant or poorly contributing attributes. Feature selection is usually performed for several reasons: to improve the performance of a machine learning algorithm by detecting irrelevant or noise features that may contribute to overfitting and result in a poor model. Model simplification, where features are ranked according to degree of importance and helps in understanding which features contribute the most. It also reduces computation resources and helps save data storage and processing through dimension reduction. Last but not least, it helps in overcoming the major problem of curse-of-dimensionality specially in sensor data. There are different approaches to feature selection, namely, filter methods and wrapper methods. Filter methods are more robust because they evaluate the features independent from any classification scheme, while wrapper methods use a measure of accuracy for a specific classifier to assess the quality of features [46].

The tsfresh package uses a variety of filter-based feature selection methods to find irrelevant features and ultimately keeping ones that are most relevant. Fortunately, for this case study, do to availability of labels for normal and abnormal classes, we can take advantage of the wrapper methods and further

analyze the selected subset of features in order to find the degree of importance of each feature. Hence, a feature selector package implemented in python was used to identify correlated, zero-importance and low-importance features for each of the feature construction methods previously mentioned. The steps in feature selection for the robot arm torque dataset is described as follows:

1. Construct manual and tsfresh features for normal and abnormal data.

2. Apply tsfresh filter selection on the tsfresh features to further reduce the 714 features to 114

3. Apply the python feature selection wrapper method to further analyze the manual and reduced tsfresh features.

   Correlated features are found using the Pearson correlation with a default threshold and the absolute ranking of features above the threshold are removed.

   Zero-importance features are found using a gradient boosting machine implemented in the LightGBM library. To reduce the variance of the calculated feature importances, the model is trained 10 times with early stopping with a validation set (15% of data) to avoid overfitting.

   Low-importance features are also determined by passing a cumulative total feature importance to the gradient boosting machine.

4. The number of features is then selected based on the cumulative feature importance graph which shows the required number of features to maximize the data representation. This is the dotted orange line on the cumulative feature importance graph (see Figure 4.4).

As a sample, the correlation matrix (Figure 4.2) and feature importance diagrams (Figures 4.3, 4.4) for the manual features are shown below:

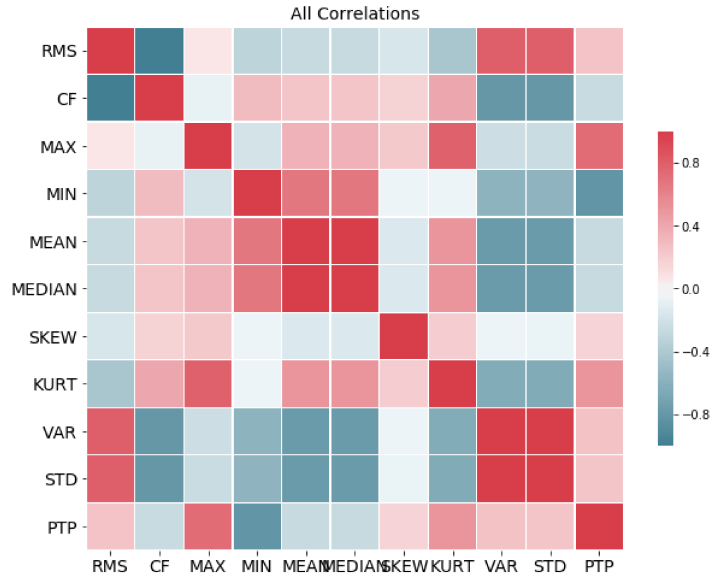

Figure 4.2: Correlation matrix for manually constructed features of the torque time series

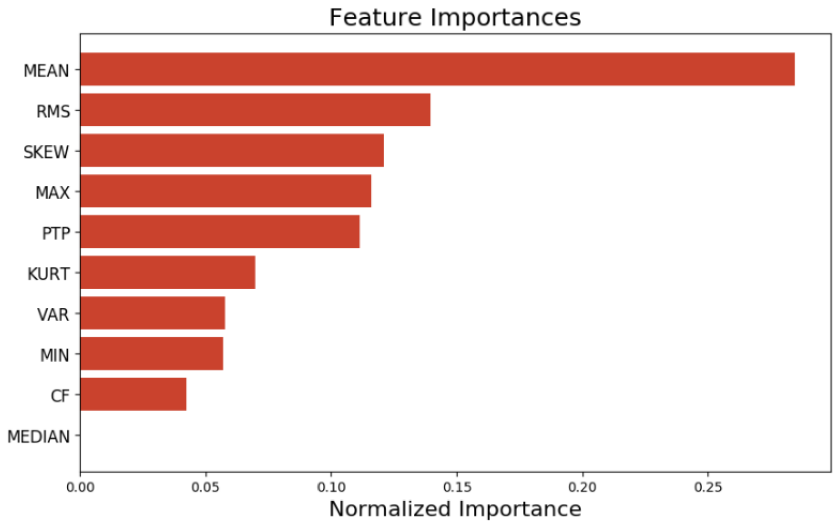

Figure 4.3: Normalized importance per manual feature

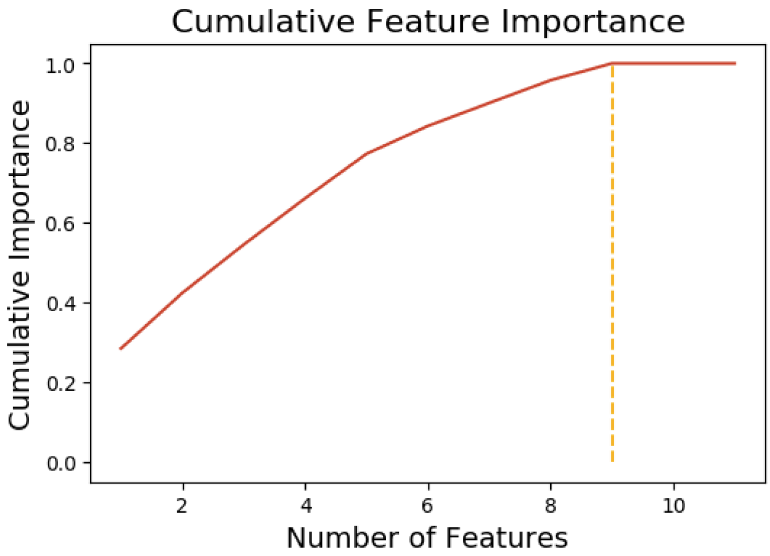

Figure 4.4: Cumulative importance for manual features. As shown in graph, a total of 9 features are required to explain 99% of data

Using the information from the cumulative feature importance and results of the correlation matrix, the correlated and less important features of the manual and tsfresh attributes were removed and results yield a total of 8 (Table 4.3) and 114 features respectively per feature construction method.

Analysis	**Features to remove**
Pearson Correlation	Crest factor, Median, Standard deviation
Zero Importance	Median, Standard deviation
Low Importance (99% threshold)	Median, Standard deviation, Crest factor
Total selected features: 8	Mean, RMS, Skewness, Max, Min, Peak-to-peak, Kurtosis, Variance

Table 4.3: Most important features for the manual feature construction after feature selection analysis

4.1.5 Feature learning

A significant task in advancing PHM analysis using machine learning and data-driven approaches is finding and extracting a set of good descriptive features.

Both quantity and quality sets the basis of a good predictive model, hence the better the features the better the performance of the machine learning model. However, figuring out and deriving good features could be very difficult and time consuming and in many cases requires expert knowledge [41]. Being able to extract such good features automatically is a subject that has gained the interest of researchers in many domains. With the emergence of deep learning and its capability in automatically finding the representation needed for feature detection and classification from raw data, the focus has been shifted towards the use of these techniques for automated feature learning. If the deep learning architecture can be used to extract features that better represent the underlying problem, the manual feature engineering process can be replaced. In PHM, state-of-the-art feature learning methods have employed unsupervised (auto-encoders, restricted Boltzmann machines, etc.) and supervised (e.g. convolutional neural networks) deep learning methods to extract features from raw data.

For this research problem, we have also employed unsupervised neural network structures, mainly auto-encoders to investigate the performance of the networks as anomaly detectors and the usefulness of the extracted features for the task of anomaly detection.

The auto-encoders include shallow and deep feedforward encoding and decoding layers with various number of neurones in each layer. A Convolutional Neural Network Auto-encoder (CNNAE) with different numbers of alternating convolutional and pooling layers was also implemented. The networks are used as anomaly detectors and the reconstruction error between the original input and the reconstructed output is used as anomaly score. The learned features from the bottleneck layer of the CNNAE network was also extracted and saved to use with the anomaly detection methods to further investigate the performance of automatically extracted features. The results of using the features learned from the CNNAE is demonstrated in the results section.

4.2 Anomaly detection evaluation

In an ideal scenario, we would like anomaly detection algorithms to detect and identify all and only anomalies. But, in reality there is a tradeoff between these two.

4.2.1 Precision, recall, F1-score

For a given anomaly detection algorithm, we are interested in characterizing how well it identifies all and only anomalies. In information retrieval, there are some key concepts that can help:

Precision - A measure of how well the detector identifies only anomalies. For example, if the algorithm returns a set of anomalies for threshold t, some of them are real anomalies and some are not. Precision is the percentage of real anomalies in a dataset.

Recall - Measures how well all anomalies are identified. For a given dataset, some samples are normal and some are anomalies. The detectors identify a set S of anomalies. S captures some portion of the complete anomalies which is measured as recall.

F1-score - The F1-score is the measure of a test's accuracy. It considers both precision and recall of the test to compute the score. Thus, we would like the F1 to be as close as to 1 as possible for a 100% accurate results.

G-Mean - The geometric mean is a measure that measures the balance between classification performances with regards to both the majority and minority classes. A low G-Mean indicates that a model is performing poor on classification of the positive cases even if negative samples are being correctly classified.

4.2.2 Area Under the Receiver Operating Characteristics (AUROC)

For a classification problem, e.g., classification between normal vs abnormal, a good measure of classification performance is the AUROC. The AUROC presents how well the model is capable of separating and distinguishing classes in our dataset. A higher AUC value denotes a model with great class separation capability. So for example, in our case, a model with high AUC can better distinguish between normal and anomalous torque samples. Hence we use the AUC as an indication of how well the models being compared perform on separating the normal from abnormal samples.

4.2.3 Statistical Significance Test

In order to assess the skills of each detector, we have used resampling method like 5-fold cross-validation and calculated the mean skill scores and compare them directly. This is a very simple approach and does not show whether the difference between the mean skill scores is real or are the results of statistical chance. To resolve this issue, we use statistical significance tests which quantify how likely the observation of a sample of the skill score is under the assumption that they come from the same distribution.

The assumption of the statistical test is called the null hypothesis, and to accept or reject this hypothesis we calculate statistical measures that help us in the decision making. A null hypothesis, suggests that no there is no significant difference among a set of given models. For the purpose of comparing machine learning models the Null and Alternate hypothesis are as follows:

- Null (H0): Given two models, no significance difference exists in the means of the skill scores
- Alternate: Given two models, there exists a significant difference and Null hypothesis is rejected.

Given a pre-defined significance level (alpha), the p-value computed by a significance test can be interpreted as follows:

- $p >$ alpha: fail to reject H0
- $p \leq$ alpha: reject H0, significant difference exists

One-way Analysis of Variance (ANOVA)

The one-way ANOVA compares the means between the different anomaly detection methods skills (any of F1-score, AUC, precision or recall) we are interested in and determines whether any of those means are statistically significantly different from each other. It tests the null hypothesis:

$$H0 : \mu_1 = \mu_2 = \mu_3 = ... = \mu_k$$

where k is the number of models. If the one-way ANOVA outputs a statistically significant result, the Alternate hypothesis is accepted, indicating that there exists at least two models with skill means that are statistically significantly different from each other. Hence, a subsequent test using other methods of statistical significance test needs to be performed to determine which specific models were different from each other.

4.3 Training and parameter tuning

As mentioned before, in order to select the most suitable methods for the task of detecting the onset of machine failure using anomaly detection methods, one is required to test and compare several different detectors. Every anomaly detection has its own advantages and disadvantages as reviewed in Chapter 2 of this manuscript. Based on these factors, ‘

Table 4.4 provides a summary of the parameter(s) and architecture(s) used in finding the most suitable settings for each detector.

Each of the algorithms and their respective configurations, were validated against a 5-fold random permutation cross-validation (shuffle & split) and was given a random seed in order to reproduce the same folds per algorithm configuration. The normal dataset was split into train, validation and test. The train and validation datasets are used at the time of training with the

Algorithm	Parameters	Configs/Values
k-NN	num. of neighbours	[10, 20, 30]
LOF	min. points	[10, 20, 30]
ABOD	num. of neighbours	[10, 20, 30]
OCSVM	kernels, nu (ν)	poly, rbf:[0.2, 0.5, 0.7, 0.9]
FFAE	arch., optim., act.	[1500, 1000, 500, 200, 100, 50, 20, 10] [1000, 500, 250, 100] [1000, 200,10], [1000, 200], [500] relu, tanh opt: adam, rmsprop
CNNAE	arch., optim., act.	con(256,10)-maxpool-4xconv(3,1)-maxpool conv(64, 10)-maxpool conv(64,1)-maxpool-4xconv(10,1)-maxpool conv(64,10)-maxpool-3xconv(8,1)-maxpool optm: adam, rmsprop relu, tanh

Table 4.4: The parameters and architectures used to spot check each algorithm and finding the best performing settings. [no. of neurons] shows the FFAE and CNNAE layers and neurones

validation dataset used for validation and early stopping criterion for the auto-encoders to prevent overfitting. The normal *test* dataset was set aside to be merged with the abnormal test datasets to obtain a more realistic testing set. Figure 4.5, shows the train, validation and test dataset. The normal train and validation datasets are used at training time and all detectors are fit using this dataset. The mixed data set is used at test time.

