Faculty of Engineering, Built Environment and Information Technology

Utilizing available time-varying operating condition information in learning-based methods for fault diagnostics of rotating machinery

AH Nortjé
17181471

Submitted in partial fulfilment of the requirements for the degree
Master of Engineering (Mechanical Engineering)

Department of Mechanical and Aeronautical Engineering
Faculty of Engineering, Built Environment and Information Technology

Supervised by Prof. PS Heyns and Prof. DN Wilke

University of Pretoria

2022

Abstract

Utilizing available time-varying operating condition information in learning-based methods for fault diagnostics of rotating machinery

Author:	Abel Hermanus Nortjé
Supervisors:	Prof. PS Heyns, Prof. DN Wilke
Department:	Department of Mechanical and Aeronautical Engineering
University:	University of Pretoria
Keywords:	Condition monitoring, deep learning, principal component analysis, variational auto-encoders, discrepancy analysis, fault detection and tracking, gear and bearing faults, time-varying operating conditions

Historical failure datasets for critical assets are rarely available. This makes it difficult to integrate condition monitoring techniques found in literature into industry, as these techniques are often either equipment-specific or highly dependent on the availability of a historical failure database. Recent research in the vibration-based condition monitoring field addressed this problem, by focusing on using unsupervised latent variable models that require only healthy data for training. By learning the distribution of the healthy data, faults can be detected and tracked when new data starts to deviate from the learned healthy distribution.

The complexity of the healthy data distribution drastically increases when machines are operated at varying rotational speeds, resulting in smearing in the frequency spectrum, amplitude modulation, and heteroscedastic noise in the data. This makes it more difficult to accurately model the healthy data distribution. Signal processing methods that incorporate available shaft speed and phase information, have been applied extensively to vibration data from varying speed conditions, to make it easier to analyse. Order tracking has been performed to convert the signals to the angle domain, and regression methods have been applied to normalize the effect of amplitude modulation. This work systematically investigates to what extent the availability of operating condition information can help to simplify the learning process for latent variable models in a semi-supervised setting. For a baseline, the operating conditions are mapped to the vibration data in a supervised setting. This is done to see how much of the variance in the vibration data can be explained by the operating conditions, and to highlight the importance of using latent variable models when the data is influenced by generative factors that cannot easily be measured or extracted from the data. Unsupervised models (that take raw unprocessed vibration data as inputs) are compared with semi-supervised models that incorporate operating condition information during data pre-processing (order tracking), and during modelling (learning distributions conditioned on the associated operating conditions).

Two latent variable frameworks that are often used for anomaly detection are investigated: Principal Component Analysis (PCA) and Variational Auto-Encoders (VAEs). This work highlights practical limitations in the PCA and VAE frameworks for condition monitoring in an unsupervised setting. It is shown that PCA can't accurately capture the heteroscedastic noise in the data, since it is formulated by assuming constant variance across the whole dataset. It is proposed that the correct variance can be learned using linear regression between either the latent space representations (unsupervised) or the available operating condition information (semi-supervised). Using the available operating condition information yielded the most robust results. For the VAE framework, it is shown that the

latent space prior commonly used to facilitate disentanglement in β-VAEs, can in this case lead to posterior collapse, which leads to poor discrepancy analysis results. This occurs because of circular structures present in the underlying latent space manifold, caused by the periodic nature of vibration signals. The isotropic variance Gaussian usually used as a latent space prior is not well suited to capture these manifolds. This challenge is overcome by conditioning the prior on the associated operating condition information.

This work also provides clear insights into how linear and nonlinear models capture these distributions differently with lower and higher dimensional latent spaces, which improves the interpretability of the latent space and the associated model performance. It is shown how the size of the latent space affects whether the damage is detected in the latent space or the reconstruction space. In addition, the latent space representations can successfully be monitored for anomalies when conditioned on the available operating condition information. The latent variable models' performances are compared to traditional signal processing approaches (envelope analysis). The semi-supervised models return promising results on a dataset that is known to be difficult to analyse with traditional signal processing methods.