4.3.1 Selection of auto-encoder architecture

In this study we developed deep auto-encoders as an anomaly detectors for detecting the abnormal behaviour of the robot arm. The auto-encoder as anomaly detector with details was explained in Chapter 2 under the auto-encoder section. In the following section, the architecture of each of the networks, FFAE and CNNAE are explained.

	Train	Validation	Test
Normal	6473	10%-20%	720
Loose l1	-	-	182
Loose l2	-	-	180
Tight	-	-	194
High temperature	-	-	210
Sand (friction)	-	-	183
Mixed test dataset (Test-norm + Loose l1 + high temp)	-	-	1112

Figure 4.5: Normal and faulty data sets. Normal is split into train, validation and test. Faulty data is mixed with the normal test split and is only used at test time

Feed Forward Auto-encoder (FFAE)

The main parameters of the feed forward auto-encoder network include the number of hidden layers, number of neurons per layer, the activation function used per layer, the optimizer, number of epochs to train and batch size. In order to determine the best model, we have followed some existing settings found in literature [73][48][74] and further tuned the network with respect to our robot arm torque dataset. Table 4.4 shows the architecture, optimizers and activation functions we have used to select the best performing model. The model with best performance (based on precision, recall, F1 and AUC) on our validation dataset was selected as the model to be compared with other anomaly detection methods. The final network architecture includes 3 hidden layers for the encoder each with 1000, 200, 10 number of neurons and the same number of layers and neurons for the decoder network. The tanh activation function was used for each layer and the well known Adam [42] optimizer which is computationally efficient, uses little memory and is well suited for problems that are large in terms of data and/or parameters for gradient-based optimization. A schematic of the feed forward auto-encoder is shown in Figure 4.6.

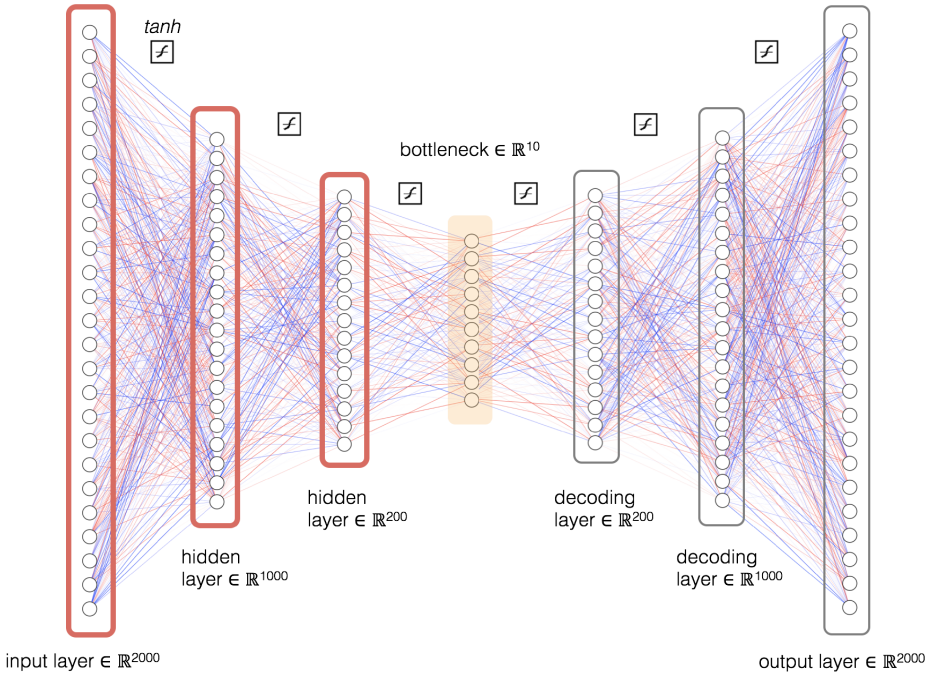

Figure 4.6: Feed forward auto-encoder architecture used as anomaly detector.

Convolutional Neural Network Auto-encoder (CNNAE)

Despite the fact that most applications of CNN for image recognition feed a 2D segment as input [26][77] and many researchers in the field of mechanical fault diagnosis have used the same method [14][29], in this study we use a 1D segment from raw input data (torque data) as the input to our CNN auto-encoder followed by 1D filters in the convolutional layers of the model. The use of 2D segments in image data maybe due to the fact that images tend to have a natural 2D space correlation [45], however our torque signal is correlated only in the time domain which is a 1D parameter. We refer the reader to the convolutional neural network section in Chapter 2 for detailed explanation of the 1D CNN.

Evidently, a deep learning model with deeper architecture achieves a better performance compared to a shallow one. However, a deeper network results in a more complicated number of hyper-parameters and a variety of network architectures. This in fact makes the building of an appropriate and efficient model quite difficult [71]. For the purpose of this study we have followed a

few general design patterns [69][39] and particularly followed that of Zhang et al., [87]. To achieve higher performance than traditional methods, Zhang et al., introduce the wide and deep architecture, where wider kernel in the first layer is followed by successive small kernels. They claim that smaller kernels in the first layer are affected much more by the high frequency noise common in industrial environments, and so to capture the useful information in the signal they use wide kernels in the first layer to extract features and then use subsequent small (3x1) kernels for a better feature representation. This allows for a deeper model with higher feature learning capability.

For the purpose of our study, the configuration of several key parameters of the network such as the size of filters (kernel), the number of filters in the convolutional layers and the number of convolutional and max-pooling layers were tested and analyzed. The performance of each configuration was analyzed by comparing the evaluation metrics (precision, recall, F1-score and AUC) and based on these metrics the best model and its parameters was selected. Finally, a CNN encoder network and a mirrored decoder network with 5 layers of alternating convolution and pooling layers was constructed. The first convolutional layer consists of 64 filters (dimensionality of output space) and kernel size of 1, followed by a max-pooling layer. The subsequent layers are alternating small (10x1) convolutional layers followed by max-pooling. The architecture of the CNNAE is shown in Figure 4.7.

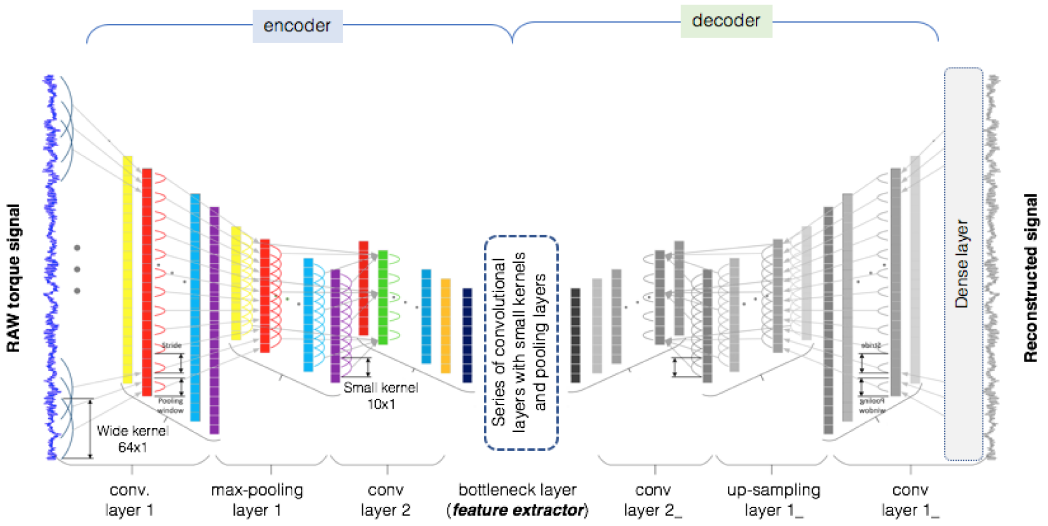

Figure 4.7: CNN auto-encoder architecture used as anomaly detector. The bottleneck layer features are used as automated features. [Image partially adapted from [87]]

The features from the bottleneck layer of the CNNAE, were also extracted and used with the classic anomaly detection methods. The original raw torque signals with 2000 data points is reduced down to 10 dimensions in the bottleneck layer. The results of using the learned features from the CNNAE is described in the results section and is compared with the other anomaly detectors.

4.3.2 Training and decision function

As a first step the algorithms were first trained using the normal only dataset with the raw torque values (2000 features). A contamination ratio which determines the amount of contamination of the data set, i.e., the proportion of outliers in the data set is predetermined (once assumed 90% of data are normal and once 95%) and used when fitting to define the threshold on the decision function. Once the detectors are fit, the outlier scores of the training set is determined using the corresponding decision function of a detector. This is because each detector has a specific way of calculating anomaly scores which was mentioned in Chapter 2 under the related algorithm. The threshold for calculating the boundary between a normal and anomalous data is calculated

as described in section 4.3.3.

The above procedure was applied for the feature-based classification of the torque signals using the hand crafted features, the tsfresh features and CNNAE extracted features.

4.3.3 Determining the threshold

A normally distributed dataset follows some standard laws. The standard deviation defines how far the distribution is spread around the mean.

- Usually 68% of all the values fall between $[\mu - \sigma, \mu + \sigma]$,
- 95% of all values fall between $[\mu - 2\sigma, \mu + 2\sigma]$,
- 99.7% of all values fall between $[\mu - 3\sigma, \mu + 3\sigma]$

These rules are known as the "three-sigma rule of thumb". When a system is following a normal distribution and it breaks the three-sigma rule, it is an indication of rare event.

To select a determining threshold as a cutoff point for flagging anomalous and normal samples, we have used two different thresholds. One at 95% and the other at 90%. The reason we did not use a number 3 standard deviations away from the mean is that, we are not looking for anomalies only, but the onset of failures.

The nearest rank method is then used to find the ordinal rank of the anomaly scores outputted by the detectors:

1. Sort the anomaly scores from smallest to largest score.
2. Determine the "ordinal rank".
3. Cut-off point is the score at the 'n^{th}' index.

$$n = \left\lceil \frac{p}{100} \times N \right\rceil$$

where

n: Ordinal rank

P: percentile (% of samples considered normal)

N: Total number of samples

4.4 Results and concluding remarks

The trained models were used on the unseen mixed normal and anomalous dataset to determine the anomaly scores. The anomaly scores were then categorized into anomalous and normal according to the set threshold (assuming 90% & 95% normal data) in previous step. An anomaly score beyond the threshold is then flagged with label '1' as being anomalous and '0' if smaller than threshold.

Figures 4.13 through 4.24 show the scores on a scatter plot with grey marking points identified as normal and orange points as anomalous for each of the raw, and manually constructed features. The threshold is represented as a golden horizontal line, marking the cut-off at which a sample is flagged normal or anomalous. Similar graphs for the T*-framework and the tsfresh features were generated which is included in Appendix A: Plots of scores for T*-framework (see Figure A.1) and Appendix B: Plots of scores for tsfresh (see Figure B.1). Tables 4.8, 4.9, 4.12 list a summary of results showing the time to train and test a model together with the evaluation metrics. For better readability, we have added Table 4.10 with a side by side comparison

of the performance of each detection algorithm using the 8 manual features. The results show that the performance of all anomaly detectors using the hand crafted features is comparable to that of the raw torque measurements/features for the detectors. Similarly, the 76 tsfresh features also produce comparable results to that of the 8 manually created and raw features, showing that the additional features extracted have no or near to none performance increase in the models. As for the results of detectors with features automatically extracted from the CNNAE (section 4.12), decent precision and recall scores are produced but they clearly demonstrate the need for further fine tuning of the CNNAE in order to produce results that are in par with the raw and manually constructed features.

In terms of detection performance of algorithms on various fault modes, all algorithms performed very well, but the most important factor for a boost in recall and F1-score is the choice of contamination ratio. We think the selection of this parameter depends on the sensitivity of the domain and requires expert knowledge. However, looking at the AUROC, we can see that all algorithms performed very well under two different contamination rates (5% and 10%). Furthermore, to further analyze the performance of detectors, we have used the one-way ANOVA and McNemar's test statistics. The results of the one-way ANOVA along with McNemar's test results are shown in Figure 4.11, which indicate no statistically significant difference among the models mean skills, hence an agreement to the Null hypothesis.

If one is to select a single anomaly detection for the detection of onset of failure for the particular case of the robot arm we recommend the isolation forest anomaly detection. Although on average, based on the evaluation metrics, the algorithm is slightly (negligible if used for smaller datasets) slower in training time (robot arm data), some of its properties makes it the winner amongst the others. These properties include: no need for calculating distance or density of samples; at the time of training a tree is constructed and at test time it passes sample points down the tree to calculate the average number of edges required to reach an external node. It is very robust in terms of choosing parameters; to configure it, one would need to (1) provide maximum

samples to draw from training dataset to train each of the base estimators, (2) number of base estimators (the number of ensembles), (3) contamination, the ratio of outliers in the dataset (if used as a semi-supervised algorithm). Liu et al., [47], show that the choice of sample size is very robust and that the algorithm converges very fast with low sample size. We think the most important configuration parameter is the contamination ratio, which stands the same for all the other algorithms. However, according to Liu et al., the choice of contamination may be rectified by using a larger sampling size. In addition to its performance, the memory requirement for isolation forest is very low because of sub-sampling and so it makes it a great practical choice for deployment and in industry.

Overall, we show that detecting the onset of failure in the designed robot arm platform is achieved with a high accuracy using raw and manually constructed features by a variety of anomaly detectors. We further discuss the results of our experiment and research study in Chapter 5.

	Thr.	Manual (8) features						RAW (2000) features						Tsfresh (76) features					
		P	R	F1	AUC	Time train	Time test	P	R	F1	AUC	Time train	Time test	P	R	F1	AUC	Time train	Time test
ABOD (n = 30)	90%	0.8533	1	92.07	95.3	33.43	6.57	0.8523	1	92.02	95.28	192.05	36.94	0.845	1	91.58	94.99	37.7	6.73
	95%	0.9245	1	96.07	97.76	34.67	7.77	0.9227	1	95.97	97.71	180.54	32.4	0.9215	1	95.9	97.66	37.32	6.46
KNN (n = 30)	90%	0.8549	1	92.17	95.38	0.19	0.11	0.8528	1	92.05	95.29	141.9	25.85	0.8539	1	92.11	95.33	3.6	0.81
	95%	0.9274	1	96.23	97.86	0.22	0.13	0.9254	1	96.11	97.79	124.09	23.12	0.9187	1	95.76	97.58	3.53	0.74
LOF (n = 30)	90%	0.847	1	91.71	95.07	0.25	0.05	0.8555	1	92.2	95.39	142.28	26.61	0.8494	1	91.84	95.15	3.72	0.62
	95%	0.9157	0.9765	94.5	96.37	0.22	0.04	0.9231	1	95.99	97.72	122.17	22.69	0.9233	1	96	97.72	3.59	0.66
HBOS	90%	0.8578	1	92.35	95.49	0.15	0.01	0.8544	0.9964	92	95.2	1.48	0.07	0.8537	0.9995	92.08	95.31	0.05	0
	95%	0.9237	0.9684	94.55	96.24	0.18	0	0.9223	0.9495	93.57	95.29	1.17	0.07	0.9146	0.9031	90.87	92.85	0.05	0
Isolation Forest	90%	0.8557	1	92.22	95.4	0.55	0.06	0.8497	0.9974	91.76	95.07	29.11	1.14	0.851	0.9995	91.93	95.21	0.93	0.08
	95%	0.9264	99.69	96.03	97.68	0.59	0.06	0.9191	0.9209	91.98	93.84	26.98	1.01	0.9199	0.9571	93.8	95.58	0.95	0.07
OC-SVM (rbf, nu:0.7)	90%	0.8564	1	92.26	95.43	1.77	0.09	0.8513	0.9709	90.71	93.92	198.99	14.59	0.8564	1	92.26	95.43	7.62	0.51
	95%	0.9235	0.9781	95	96.69	1.85	0.11	0.9072	0.8235	86.3	88.88	180.57	13.39	0.9257	0.9765	95.04	96.69	7.74	0.52
PCA	90%	0.8613	1	92.54	95.61	0.02	0	0.8537	0.9903	91.69	94.89	44.02	6.35	0.8522	1	92.02	95.28	0.11	0.01
	95%	0.9255	1	96.13	97.81	0.02	0	0.9199	0.9128	91.63	93.47	41.39	5.96	0.9095	0.9005	90.49	92.58	0.1	0.01
FFAE (1000-200-10)	90%	-	-	-	-	-	-	0.8480	0.9837	91.06	94.37	17.12	0.10	-	-	-	-	-	-
	95%	-	-	-	-	-	-	0.9099	0.8541	88.08	90.40	18.76	0.1	-	-	-	-	-	-
CNNAE (64_10_3x8_1)	90%	-	-	-	-	-	-	0.8348	0.9745	89.90	93.46	87	0.57	-	-	-	-	-	-
	95%	-	-	-	-	-	-	0.9139	.90	90.50	92.55	88.84	0.616	-	-	-	-	-	-
CNNAE (64_1_4x10_1) b100-e50-adam-tanh	90%	-	-	-	-	-	-	0.8430	1	91.47	94.92	96.82	0.61	-	-	-	-	-	-
	95%	-	-	-	-	-	-	0.9152	0.9556	93.46	95.36	96.7	0.63	-	-	-	-	-	-
CNNAE (8_1_3_1)	90%	0.8464	0.9413	88.98	92.47	15.30	0.10	-	-	-	-	-	-	-	-	-	-	-	-
	95%	0.9159	0.9036	90.26	92.97	15.32	0.09	-	-	-	-	-	-	-	-	-	-	-	-

Figure 4.8: Results of the detectors applied to test data (mix of normal + faulty [loose l1 + high temperature]). The F1-score is an indication of the final accuracy. The closer to 1 the better the performance of a model

	T* Framework – Manual (8) features					
	Thr.	Avg. P	Avg. R	Avg. F1	Avg. AUC	Avg. Time train & test
ABOD (n = 30)	90%	0.802	0.796	78	84.45	1876
	95%	"	"	"	"	1881
KNN (n = 30)	90%	0.852	1	92	94.88	13.70
	95%	"	"	"	"	13.70
LOF (n = 30)	90%	0.832	1	90.80	94.44	2.55
	95%	"	"	"	"	2.25
HBOS	90%	0.858	1	92.40	95.49	0.30
	95%	"	"	"	"	0.056
Isolation Forest	90%	0.854	1	92.20	95.38	18.67
	95%	"	"	"	"	17.6
OC-SVM (rbf, nu:0.7)	90%	0.852	1	92.2	94.88	62.32
	95%	"	"	"	"	62.37
PCA	90%	0.852	0.988	91.60	94.64	0.083
	95%	"	"	"	"	0.083

Figure 4.9: T*-framework applied to mixed data (normal + faulty[loose l1, high temperature]) with threshold set at 90% and 95%. The symbol " means the same results for 95%

	k-NN	T*k-NN	LOF	T*LOF	ABOD	T*ABOD	iForest	T*iForest	HBOS	T*HBOS	OCSVM	T*OCSVM	PCA	T*PCA	CNNAE
F1-Score (Mean ± STD)	92.17 ±0.005	92 ±0.0071	91.71 ±0.01	90.8 ±0.0045	92.07 ±0.009	79.8 ±0.013	92.22 ±0.006	92.20 ±0.0071	92.35 ±0.006	92.40 ±0.0055	92.26 ±0.006	92.2 ±0.0084	92.54 ±0.006	91.6 ±0.0055	88.98 ±0.0035
g-means (Mean ± STD)	95.28 ±0.0039	95.2 ±0.0045	94.92 ±0.007	94.4 ±0.005	95.32 ±0.006	84.4 ±0.0055	95.65 ±0.0043	95.20 ±0.0045	95.14 ±0.0041	95.40 ±0.0055	95.54 ±0.004	95.2 ±0.0084	95.51 ±0.004	94.8 ±0.0045	90.1 ±0.005
Train time (sec)	0.19	13.7	0.25	2.55	33.43	1876	0.55	18.67	0.15	0.30	1.77	62.32	0.02	0.083	15.37
Test time (sec)	0.11		0.05		6.57		0.06		0.01		0.09		0		0.09
Mem. Footprint (KB)	705	9.66MB	2300	53.2MB	612	7.73MB	924	27.66MB	106	2.94MB	450	6.49MB	107	2.94MB	2.0MB

(a)

	k-NN	T*k-NN	LOF	T*LOF	ABOD	T*ABOD	iForest	T*iForest	HBOS	T*HBOS	OCSVM	T*OCSVM	PCA	T*PCA	CNNAE
F1-Score	96.23 ±0.011	92 ±0.0071	94.5 ±0.01	90.8 ±0.0045	96.07 ±0.011	79.8 ±0.013	96.03 ±0.0105	92.20 ±0.0071	94.55 ±0.008	92.40 ±0.0055	95 ±0.005	92.2 ±0.0084	96.13 ±0.006	91.6 ±0.0055	90.26 ±0.003
g-means	97.79 ±0.007	95.2 ±0.0045	96.68 ±0.006	94.4 ±0.005	97.72 ±0.006	84.4 ±0.0055	97.33 ±0.007	95.20 ±0.0045	93.14 ±0.005	95.40 ±0.0055	97.96 ±0.0031	95.2 ±0.0084	97.78 ±0.0037	94.8 ±0.0045	92.3 ±0.005
Train time (sec)	0.22	13.7	0.22	2.55	34.67	1881	0.59	17.6	0.18	0.056	1.85	62.37	0.02	0.083	15.32
Test time (sec)	0.13		0.04		7.77		0.06		0.0		0.11		0		0.09
Mem. Footprint (KB)	705	9.66MB	2300	53.2MB	612	7.73MB	924	27.66MB	106	2.94MB	450	6.49MB	107	2.94MB	2.0MB

(b)

Figure 4.10: A side by side comparison of the anomaly detection algorithms applied to mixed data (normal + faulty[loose l1, high temperature]) with threshold set at (a) 90% and (b) at 95%.

	Precision	Recall	F1-score	AUC
One-way ANOVA (Statistic, p-value) at α = 0.05	(0.92, 0.493)	(1, 0.444)	(0.95, 0.474)	(0.97, 0.461)

α = 0.05	ABOD	KNN	LOF	HBOS	iForest	OCSVM	PCA
ABOD χ2 p-value		1.89 0.168	0.484 0.486	0.984 0.321	2.160 0.141	2.22 0.136	0.79 0.374
KNN χ2 p-value			2.75 0.0972	0.015 0.902	0.301 0.582	0.214 0.643	0.0 1.0
LOF χ2 p-value				2.11 0.146	3.71 0.054	3.51 0.06	1.87 0.171
HBOS χ2 p-value					0.102 0.748	0.026 0.871	0.0 1.0
iForest χ2 p-value						0.0 1.0	0.321 0.570
OCSVM χ2 p-value							0.148 0.700
PCA χ2 p-value							
At α = 0.05 all models are performing similar, so the H0 hypothesis is not rejected!							

Figure 4.11: The one-way ANOVA and McNemar's statistical significance test results. The one-way ANOVA statistics do not reject the Null hypothesis (H0), hence, skills of models are not considered statistically significantly different.

	CNN (10) features						
	Thr.	Avg. P	Avg. R	Avg. F1	Avg. AUC	Avg. Time train	Avg. Time test
ABOD (n = 30)	90%	0.83	0.913	87	90.56	33.09	6.60
	95%	0.83	0.88	0.85	89.22	37.37	6.30
KNN (n = 30)	90%	0.83	0.912	86.8	90.48	0.19	0.11
	95%	0.83	0.88	85	88.88	0.17	0.10
LOF (n = 30)	90%	0.828	0.90	87	90.79	0.2	0.034
	95%	0.83	0.86	84	88.22	0.2	0.03
HBOS	90%	0.782	0.8557	72.6	79.4	0.176	0
	95%	0.78	0.74	76	81.49	0.20	0
Isolation Forest	90%	0.814	0.80	80	85.188	0.572	0.056
	95%	0.81	0.82	81	85.67	0.48	0.08
OC-SVM (rbf, nu:0.7)	90%	0.786	0.714	74.2	80.73	1.82	0.12
	95%	0.80	0.75	0.77	82.54	1.78	0.10
PCA	90%	0.784	0.73	74.6	81.50	0.02	0
	95%	0.80	0.74	77	82.09	0	0

Figure 4.12: CNN features extracted from the raw robot arm data. The extracted features are used with the detectors to compare raw vs. manual vs. automated features

Figure 4.13: Trained models applied to mixed data (normal + faulty[loose l1, high temperature]) with threshold set at 90%. Gret points mark normal, orange points mark abnormal data

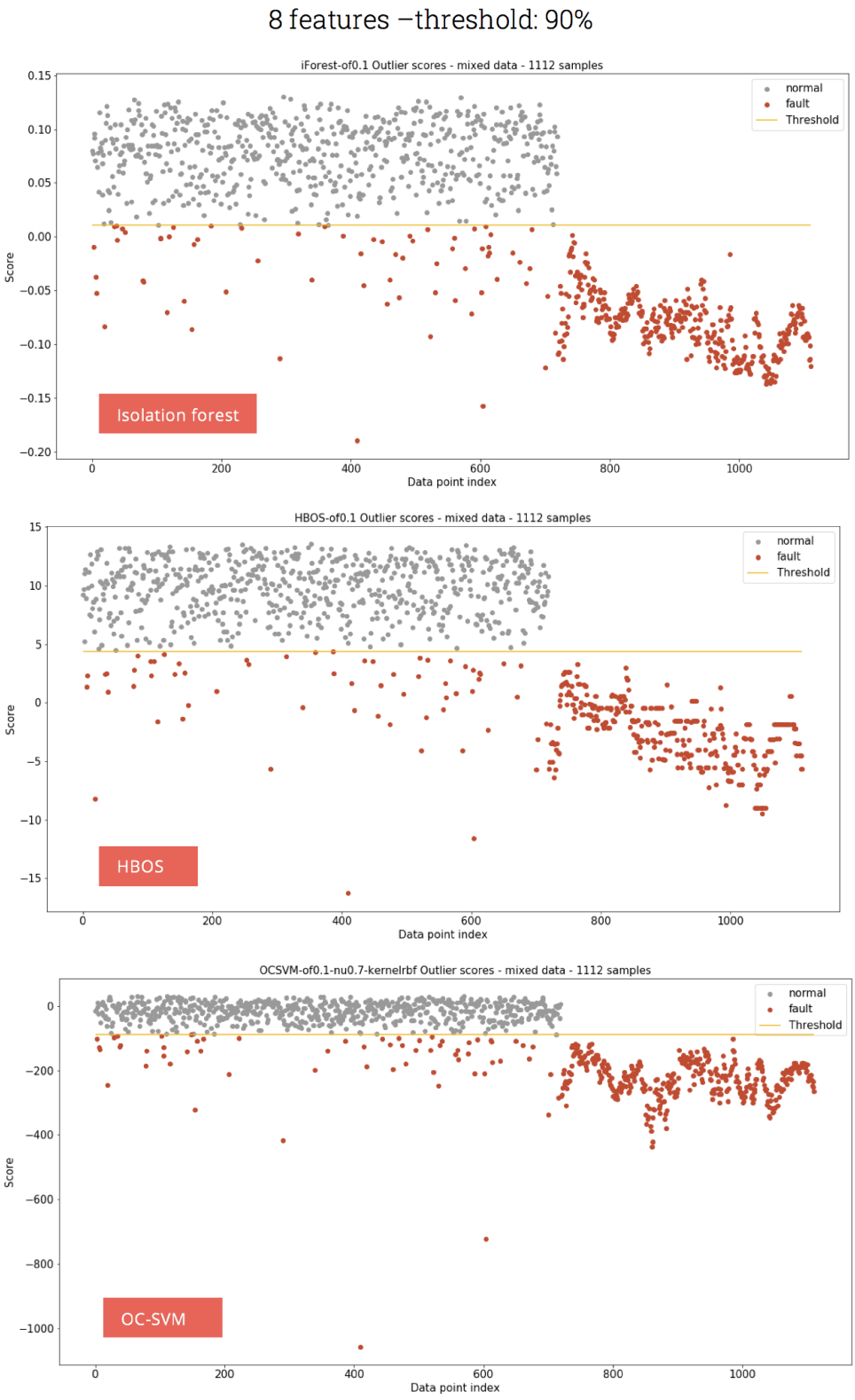

Figure 4.14: Trained models applied to mixed data (normal + faulty[loose l1, high temperature]) with threshold set at 90%. Grey points mark normal, orange points mark abnormal data