Acknowledgements

I would like to thank:

- Prof. Stephan Heyns, Prof. Nico Wilke and Mr. George Breitenbach, for their continuous guidance, advice and motivation throughout my research.
- ZZ2, for funding this research.
- Thys, Esmi, Albertus and Jeandre, for making sure that I got an adequate amount of coffee and sunlight while writing this dissertation.

Table of Contents

List of Abbreviations

Adam	Adaptive moment estimation
AE	Auto-encoder
AUC	Area-under-the-curve
BPFI	Ball pass frequency inner race
BPFO	Ball pass frequency outer race
BSF	Ball spin frequency
CBM	Condition-based maintenance
CCR	Cumulative Contribution Rate
CNN	Convolutional Neural Network
C-AIM	Centre for Asset Integrity Management
COT	Computed Order Tracking
DDL	Damage Detection Level
DFT	Discrete Fourier Transform
ELBO	Evidence Lower Bound
FFT	Fast Fourier Transform
FTF	Fundamental Train Frequency
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
ICA	Independent Component Analysis
iVAE	Identifiable Variational Auto-encoder
KL	Kullback-Leibler
LVM	Latent Variable Model
MSE	Mean Squared Error
NAMVOC	Normalisation of the Amplitude Modulation caused by Varying Operating Conditions
NES	Normalized squared-magnitude of the squared Envelope Spectrum
OC	Operating Condition
OOD	Out-of-distribution
OST	Order Spectra Tracking
OT	Order Tracking
OTA	Order Tracking and Aligning
PCA	Principal Component Analysis
PPCA	Probabilistic Principal Component Analysis
PDF	Probability Density Function
PHM	Prognostics and Health Management
RMS	Root Mean Square
RPM	Revolutions Per Minute
RUL	Remaining Useful Life
SAT	Synchronous Average Tracking
SES	Squared Envelope Spectrum
SNR	Signal-to-noise ratio
SO	Section Only
TPR	True Positive Rate
VAE	Variational Auto-encoder

1 Introduction

1.1 Background

For companies in the power generation, mining or manufacturing industries to remain competitive, their critical assets must be as reliable as possible. Assets deteriorate over time, and without intervention, this will lead to failures. To get the maximum life and performance out of critical assets, maintenance must be performed when components start to degrade. To minimize the costs and downtime caused by failures and maintenance programs, condition-based maintenance (CBM) techniques can be implemented. CBM cuts on maintenance costs by monitoring the condition of the asset, and only implementing maintenance when degradation of the asset is detectable (Jardine et al. 2006).

A CBM program can be used to make better maintenance decisions. A maintenance decision addresses a diagnostic question - 'What should be repaired/replaced?', as well as a prognostic question - 'How much time before this repair/replacement should take place?'. Diagnostics is focused on identifying the fault present in the asset, while with prognostics, the aim is to predict the Remaining Useful Life (RUL) of the asset before it fails. The framework of Prognostics and Health Management (PHM) combines diagnostics and prognostics (Lee et al. 2014). The aim is to identify and isolate a fault as early as possible, and then track how the fault develops to continuously update the RUL estimation, and perform maintenance before some failure threshold is reached.

Three different methods of PHM are often implemented: physics-driven methods, data-driven methods, and hybrid methods (a combination of physics- and data-driven methods) (Liao & Köttig 2014).

Physics-driven methods aim to model the failure mechanisms (e.g. crack growth) using mathematical models that describe some underlying physics. Analytical models are used where possible, but for most cases, the physics over complex geometries cannot be represented in an elegant closed-form equation or are only poorly approximated. In these cases, finite element models allow for accurate and flexible results. These models are however computationally expensive, and to get the models to accurately resemble the failure, the model parameters need to be carefully calibrated to match the actual material properties of the equipment. The models are equipment-specific, and it can be difficult to accurately model the physics of machines with many interacting components. In many cases, the underlying physics of the machine is simply not understood well enough to allow appropriate and effective modelling of the physics.