Figure 4.15: Trained models applied to mixed data (normal + faulty[loose l1, high temperature]) with threshold set at 90%. Grey points mark normal, orange points mark abnormal data

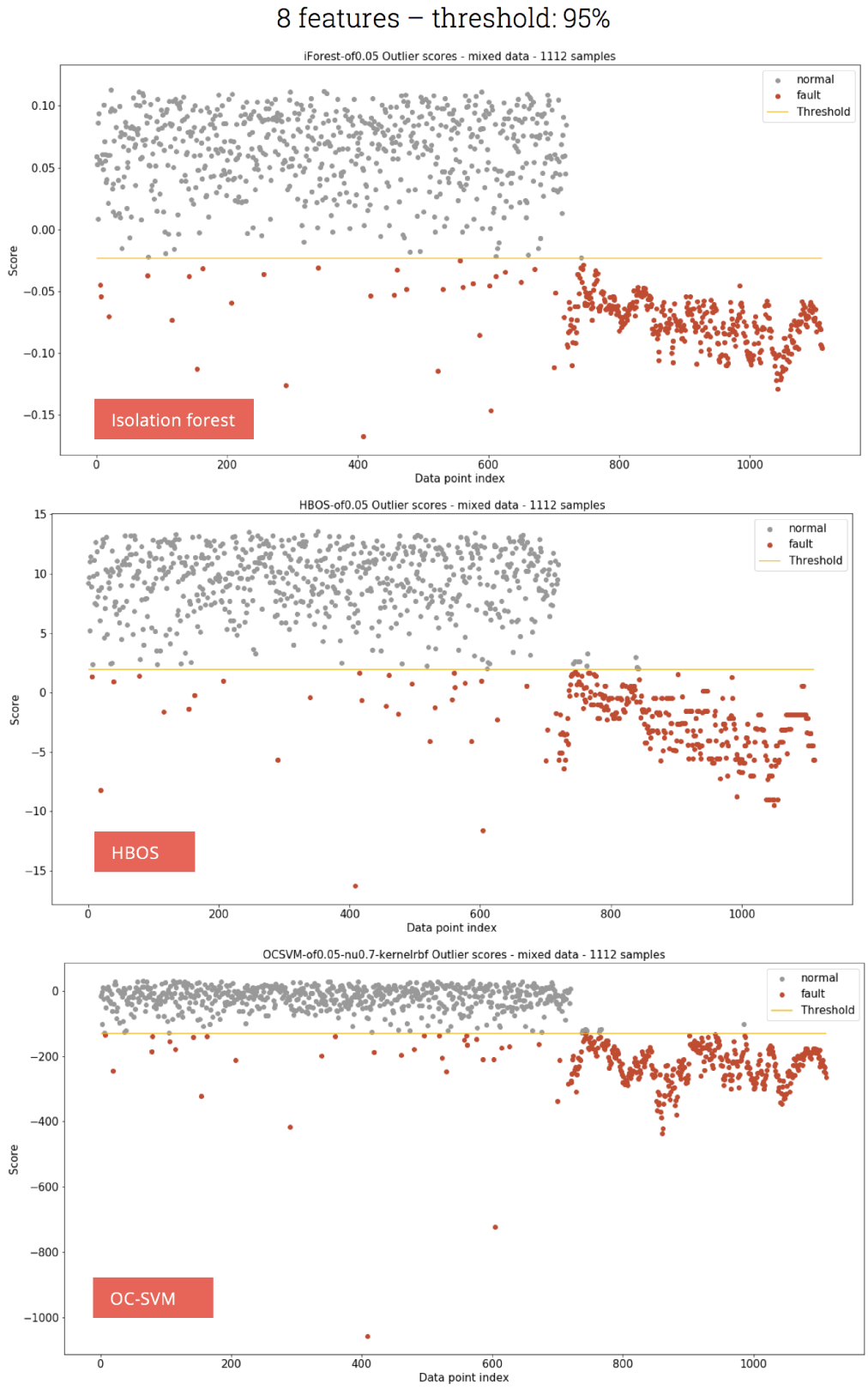

Figure 4.16: Trained models applied to mixed data (normal + faulty[loose l1, high temperature]) with threshold set at 95%. Grey points mark normal, orange points mark abnormal data

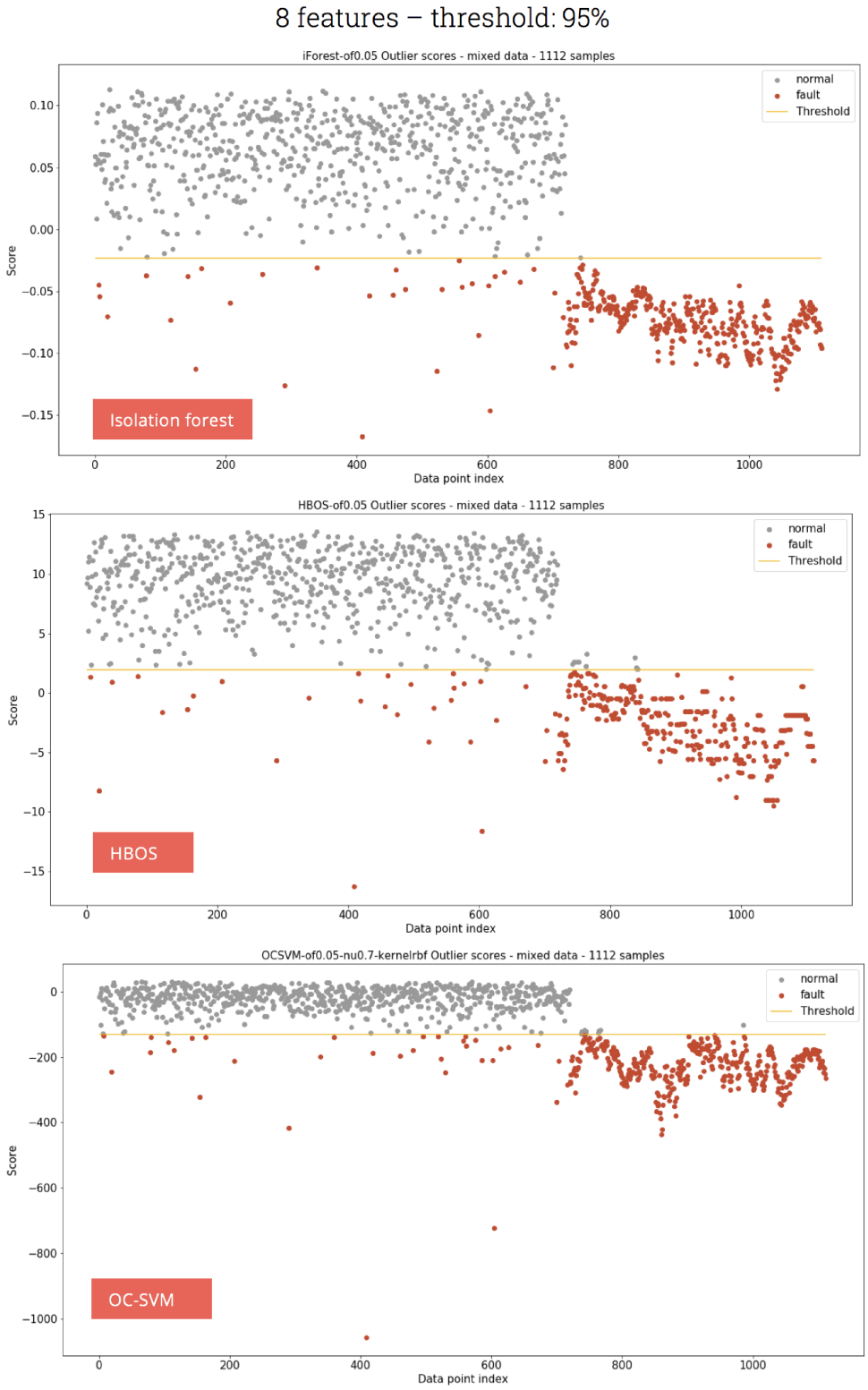

Figure 4.17: Trained models applied to mixed data (normal + faulty[loose l1, high temperature]) with threshold set at 95%. Grey points mark normal, orange points mark abnormal data

Figure 4.18: Trained models applied to mixed data (normal + faulty[loose l1, high temperature]) with threshold set at 95%. Grey points mark normal, orange points mark abnormal data

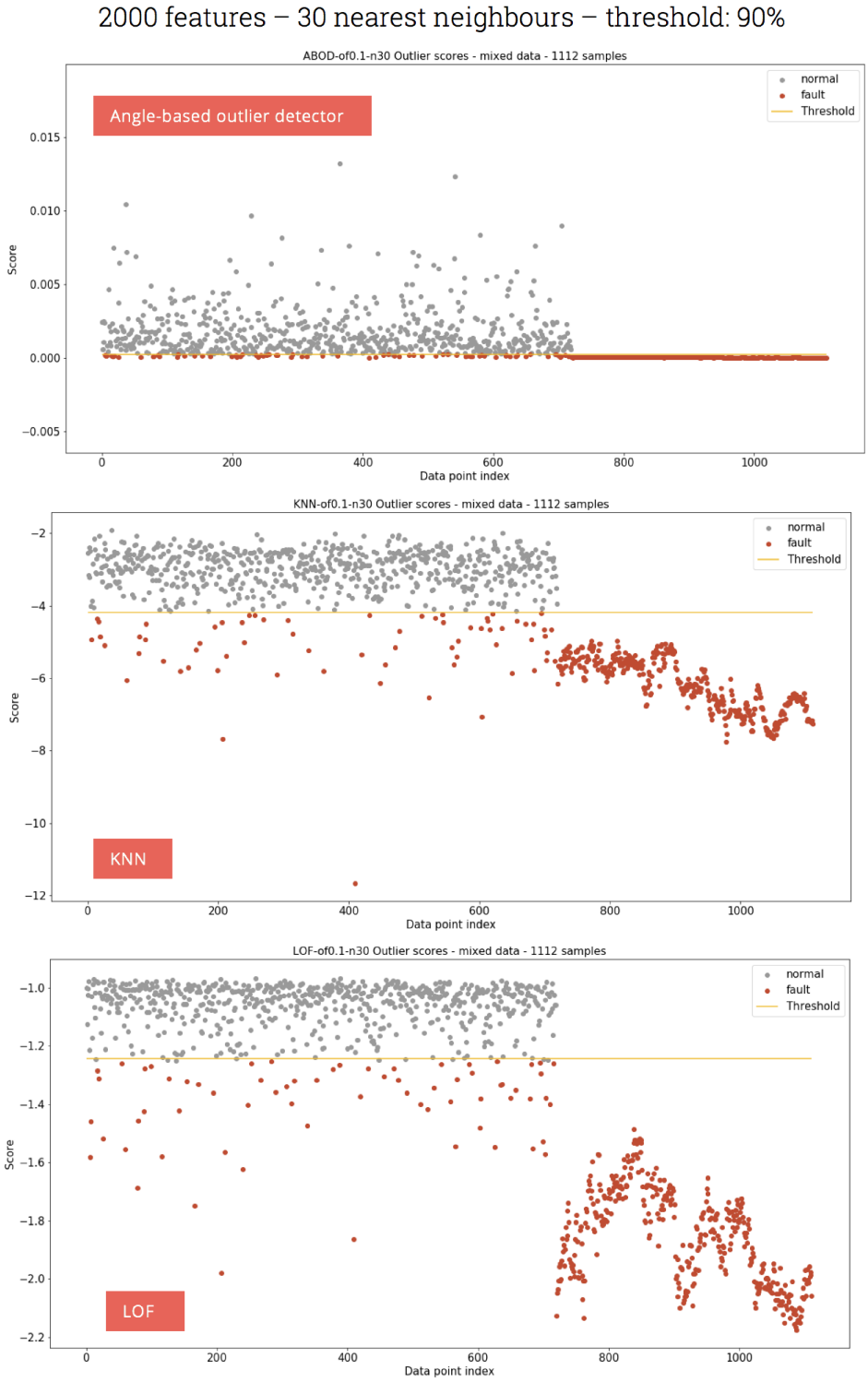

Figure 4.19: Trained models applied to mixed data - 2000 features (normal + faulty[loose l1, high temperature]) with threshold set at 90%. Gret points mark normal, orange points mark abnormal data