When the underlying physics cannot be modelled accurately, one can turn to data-driven methods. Data-driven methods predict the condition of the asset based on condition monitoring signals obtained through sensors. Different sources of data can be used to gain insight into the condition of a machine. Vibration signals, temperature signals, acoustic signals, oil debris measurements or any other sensor measurement that has some correlation with the underlying condition of the machine, can be used to monitor the condition (Jardine et al. 2006). Vibration-based condition monitoring is however the most common technique, because of the amount of diagnostic information that is captured in the data.

A data-driven PHM methodology should ideally have the following characteristics for it to be easily integrated into industry:

1. **Little expert domain knowledge required**: Many traditional data-driven techniques rely on extensive domain knowledge to extract informative features from the data from which the

condition of the machine can be inferred. Most companies do not have the domain knowledge and resources to do this.

2. **Applicable to a range of different assets**: Many of the data-driven techniques found in literature are equipment-specific, making it difficult for companies that want to monitor a wide range of assets.

3. **No asset-specific historical failure data required**: While there are some labelled failure datasets available that have been used extensively in literature for investigations, historical failure datasets are rarely available for the specific asset of interest that one wants to monitor. For the largest part of the life cycle of an asset, the asset operates in the healthy regime, with the asset only entering the damaged regime near the end of its life cycle. In most cases, the damaged regime is not recorded until failure, as it is costly to run the asset up to failure. Only part of the damage regime is available, up to the point where maintenance is applied. This leads to a large imbalance in the availability of healthy data compared to unhealthy data.

4. **Applicable to time-varying operating conditions**: Many machines are operated at fluctuating speeds and loads, which makes it significantly more challenging to use data-driven methods for condition monitoring.

Learning-based methods (machine learning and deep learning) can be used to automate fault diagnostics and remove the requirement for expert domain knowledge (Lei et al. 2016, Khan & Yairi 2018, Zhang et al. 2020). Recent work has focused on developing unsupervised learning-based methods that do not require failure datasets, where models are trained to capture the distributions of healthy data, and then detect and track how a fault develops, by monitoring how new data samples deviate from the healthy distribution (Heyns et al. 2012, Booyse et al. 2020, Balshaw et al. 2022). These techniques are not tailored to monitor specific assets and fault types more accurately than others, and have been shown to be applicable to time-varying operating conditions. To automatically classify faults, one still requires labelled failure data to train models. Some research has been done on transfer learning techniques, where models can be trained on large labelled failure datasets from other assets, and that knowledge can then be transferred to classify faults on assets where no historical failure datasets are available (Guo et al. 2019).

This work focuses on using latent variable models (LVMs) to capture the distribution of the healthy data, in order to automatically detect and track faults. It is explained how this framework can be integrated with transfer learning techniques to automatically classify faults, but transfer learning techniques are not explored in this dissertation. The complexity of the healthy data distribution drastically increases when machines are operated at varying rotational speeds. This work systematically investigates to what extent the availability of shaft speed and phase information can be used to simplify the learning process for the latent variable models, by integrating the information into the learning process in a semi-supervised setting.

1.2 Data-driven fault diagnostics

In the data-driven condition monitoring context, the primary goal is to use the available raw condition monitoring data samples, denoted here as $\boldsymbol{X} \in \mathbb{R}^{n \times D_{\boldsymbol{x}}}$, to infer information related to the health condition of the machine, denoted here as $\boldsymbol{h} \in \mathbb{R}^{n}$. The number of samples is denoted by n. From each sample $\boldsymbol{x} \in \mathbb{R}^{D_{\boldsymbol{x}}}$, a health condition h is inferred. $\boldsymbol{x}$ would typically be a time-series vibration signal, where the signal length $D_{\boldsymbol{x}}$ is influenced by the sampling frequency and duration. The information contained by h is dependent on the diagnostic task at hand. Diagnostics can be divided into three stages: fault detection, fault classification and fault tracking (Cerrada et al. 2018). Before looking at the three diagnostic stages in detail, we first consider the process of feature extraction.

1.2.1 Feature extraction

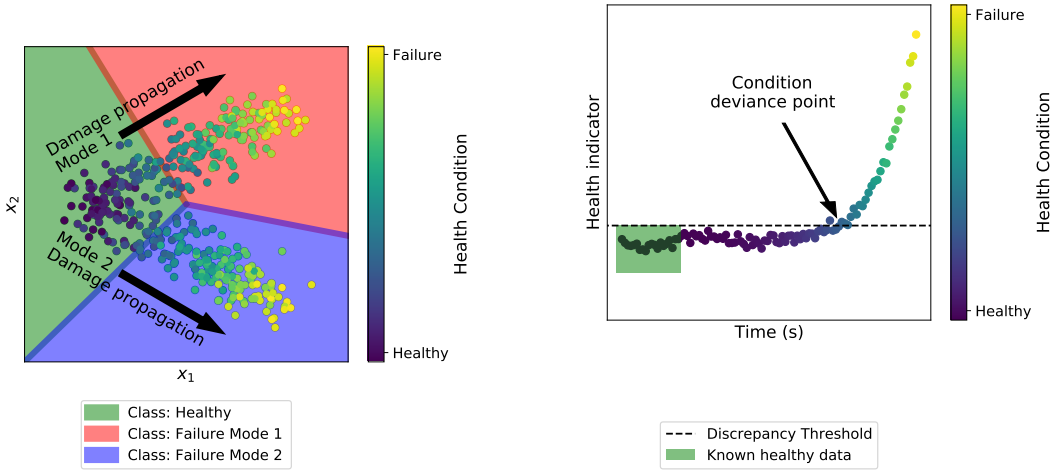
In most cases, each raw condition monitoring signal $\boldsymbol{x}$ contains a lot of information that is not related to the condition of the machine, making it more difficult to infer h directly from it. In these cases, features are extracted from $\boldsymbol{x}$, denoted here as $\boldsymbol{x}^* \in \mathbb{R}^{D_{\boldsymbol{x}^*}}$, to get rid of the information irrelevant to inferring h, and to amplify the relevant information. When features are extracted manually, one needs to know how the information of interest (the health condition of the machine) presents itself in the data $\boldsymbol{x}$, and then carefully select feature extraction methods to extract these characteristics. Traditionally, statistical features such as the root-mean-square (RMS) or kurtosis of a signal were monitored to get indications of damage. These features are however highly sensitive to fluctuating operating conditions (Zimroz et al. 2014), and still contain a lot of information irrelevant to the health condition h. Advanced signal processing methods have been proposed in literature that focus on extracting impulsive cyclostationary components from the signals in frequency bands that are known to contain the fault, using envelope analysis. For a cyclostationary signal, certain statistical features of the signal vary periodically over time, caused by the energy flow in the signal (Antoni 2009). Extracting cyclostationary information from the signals is particularly useful, since bearing faults manifest as periodic impulses in vibration signals. Envelope analysis is typically done by first bandpass filtering the signal in a frequency band that is known to contain the impulsive components, and then demodulating the signal to extract the envelope (Randall & Antoni 2011). The envelope spectrum can then be analysed to detect fault frequencies, with the squared envelope spectrum (SES) being the benchmark technique for bearing diagnostics (Abboud et al. 2019). Sparsity measures have been proposed to automatically identify the most suitable band for demodulation (Antoni 2007). Wang et al. (2020) showed that popular sparsity measures, such as the spectral kurtosis, spectral negative entropy, spectral Gini index and spectral smoothness index can all be formulated as the sum of a weighted normalized square envelope, with the weights being the only difference between the sparsity measures. The work of Abboud et al. (2019) compares some of the state-of-the-art bearing diagnostics approaches. These techniques are however often tailored to specific faults. To completely remove the requirement of expert knowledge from the feature extraction process and make it applicable to any fault type, automatic feature extraction methods have been proposed using unsupervised learning techniques (Lei et al. 2016). For unsupervised learning, the goal is to learn the distribution of the input data $p(\boldsymbol{x})$. When the input data $\boldsymbol{x}$ is high dimensional, the models aim to learn the underlying relationships in the high dimensional input data. The models learn by projecting the high-dimensional input sample $\boldsymbol{x}$ to a lower-dimensional representation, denoted here as $\boldsymbol{z} \in \mathbb{R}^{D_{\boldsymbol{z}}}$, where $D_{\boldsymbol{z}} << D_{\boldsymbol{x}}$, and then try to reconstruct the high-dimensional input data as well as possible from this lower-dimensional representation. By doing this, the models learn to retain the important information about $\boldsymbol{x}$ in this lower-dimensional representation $\boldsymbol{z}$. $\boldsymbol{z}$ can then be used as features $\boldsymbol{x}^*$. Note that $\boldsymbol{z}$ is learned automatically using learning-based methods, while $\boldsymbol{x}^*$ is extracted manually.