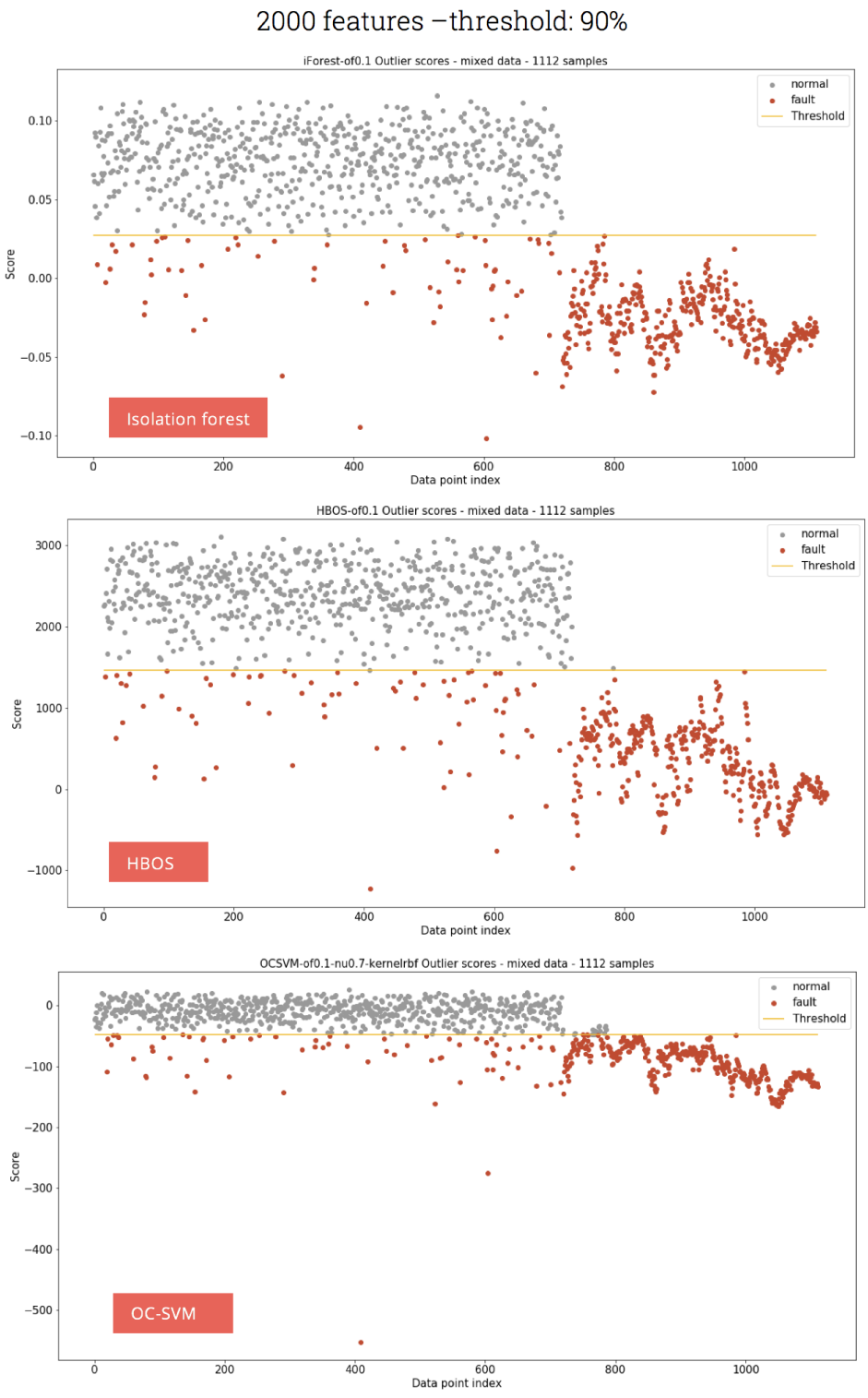

Figure 4.20: Trained models applied to mixed data - 2000 features (normal + faulty[loose l1, high temperature]) with threshold set at 90%. Grey points mark normal, orange points mark abnormal data

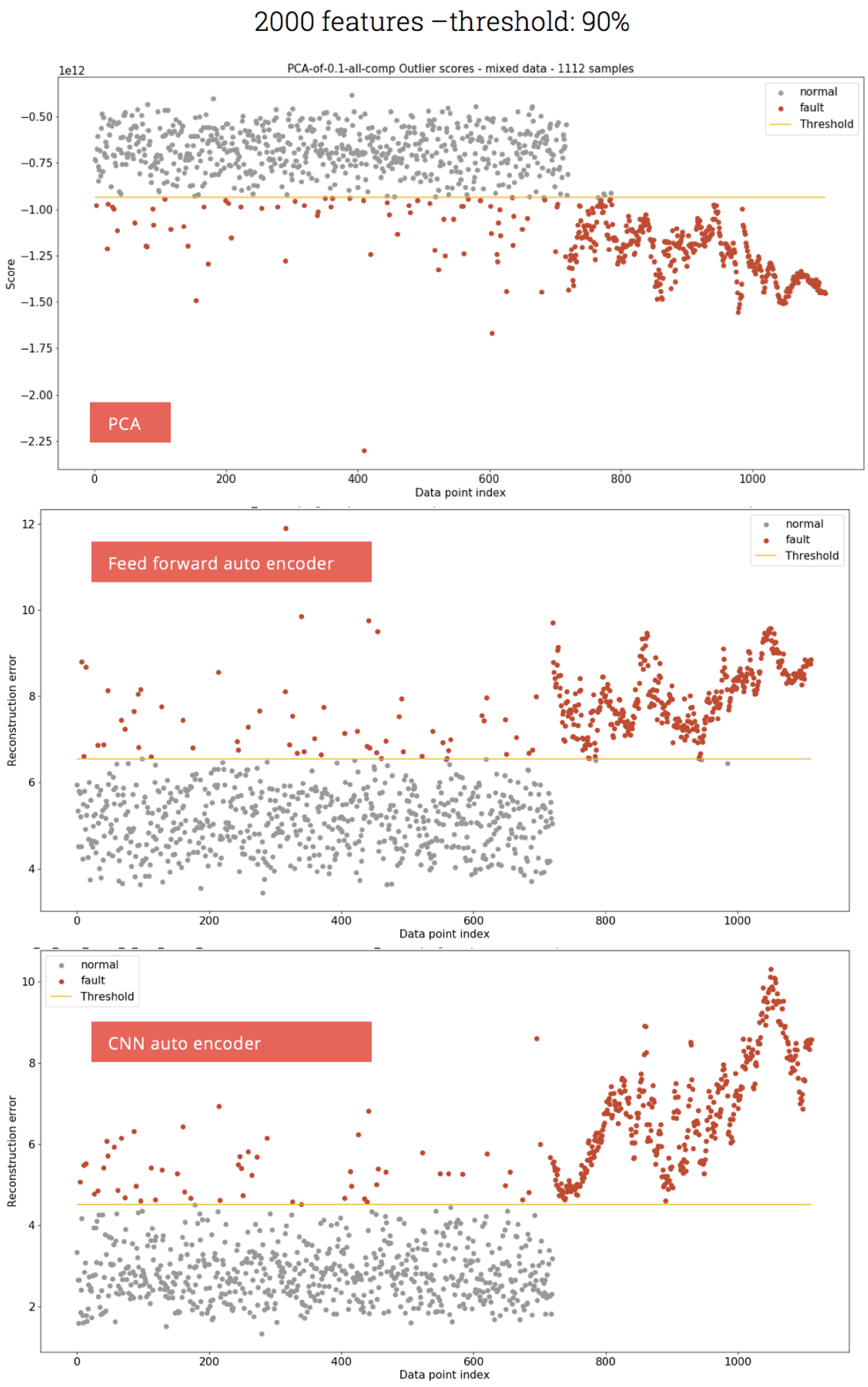

Figure 4.21: Trained models applied to mixed data - 2000 features (normal + faulty[loose l1, high temperature]) with threshold set at 90%. Grey points mark normal, orange points mark abnormal data

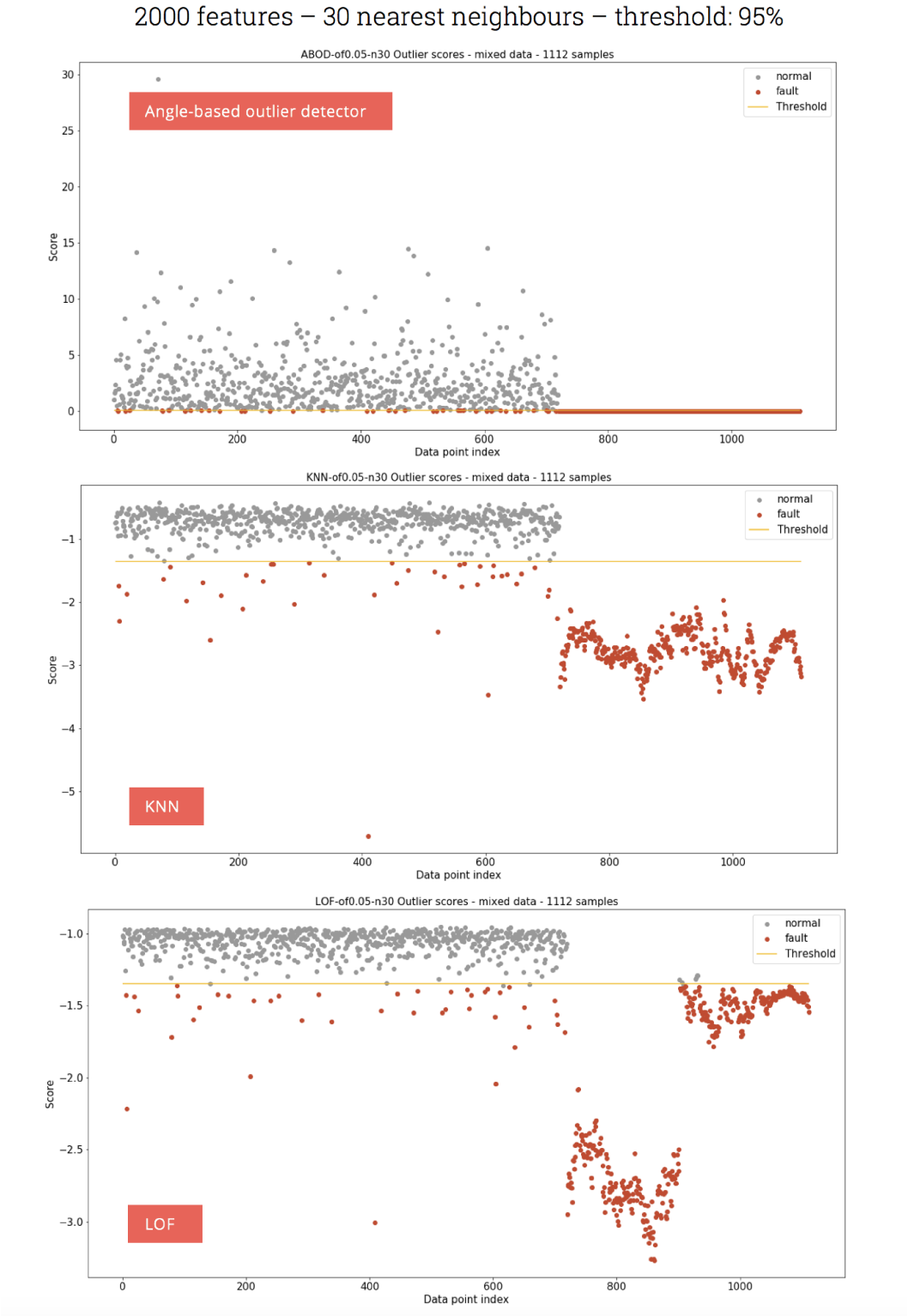

Figure 4.22: Trained models applied to mixed data - 2000 features (normal + faulty[loose l1, high temperature]) with threshold set at 95%. Grey points mark normal, orange points mark abnormal data

Figure 4.23: Trained models applied to mixed data - 2000 features (normal + faulty[loose l1, high temperature]) with threshold set at 95%. Grey points mark normal, orange points mark abnormal data

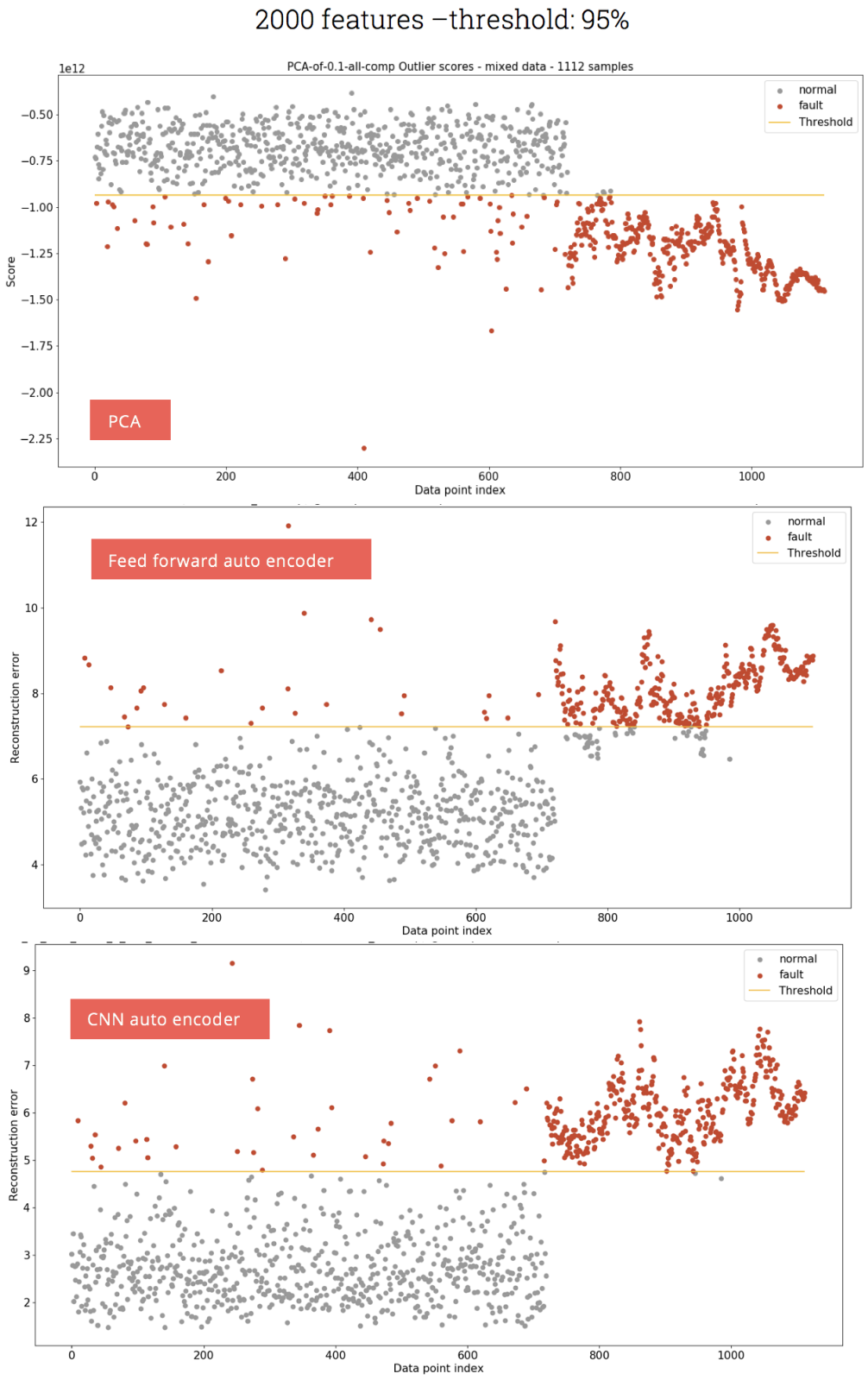

Figure 4.24: Trained models applied to mixed data - 2000 features (normal + faulty[loose l1, high temperature]) with threshold set at 95%. Grey points mark normal, orange points mark abnormal data