Feature extraction has been extensively integrated into each diagnostic stage to simplify the process of inferring $\boldsymbol{h}$. Wang et al. (2018) provided a thorough literature review of vibration-based features used as bearing and gear health indicators.

1.2.2 Fault classification

With fault classification, the goal is to identify the source that causes the anomalous behaviour. It is usually not concerned with estimating the severity of the fault, but only with identifying the particular failure mode. In this case, h will be some class label ('Healthy', 'Damaged gear tooth', 'Damaged bearing inner race', etc.), making it a classification task, or the damage severity (tool wear, crack size, etc.), making it a regression task. To classify faults, the model needs to know what each fault looks like, and therefore labelled failure data is required. Fault classification occurs in the supervised learning setting. With supervised learning, the goal is to learn the conditional distribution $p(h|\boldsymbol{x})$,

i.e. given the measured input data $\boldsymbol{x}$, determine the probability that the output h is a certain class. The input data sample $\boldsymbol{x}$ is then assigned to the most probable class, as depicted in Fig. 1(a). To learn $p(h|\boldsymbol{x})$, one needs access to labelled input data, where for each sample in $\boldsymbol{x}$, the desired output h is available. An example of supervised learning used for fault classification is the work of Janssens et al. (2016), where a Convolutional Neural Network (CNN) was trained to classify bearing faults, taking the Discrete Fourier Transform (DFT) of the vibration signals as input.



(a) Classification

(b) Detection and tracking

Figure 1: Automatic fault classification, detection and tracking using learning-based methods

To simplify the learning task, $p(h|\boldsymbol{x}^*)$ can also be learned, since it is easier to learn the relationship between the low-dimensional features $\boldsymbol{x}^*$ and h than it is to learn the relationship between the high-dimensional input data $\boldsymbol{x}$ and h. This is particularly useful when limited labelled data is available. Supervised and unsupervised learning methods can be combined (referred to as semi-supervised methods), by automatically extracting a lower-dimensional representation $\boldsymbol{z}$ in an unsupervised manner to use as $\boldsymbol{x}^*$, and then feeding $\boldsymbol{z}$ into a supervised classification model to learn $p(h|\boldsymbol{z})$. An example of semi-supervised learning in the condition monitoring literature includes the work of Jia et al. (2016), where a deep neural network was trained to classify gearbox faults. The network was pre-trained layer-wise using Auto-encoders (AEs) in an unsupervised setting, and thereafter fine-tuned on the available labelled fault data in a supervised setting. Zhang et al. (2021) showed how semi-supervised methods can be used to robustly train classifiers when only a small part of the dataset is labelled, or when the dataset includes many mislabelled samples. A Variational Auto-encoder (VAE) model was trained to reconstruct the raw data, with the latent representations $\boldsymbol{z}$ of the raw data used as inputs to the classifier.