Chapter 5

Conclusion

5.1 Discussion

The main focus of this study was to investigate the application of anomaly detection algorithms for detecting the onset of failure in a machinery equipment. Unfortunately, access to data on such a problem set is limited or not available at all. Hence, in order to generate a realistic and close to real-life data that includes different operating conditions (e.g. normal and subtle abnormal faults), a single degree, belt-driven robot arm platform was designed and commissioned to simulate a gate system opening and closing operation. The health status of the robot arm was conditioned on the output torque of the motor measured via the calibrated strain gauge configuration on the shaft. The torque values for the normal operating conditions were collected in order to train a number of different anomaly detection methods including k-nearest neighbour, local outlier factor, angle-based outlier detection, histogram- based outlier score, isolation forest, one-class support vector machine, principal component analysis, T*-framework and deep learning-based methods, namely a feed forward auto-encoder and a convolutional auto-encoder. A diverse set of methods were chosen because they fall under different categories of anomaly detection techniques and provides a better overview of the performance of detectors with different properties and requirements.

Data under different fault modes was also collected. The failure modes for which data was collected under include, loose belt tension level 1, loose belt tension level 2, tight belt tension, high temperature and friction. For

test purposes the high temperature and loose belt tension level 1 were mixed with an unused normal condition dataset. These specific fault modes were selected deliberately, because loose level 1 fault is a very subtle fault and so it is considered a good candidate for detecting the onset of failure in the robot arm. High temperature on the other hand is at the higher end of the subtle change spectrum and provides a good indication of how the detector performs under different incipient faults. The data were mixed sequentially, such that the dataset starts with the normal condition samples and then the loose level 1 and finally the high temperature. This allows for easier visual comparison while plotting the anomaly scores on a scatter plot.

To evaluate the performance of the anomaly detectors, a number of metrics including the precision, recall, F1-score, AUROC, training and testing time for each detector was computed and compared. A method with higher precision and recall values is considered a superior detector.

An initial study, with the raw torque measurements scaled to a range between 0 and 1 was conducted. The scaling is performed to cope with the requirements of the aforementioned anomaly detection algorithms. Each algorithm was trained or fitted using the normal condition only torque data assuming at one time a 5% and at another a 10% fault contamination. The anomaly detection algorithms output anomaly score per input sample. The contamination ratio together with the anomaly scores were used to determine the threshold required to make a decision on a sample observation being normal or abnormal. Once the threshold is determined, the learned or fitted detectors were fed with the mixed normal, loose level 1 and high temperature test dataset. The anomaly scores per detector were then shown on scatter plots with the threshold marking the boundary between normal and anomalous observations. For a more quantitative comparison, the evaluation metrics per algorithm was collected. A high performing detector should result in a very high precision and recall measures. Values closer to 1.0 indicate superior detection performance. When assumed 10% of normal training data were anomalous, the precision which is a measure indicating real anomalies ranged between 0.83 and 0.85 with majority at 0.85. The recall which is a measure

of how all anomalous data are detected were measured between 0.97 and 1. To find a balance between the precision and recall the F1-score is used. The F1-score ranged between a minimum of 90.7% and maximum of 92.2%. The same experiment performed assuming 95% of training dataset being normal, resulted in higher precision score (0.907 - 0.927) which was what it was expected because fewer of the normal test dataset are considered anomalous compared to that of previous experiment. As a consequence, the recall values slightly decreased with minimum at 0.8235 and 0.8541 for OCSVM and FFAE respectively and 0.90 and 1.0 for the remaining detectors. The F1-scores for OCSVM and FFAE are again the lowest values (86.03, 88.08) and for other detectors they range between 90.05 and 96.11. We think that the lower values for OCSVM and FFAE may be improved with further fine tuning of the parameters associated with each algorithm.

Moreover, the results are averages of the 5 fold cross validations and a poor performing model contributes immensely to the lower values. We suggest using more training data for fitting the models so a variety of normal operating conditions are learned by the detectors and that we would not need to compromise between training and testing datasets sizes.

Looking at the area under the curve for both thresholds reveals a very high discrimination measure which means the models are capable of correctly classifying normal and abnormal samples. The lowest AUC corresponds to OCSVM at 95% threshold and above 90.4% with a maximum of 97.79% for k-NN at the 95% threshold.

In addition to the direct comparison of model's skills using the previously mentioned measures, to show that the model's skill or performance are not statistical fluke, we performed statistical significance tests, namely the one-way ANOVA and McNemar's test. The results (Figure 4.11) show that the performance differences between the models are not statistically significant and so the null hypothesis is not rejected.

The detectors were further compared on the basis of time to train and test and space complexity. Despite the fact that their performance in terms of precision is statistically equivalent the time to train on our robot arm dataset

ranged from 0.02 secs (PCA) to 33.43 secs (ABOD) and the space complexity ranged from 106KB (HBOS) to 27.66MB (T*Framework with iForest).

The fastest of all algorithms was the HBOS algorithm (1.17 secs training time - 0.07 secs test time) and this is because anomaly scores in HBOS are computed using binning and it is assumed that features are independent of each other. This in turn increases processing speed and makes HBOS a good candidate for near real-time large-scale applications. The nearest neighbour-based algorithms all have slower training time due to the complexity of algorithms $O(n^2)$ requiring the computation of nearest neighbours. As for OCSVM, many factors determine the speed of the algorithm, among which are the number of support vectors depending on the dataset size. Hence as data size increases a considerable amount of time is required. For our robot data set with 2000 data points the OCSVM performed the poorest (180 secs $<$ t $<$ 198 secs) among others. One thing to consider is the amount of data being trained or tested. At test time the algorithms had proportional results according to the size of the test dataset with HBOS the fastest amongst all with 0.07 secs. The T*-framework was also applied to the raw features with a dimensionality of 2000, however this resulted in 1,998,200 subspaces which is intractable and would require tremendous computing resources.

In order to reduce the dimensionality of the dataset and minimize the storage needs and increase the speed of training and testing of the models, we further investigated the construction of features that can represent the time series torque signal. As an initial step some descriptive statistics (mean, median, max, min, variance, kurtosis, skewness and peak to peak) were used to describe our dataset. We also tried the tsfresh automated features using 100s of algorithms to extract further features. This manual feature construction reduced the dimensionality of the original dataset from 2000 down to 8 (approximately 200 times), which realized a substantial decrease in dataset size and memory requirements, with comparable results in model performance but less computation time. This emphasizes the fact that proper feature engineering favourably contributes to a model's performance and the correct features enable the application of alternative methods. As a result, the T*-framework

was applied using the handcrafted features and the results yield a comparable performance. The tsfresh features (76 features) were also tested, however no significant increase was seen, denoting the fact that more features does not necessarily mean better performance and feature selection process should be considered to discard irrelevant features.

In another attempt and due to significant reduction in features, we re-implemented (originally in java and only for 2-dimensional data) the reference-based outlier detection algorithm to work with higher than 2-dimensional dataset and evaluated its performance on a dataset other than 2D synthetic data used by Pei et al., [62]. Because the algorithm requires the the appropriate placement of reference points we decided to further reduce the dimensions of the data and used the first 3 principal components using PCA applied to the manual features. This enabled us to visualize the dataset and carefully pick and choose a number of different references with varying locations in the 3D space. The algorithm was then tried on the normal only dataset with 27 reference points and the results were promising both in terms of computation time and detection of anomalies. Like the other algorithms, we tried a mix of normal and abnormal data with the reference based algorithm and results confirm that the method is very sensitive to the number of reference points and how they are placed in the space. If reference points are placed near each other, the algorithm performs poorly because the judgment of a point being anomalous is seen through the eyes of the reference points with the same point of view. Moreover, when we tried placing the reference points at locations so that the dataset was surrounded by these points, the algorithm was unable to detect all of the anomalies. We tried decreasing and increasing the reference points and observed that indeed the placement and number of reference points have a very high impact on how the algorithm performed. Because of this sensitivity, using the algorithm with higher dimensions was not feasible. A visual 3D graph of our experiment with the reference based anomaly detection is shown in the Appendix C: Results of reference-based outlier detection (see Figure. C.1).

Moreover, due to the rise of deep learning and the many advantages such

as automatic feature learning, we implemented a CNN auto-encoder and used the bottleneck layer to extract the learned features. A total of 10 features were fed to the same anomaly detectors and results were collected. The detectors performance may be considered as acceptable for many of the detectors since the F1-scores were above 80% and only HBOS and OCSVM were considered less performing (approximately 72% & 74% respectively). The results indicate that although the CNNAE performs well when using its learned features, but the features learned may not necessarily perform as well with other detectors. To increase the effectiveness of the features learned by the CNNAE, a variety of network architectures were tried, however since the use of deep learning for feature extraction is a recent trend in the research field, further study is required.

Additionally, as found from the review of literature, applying only a single anomaly detector is not considered the best practice. Depending on the dataset several anomaly detection techniques should be tested. Another factor to consider when choosing an anomaly detection method is the time required for training and prediction. Some domain demand an online detector while others can compensate longer training times for faster and more accurate prediction time. Moreover, a major challenge with data-driven techniques such as anomaly detection for condition-based management and the PHM research field is the lack of transparency and interpretability. Well constructed features may help towards more interpretable models. Some of the open problems with regards to detecting the onset of failure is the selection of the right threshold or cut-off point for determining whether an observation is normal or anomalous. Current methods use domain knowledge combined with expertise for setting such threshold, which may be replaced by a more reliable and automated procedure.

In conclusion we believe that the results of our comparative assessment of anomaly detection algorithms for the goal of detecting onset of failures in machinery equipment has a significant potential contribution for the PHM and CBM research field.

5.2 Future directions

We have designed and commissioned a robot arm platform that is capable of producing a variety of failures. As future work, the failure types can be further extended and normal and abnormal data can be collected over a longer period of time. This allows for training the detectors with higher variations of normal operating conditions which may lead to an increase in performance.

In addition to generating different faults for the purpose of evaluating the anomaly detection algorithms, the failure types could be collected simultaneously and while the platform is operating. This provides a different test and evaluation of the anomaly detectors.

In this research, we have looked at a variety of anomaly detection algorithms, in future other techniques could be applied to the same data and compared to that of already tested in this study. For example, anomaly detection using reinforcement learning may be an interesting research study. A recent paper by Gunther et al., [78], introduce the concept of surprise and use General Value Functions (GVF) for learning and representing a model of a robot arm. The model consists of three types of signals where one is the predictions of surprise. In their experiment a robot arm is repeatedly disturbed in a manual way and the these recurring disturbances are seen as peaks in the surprise signal. This notion of surprise can be mapped to the detection of anomalous behaviour (faults) which could be an interesting idea to pursue with the robot arm platform.

An additional study that may be interesting to investigate is the comparison of the physics-based model of the robot arm with the machine learning models applied in our study. Last but not least, we have looked at the onset of failures in an equipment, which is only a part of the whole PHM framework. A potential next phase could be detecting failure types and determining time to failure.

References

[1] R. S. Aboul-Yazeed, A. El-Bialy, and A. S. Mohamed, "Prediction of medical equipment failure rate: A case study," in *International Conference on Advanced Intelligent Systems and Informatics*, Springer, 2016, pp. 650–659. 16

[2] C. C. Aggarwal, "Outlier analysis," in *Data mining*, Springer, 2015, pp. 237–263. 5

[3] G. O. Allgood and B. R. Upadhyaya, "Model-based high-frequency matched filter arcing diagnostic system based on principal component analysis (pca) clustering," in *Applications and Science of Computational Intelligence III*, International Society for Optics and Photonics, vol. 4055, 2000, pp. 430–441. 31

[4] H. J. Ashton H. (2017). Idc manufacturing insights, [Online]. Available: `http://www.idc.com` (visited on 09/12/2018). 1

[5] P. Baraldi, F. Di Maio, M. Rigamonti, E. Zio, and R. Seraoui, "Transients analysis of a nuclear power plant component for fault diagnosis," *Chemical Engineering Transactions*, vol. 33, pp. 895–900, 2013. 20

[6] P. Baraldi, F. Di Maio, and E. Zio, "Unsupervised clustering for fault diagnosis," in *Prognostics and System Health Management (PHM), 2012 IEEE Conference on*, IEEE, 2012, pp. 1–9. 20

[7] W. Bartelmus, F. Chaari, R. Zimroz, and M. Haddar, "Modelling of gearbox dynamics under time-varying nonstationary load for distributed fault detection and diagnosis," *European Journal of Mechanics-A/Solids*, vol. 29, no. 4, pp. 637–646, 2010. 10

[8] A. M. Bianco, M. Garcia Ben, E. Martinez, and V. J. Yohai, "Outlier detection in regression models with arima errors using robust estimates," *Journal of Forecasting*, vol. 20, no. 8, pp. 565–579, 2001. 16

[9] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015. 16

[10] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *ACM sigmod record*, ACM, vol. 29, 2000, pp. 93–104. 25

[11] T. Brotherton, G. Jahns, J. Jacobs, and D. Wroblewski, "Prognosis of faults in gas turbine engines," in *Aerospace Conference Proceedings, 2000 IEEE*, IEEE, vol. 6, 2000, pp. 163–171. 11

[12] C. Cecati, "A survey of fault diagnosis and fault-tolerant techniques—part ii: Fault diagnosis with knowledge-based and hybrid/active approaches," *IEEE Transactions on Industrial Electronics*, 2015. 5

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009. 12, 14, 16, 28, 30

[14] Z. Chen, C. Li, and R.-V. Sanchez, "Gearbox fault identification and classification with convolutional neural networks," *Shock and Vibration*, vol. 2015, 2015. 68

[15] S. Cheng, M. H. Azarian, and M. G. Pecht, "Sensor systems for prognostics and health management," *Sensors*, vol. 10, no. 6, pp. 5774–5797, 2010. 3

[16] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, 2018. 56

[17] M. Christ, A. W. Kempa-Liehr, and M. Feindt, "Distributed and parallel time series feature extraction for industrial big data applications," *arXiv preprint arXiv:1610.07717*, 2016. 56

[18] D. A. Clifton, P. R. Bannister, and L. Tarassenko, "A framework for novelty detection in jet engine vibration data," in *Key engineering materials*, Trans Tech Publ, vol. 347, 2007, pp. 305–310. 19

[19] S. Das, B. L. Matthews, and R. Lawrence, "Fleet level anomaly detection of aviation safety data," in *Prognostics and health management (phm), 2011 ieee conference on*, IEEE, 2011, pp. 1–10. 30

[20] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013. 56

[21] I. Diaz and J. Hollmén, "Residual generation and visualization for understanding novel process conditions," in *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, IEEE, vol. 3, 2002, pp. 2070–2075. 30

[22] M. Du, L. B. Tjernberg, S. Ma, Q. He, L. Cheng, and J. Guo, "A som based anomaly detection method for wind turbines health management through scada data," *International Journal of Prognostics and Health Management*, vol. 7, pp. 1–13, 2016. 21

[23] J. Fan, C. Qian, X. Fan, G. Zhang, and M. Pecht, "In-situ monitoring and anomaly detection for led packages using a mahalanobis distance approach," in *Reliability Systems Engineering (ICRSE), 2015 First International Conference on*, IEEE, 2015, pp. 1–6. 24

[24] A. Foss, O. R. Zaıane, and S. Zilles, "Unsupervised class separation of multivariate data through cumulative variance-based ranking," in *2009 Ninth IEEE International Conference on Data Mining*, IEEE, 2009, pp. 139–148. 32

[25] B. D. Fulcher, M. A. Little, and N. S. Jones, "Highly comparative time-series analysis: The empirical structure of time series and their methods," *Journal of the Royal Society Interface*, vol. 10, no. 83, p. 20 130 048, 2013. 55

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587. 68

[27] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," *KI-2012: Poster and Demo Track*, pp. 59–63, 2012. 17

[28] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS one*, vol. 11, no. 4, e0152173, 2016. 24

[29] X. Guo, L. Chen, and C. Shen, "Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis," *Measurement*, vol. 93, pp. 490–502, 2016. 68

[30] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *International Conference on Data Warehousing and Knowledge Discovery*, Springer, 2002, pp. 170–180. 30

[31] Q. He, R. Yan, F. Kong, and R. Du, "Machine condition monitoring using principal component representations," *Mechanical Systems and Signal Processing*, vol. 23, no. 2, pp. 446–466, 2009. 32

[32] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-d convolutional neural networks.," *IEEE Trans. Industrial Electronics*, vol. 63, no. 11, pp. 7067–7075, 2016. 39

[33] R. Isermann, "Model-based fault-detection and diagnosis–status and applications," *Annual Reviews in control*, vol. 29, no. 1, pp. 71–85, 2005. 10

[34] V. M. Janakiraman and D. Nielsen, "Anomaly detection in aviation data using extreme learning machines," in *Neural Networks (IJCNN), 2016 International Joint Conference on*, IEEE, 2016, pp. 1993–2000. 36

[35] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, R. Van de Walle, and S. Van Hoecke, "Convolutional neural network based fault detection for rotating machinery," *Journal of Sound and Vibration*, vol. 377, pp. 331–345, 2016. 39

[36] A. K. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical systems and signal processing*, vol. 20, no. 7, pp. 1483–1510, 2006. 4

[37] X. Jin and T. W. Chow, "Anomaly detection of cooling fan and fault classification of induction motor using mahalanobis–taguchi system," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5787–5795, 2013. 23

[38] X. Jin, Y. Sun, Z. Que, Y. Wang, and T. W. Chow, "Anomaly detection and fault prognosis for bearings," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 9, pp. 2046–2054, 2016. 23

[39] L. Jing, M. Zhao, P. Li, and X. Xu, "A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox," *Measurement*, vol. 111, pp. 1–10, 2017. 69

[40] N. Jones, "Top 10 iot technologies for 2017 and 2018," *Gartner, Inc., Tech. Rep*, 2016. 33

[41] N. Jones, "Computer science: The learning machines," *Nature News*, vol. 505, no. 7482, p. 146, 2014. 62

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 67

[43] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990. 20

[44] H.-P. Kriegel, A. Zimek, *et al.*, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2008, pp. 444–452. 27, 28

[45] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–44, 2015. 68

[46] B. Li, P.-l. Zhang, H. Tian, S.-S. Mi, D.-S. Liu, and G.-Q. Ren, "A new feature extraction and selection scheme for hybrid fault diagnosis of gearbox," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 000–10 009, 2011. 58

[47] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, IEEE, 2008, pp. 413–422. 33, 74

[48] C. Lu, Z.-Y. Wang, W.-L. Qin, and J. Ma, "Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification," *Signal Processing*, vol. 130, pp. 377–388, 2017. 67

[49] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, IEEE, vol. 3, 2003, pp. 1741–1745. 29

[50] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297. 19

[51] G. J. McLachlan and K. E. Basford, *Mixture models: Inference and applications to clustering*. Marcel Dekker, 1988, vol. 84. 14

[52] V. Miranda, A. R. G. Castro, and S. Lima, "Diagnosing faults in power transformers with autoassociative neural networks and mean shift," *IEEE Transactions on Power Delivery*, vol. 27, no. 3, pp. 1350–1357, 2012. 36

[53] F. Mörchen, *Time series feature extraction for data mining using dwt and dft*, 2003. 56

[54] M. M. Moya, M. W. Koch, and L. D. Hostetler, "One-class classifier networks for target recognition applications," *NASA STI/Recon Technical Report N*, vol. 93, 1993. 13

[55] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," *International Journal of Computer Research*, vol. 10, no. 3, pp. 49–61, 2001. 56

[56] P. Nectoux, R. Gouriveau, K. Medjaher, emmanuel. ramasso, B. C. Morello, and N. Zerhouni, "Pronostia : An experimental platform for bearings accelerated degradation tests," 2012. 24

[57] R. E. News. (2017). 4 ways predictive maintenance streamlines manufacturing, [Online]. Available: `http://www.reuters.com/media-campaign/brandfeatures/intel/gated/TCM_1610337_intel_adlink_IOT_whitepaper_R6.pdf`. 1

[58] V. R. Patel and R. G. Mehta, "Impact of outlier removal and normalization approach in modified k-means clustering algorithm," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 5, p. 331, 2011. 54

[59] M. Pecht, "Prognostics and health management of electronics wiley-intersicence," *New York, NY*, 2008. 3, 4, 23

[60] M. Pecht, "Prognostics and health management of electronics," *Encyclopedia of Structural Health Monitoring*, 2009. 54

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. 54

[62] Y. Pei, O. R. Zaiane, and Y. Gao, "An efficient reference-based approach to outlier detection in large datasets," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*, IEEE, 2006, pp. 478–487. 26, 96

[63] W. Peizhuang, "Pattern recognition with fuzzy objective function algorithms (james c. bezdek)," *SIAM Review*, vol. 25, no. 3, p. 442, 1983. 19

[64] T. Plante, L. Stanley, A. Nejadpak, and C. X. Yang, "Rotating machine fault detection using principal component analysis of vibration signal," in *IEEE AUTOTESTCON, 2016*, IEEE, 2016, pp. 1–7. 31

[65] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM Sigmod Record*, ACM, vol. 29, 2000, pp. 427–438. 24

[66] D. Reynolds, "Gaussian mixture models," *Encyclopedia of biometrics*, pp. 827–832, 2015. 15

[67] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. John wiley & sons, 2005, vol. 589. 16

[68] M. A. Salama, A. E. Hassanien, and A. A. Fahmy, "Reducing the influence of normalization on data classification," in *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on*, IEEE, 2010, pp. 609–613. 53

[69] A. Shaheryar, Y. Xu-Cheng, and W. Y. Ramay, "Robust feature extraction on vibration data under deep-learning framework: An application for fault identification in rotary machines," *International Journal of Computer Applications*, vol. 167, no. 4, 2017. 69

[70] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL and COMPUTER ENGINEERING, Tech. Rep., 2003. 31

[71] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959. 68

[72] C. Stellman, K. Ewing, F. Bucholtz, and I. Aggarwal, "Monitoring the degradation of a synthetic lubricant oil using infrared absorption, fluorescence emission and multivariate analysis: A feasibilty study," *Tribology & Lubrication Technology*, vol. 55, no. 10, p. 42, 1999. 31

[73] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, and X. Chen, "A sparse auto-encoder-based deep neural network approach for induction motor faults classification," *Measurement*, vol. 89, pp. 171–178, 2016. 67

[74] S. Tao, T. Zhang, J. Yang, X. Wang, and W. Lu, "Bearing fault diagnosis method based on stacked autoencoder and softmax regression," in *Control Conference (CCC), 2015 34th Chinese*, IEEE, 2015, pp. 6331–6335. 67

[75] J. Tian, M. H. Azarian, and M. Pecht, "Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm," in *Proceedings of the European Conference of the Prognostics and Health Management Society*, Citeseer, 2014. 20

[76] P. Y. Tu, R. Yam, P. Tse, and A. Sun, "An integrated maintenance management system for an advanced manufacturing company," *The International Journal of Advanced Manufacturing Technology*, vol. 17, no. 9, pp. 692–703, 2001. 11

[77] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung, "Convolutional networks can learn to generate affinity graphs for image segmentation," *Neural computation*, vol. 22, no. 2, pp. 511–538, 2010. 68

[78] J. G. unther, A. Kearney, M. R. Dawson, C. Sherstan, and P. M. Pilarski, "Predictions , surprise , and predictions of surprise in general value function architectures," 2018. 98

[79] G. Wang, C. Liu, and Y. Cui, "Clustering diagnosis of rolling element bearing fault based on integrated autoregressive/autoregressive conditional heteroscedasticity model," *Journal of Sound and Vibration*, vol. 331, no. 19, pp. 4379–4387, 2012. 19

[80] Y. Wang, Q. Miao, E. W. Ma, K.-L. Tsui, and M. G. Pecht, "Online anomaly detection for hard disk drives based on mahalanobis distance," *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 136–145, 2013. 24

[81] Y. Wang, K. Tsui, E. W. Ma, and M. Pecht, "A fusion approach for anomaly detection in hard disk drives," in *Prognostics and System Health Management (PHM), 2012 IEEE Conference on*, IEEE, 2012, pp. 1–5. 24

[82] W. Weber, J. Jungjohann, and H. Schulte, "Physically-based modeling of speed sensors for fault diagnosis and fault tolerant control in wind turbines," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 570, 2014, p. 072 008. 10

[83] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu, "A comparative study of rnn for outlier detection in data mining," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, IEEE, 2002, pp. 709–712. 30

[84] L. Xiong, H.-D. Ma, H.-Z. Fang, K.-X. Zou, and D.-W. Yi, "Anomaly detection of spacecraft based on least squares support vector machine," in *Prognostics and System Health Management Conference (PHM-Shenzhen), 2011*, IEEE, 2011, pp. 1–6. 30

[85] T. Yairi, Y. Kato, and K. Hori, "Fault detection by mining association rules from house-keeping data," in *proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Citeseer, vol. 18, 2001, p. 21. 11

[86] W. Yan and L. Yu, "On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach," in *Proceedings of the annual conference of the prognostics and health management society*, 2015. 37

[87] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, "A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals," *Sensors*, vol. 17, no. 2, p. 425, 2017. 69, 70

[88] Z. Zhang and A. Kusiak, "Monitoring wind turbine vibration based on scada data," *Journal of Solar Energy Engineering*, vol. 134, no. 2, p. 021 004, 2012. 19

[89] J. Zhao, L. Xu, and L. Liu, "Equipment fault forecasting based on arma model," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, IEEE, 2007, pp. 3514–3518. 16

[90] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring: A survey," *arXiv preprint arXiv:1612.07640*, 2016. 33, 34

[91] F. Zorriassatine, A. Al-Habaibeh, R. Parkin, M. Jackson, and J. Coy, "Novelty detection for practical pattern recognition in condition monitoring of multivariate processes: A case study," *The International Journal of Advanced Manufacturing Technology*, vol. 25, no. 9-10, pp. 954–963, 2005. 15

Appendix A

Plots of anomaly scores for T*-framework

Figure A.1 shows the results of using the T*-framework with different anomaly detection methods. As mentioned in Chapter 2, the T*-framework can use any outlier detection algorithm within the framework.