Learning-based classification techniques generally require large labelled failure datasets to successfully train the models. Recent condition monitoring research has focused on developing transfer learning techniques to address the problem when labelled failure datasets are not available for a machine (Guo et al. 2019, Wen et al. 2019). These methods focus on using labelled failure datasets from other machines where it is easy to obtain labelled failure data (referred to as the source domain) to train classification models, and then transfer the knowledge gained by the models to a target domain (the machine with no historical failure database). When the knowledge is successfully transferred, the models can classify faults in machines where it is difficult to obtain labelled failure data.

In Fig. 1(a), we can see that simply attaching a class label of 'Healthy', 'Failure Mode 1' or 'Failure Mode 2' to the data does not truly capture the condition of the asset, as it does not take the fault severity, depicted by the colour bar, into consideration. During normal degradation processes, the

asset slowly transitions from one state to the other, and there is no clear boundary that indicates that the condition changed from one state to the other. Most learning-based diagnostics implementations in literature follow a classification approach, using large historical datasets, where the samples are labelled as healthy or as a certain damage mode. Classifiers are then trained on these target labels in a supervised manner, to create decision boundaries between these classes. The severities of the faults in the training data are not considered. The data that is recorded during the transition phase (the early stages of damage) can either be classified as healthy or unhealthy, based on the severity of the faults that were present in the training data. This approach can give poor predictions if limited labelled failure data is available, as the models can easily overfit on the fault data, giving poor predictions when the failure mode is at a slightly different severity as was seen in the training data (Schmidt & Heyns 2019). Classification is important for industrial applications to inform the maintenance personnel what part should be repaired or replaced, but to truly get value from a CBM program, they also need to be able to detect a fault as early as possible, and then accurately track how the fault develops.

1.2.3 Fault detection and tracking

The tasks of fault detection and fault tracking are usually addressed simultaneously by monitoring a health indicator, as depicted in Fig. 1(b). The health indicator values at the start of the asset's life cycle are used as reference points, since it can be assumed that the asset would normally then be in a healthy condition. Using these reference health indicator values, a discrepancy threshold can be set. The health indicator is then continuously monitored, and when the threshold is exceeded, it is an indication that the condition of the asset has changed, so a fault is detected. The health indicator can then be further monitored to track the rate at which the fault develops. Accurate prognostic predictions can be made from historical data if the health indicator rises steadily as the fault develops. Various metrics have been proposed to evaluate how suitable a health indicator is for prognostic predictions, such as monotonicity, the health indicator's robustness to noise, and the trendability over time (Lei et al. 2018). Features extracted directly from the signal, such as statistical features or the SES, are often used as health indicators (Bartelmus & Zimroz 2009, Smith & Randall 2015), but to know which features will be sensitive to changes in the health condition of the machine, one needs to know how the damage presents itself in the data. To remove this requirement, one can also use unsupervised methods to perform discrepancy analysis.

Discrepancy analysis is a novelty detection approach, that has been successfully applied to bearing and gear diagnostics (Schmidt et al. 2019*a*, Heyns et al. 2012). A discrepancy measure is assigned to each point in a signal, that quantifies how much the point deviates from a model of the signal under healthy conditions. The discrepancy scores can then be used as health indicators, since it is expected that the closer to failure the machine is, the more the condition monitoring signals will differ from the signals associated with healthy conditions. The discrepancy analysis framework is similar to residual signal analysis, where the goal is to remove the deterministic/first-order cyclostationary part of the signal (mostly influenced by healthy factors), so that the random/second-order cyclostationary part, where faults manifest, can be analysed (Randall & Antoni 2011). To perform discrepancy analysis, the healthy data distribution $p(\boldsymbol{x}_{healthy})$ must be modelled. Unsupervised learning can be used to learn the distribution of the healthy data $p(\boldsymbol{x}_{healthy})$ (Heyns et al. 2012, Booyse et al. 2020, Balshaw et al. 2022). To simplify the learning task, $p(\boldsymbol{x}^*_{healthy})$ can also be learned, where a fault is then detected when the extracted features start to differ from the distribution of the extracted features associated with the healthy data.