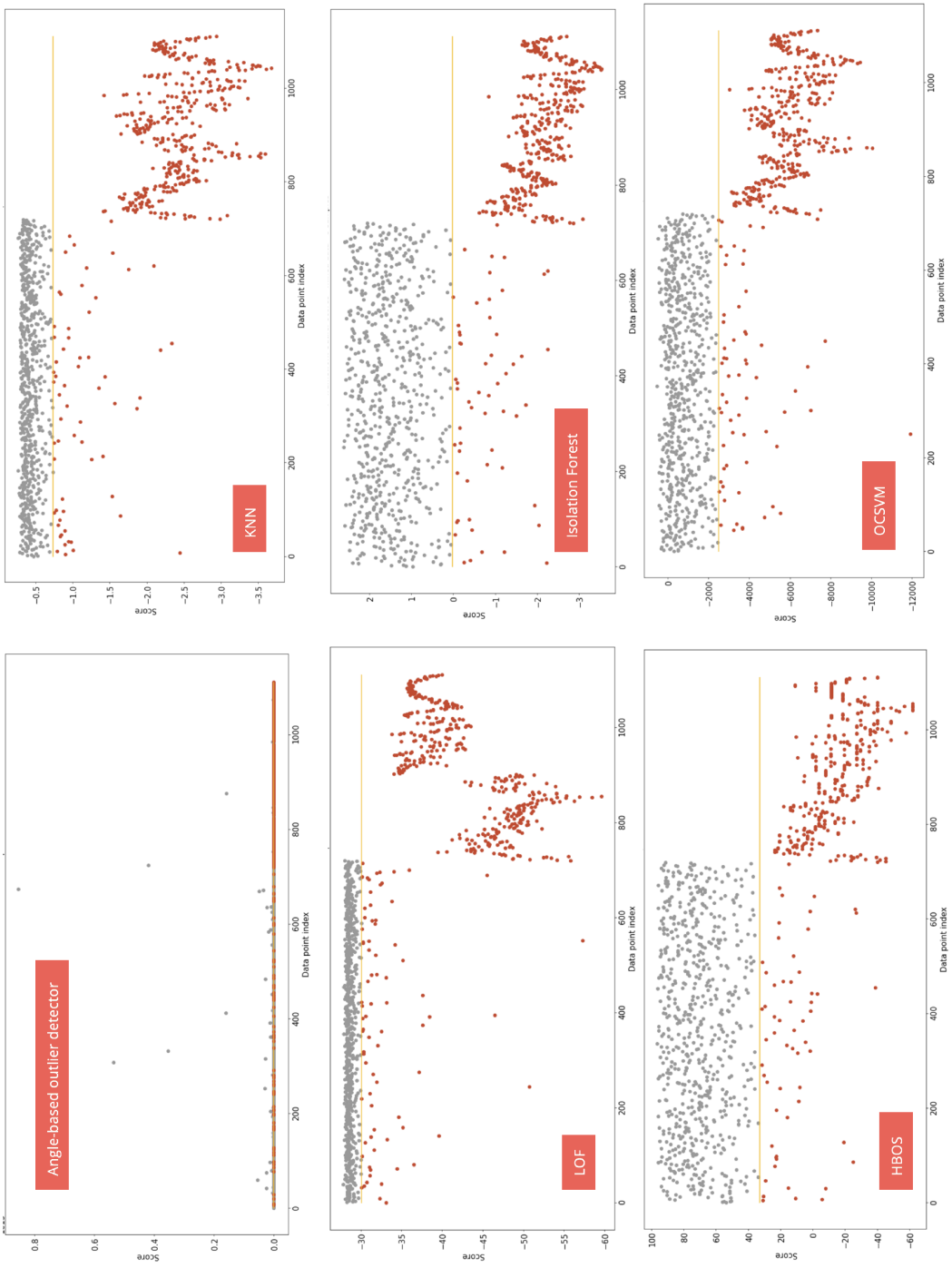

Figure A.1: T*-framework (8 features) anomaly scores for the different outlier detection algorithms assuming 10% of normal training data contain anomalies.

Appendix B

Plots of anomaly scores with tsfresh features

Figure B.1 shows the results of applying the suggested anomaly detection algorithms with the 76 features extracted using the tsfresh package. As mentioned in Chapter 4, section 4.1.3, under feature-based representation of time series, tsfresh is a python library that applies numerous algorithms to a time series data and extracts 100s of different features.

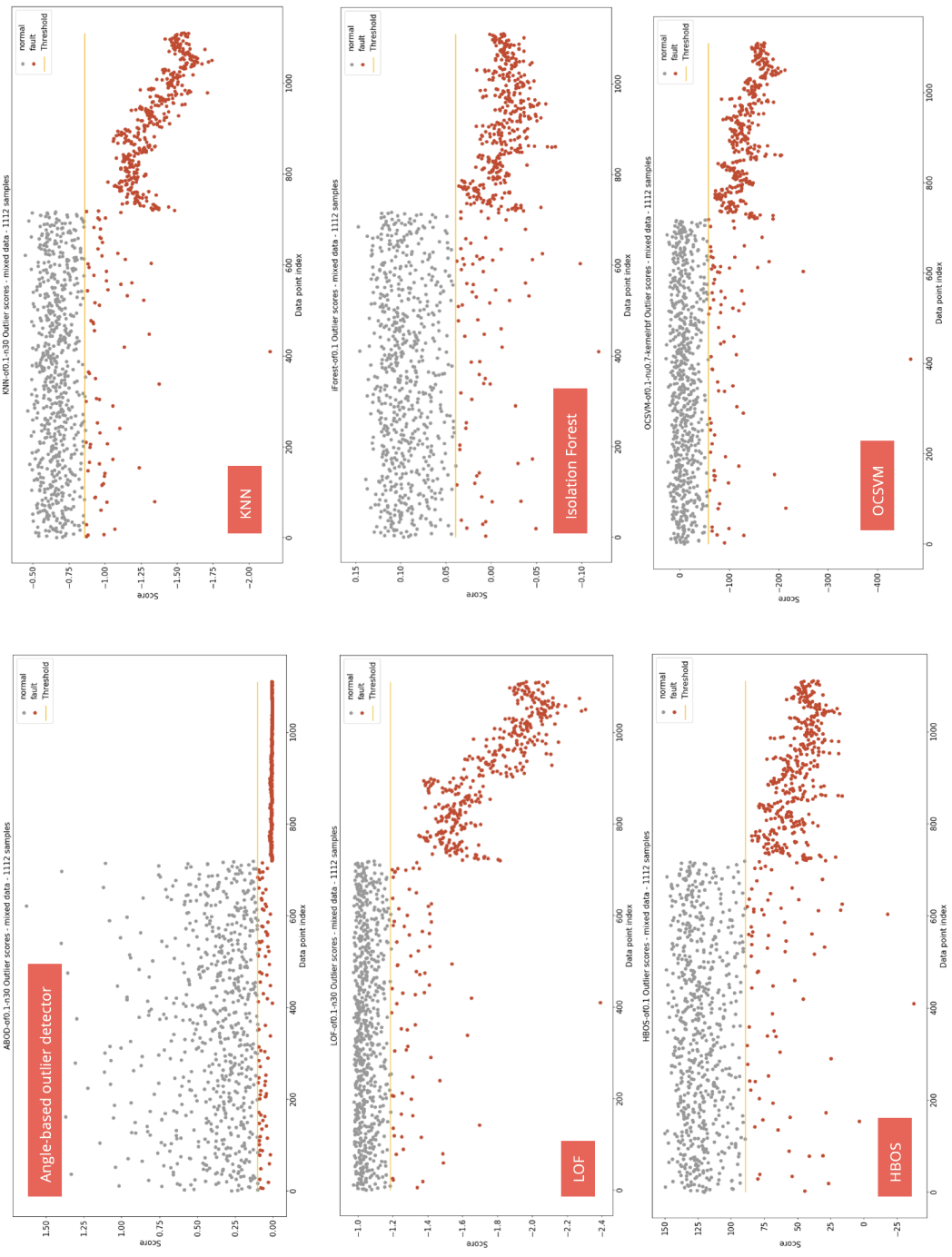

Figure B.1: tsfresh (76 features) anomaly scores for the different outlier detection algorithms assuming 10% of normal training data contain anomalies.

Appendix C

Results of reference-based outlier detection

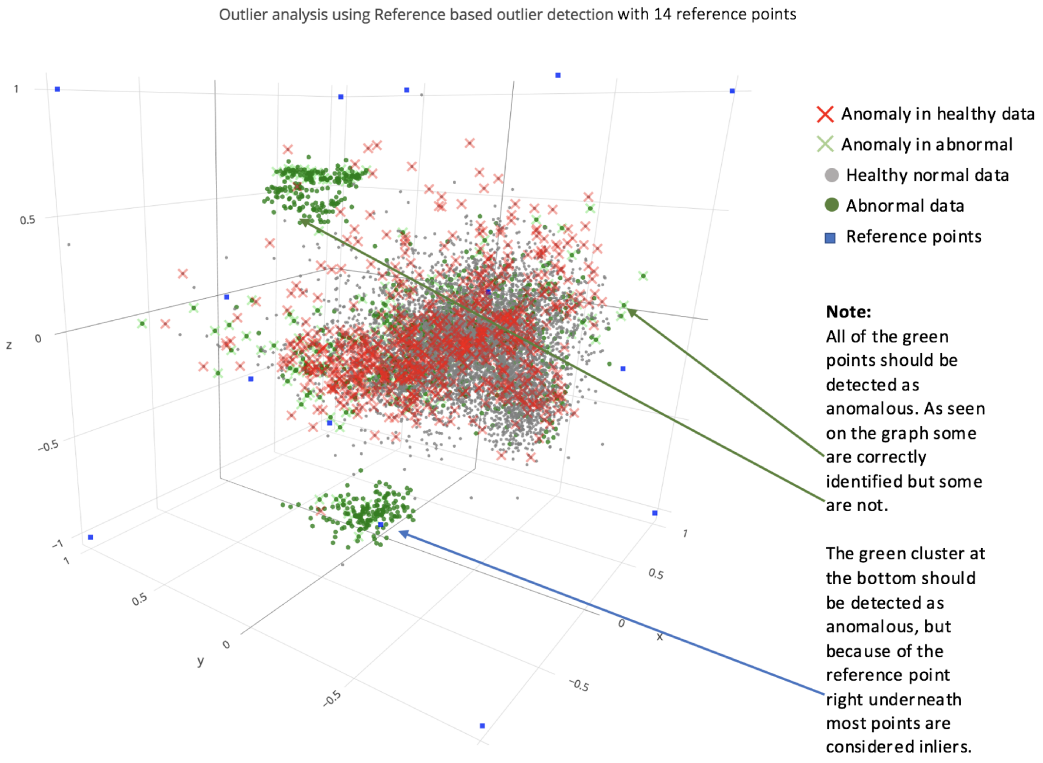

Figure C.1: Reference-based outlier detection algorithm with robot arm data using 14 reference points.

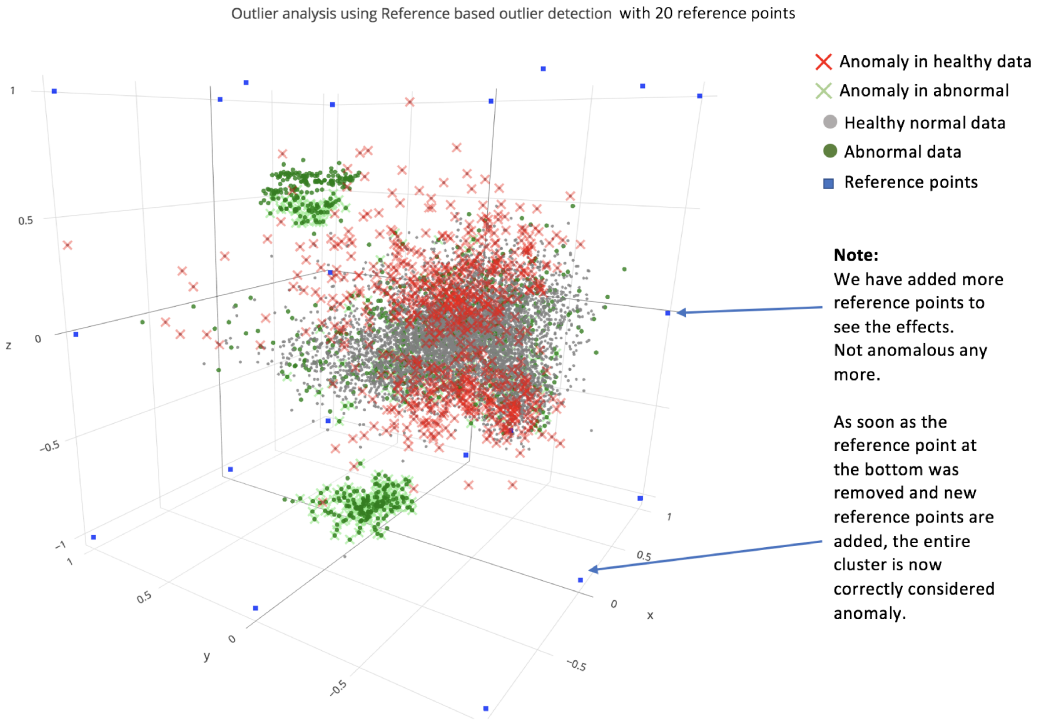

Figure C.2: Reference-based outlier detection algorithm with robot arm data using 20 reference points.