Another interesting approach to unsupervised fault detection is to continuously re-train a model to capture new data as it becomes available, and then monitor the model parameters for changes, which will indicate that the data distribution starts to change because of damage. Zimroz et al. (2014)

continuously re-trained a simple linear regression model to fit the relationship between the vibration signal features (RMS or peak-to-peak) and the operating conditions, monitoring the linear regression model's parameters to detect damage. Hou et al. (2022) proposed an optimized SES, where a hyperplane is fit to optimally separate two groups of normalized square envelope spectra, with one group obtained from healthy data, and the other from data with an unknown health condition. In this case, the model parameters are interpretable, as the optimal parameters for the model will capture the fault frequencies present in the samples from the unknown health condition. The group with the unknown health condition is continuously updated as new data becomes available, and the model is then re-trained. The model parameters are monitored to detect any fault frequencies in the new data.

In the prognostic setting, h will be the RUL of the asset, making it a supervised regression task, where $p(h|\boldsymbol{x})$ or $p(h|\boldsymbol{x}^*)$ must be learned. Li et al. (2018) used deep CNNs to predict the RUL of turbofan engine units. It is not possible to give a good prognostic prediction if only healthy data is available, as the expected rate of degradation once it reaches a certain degradation state will not be known. Successful fault tracking can however already add great value to a maintenance program, as the rate of degradation can be continuously monitored to see when a fault is stable, or when it develops quickly. Fault tracking can also easily be extended to prognostic predictions when historical failure data become available, by mapping the health indicators to the RUL values.

Fault classification tasks and fault detection and tracking tasks are often addressed separately. Schmidt & Heyns (2019) proposed using an open set recognition methodology, where these tasks are combined to provide a clear framework for how companies can address the problem of not having historical failure data for classification. Discrepancy analysis is used to detect when the newly observed data starts to differ from the healthy data in the feature space, and when it reaches a certain threshold, it is classified as a damage mode. As the fault develops, this new data is added to the set of known conditions, and the feature space for that particular damage mode starts to become populated. This newly acquired failure data can then be used in the future to classify similar failure trajectories in the feature space. If a condition does not match any of the historical data, a novelty is detected, and a new failure class is populated. Gaussian Mixture Models (GMMs) are used to allow for probabilistic inference, as it is recognized that during the transition phase between healthy and damaged, many of the classes overlap.

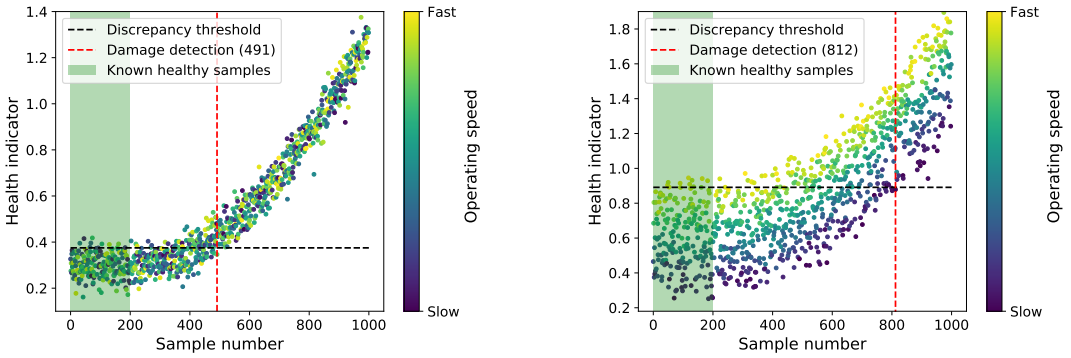
In this work, latent variable models are used to capture the healthy data distribution. Discrepancy signals are generated from these models, and are used to automatically detect and track faults. While no classification models are implemented in this work, it is also proposed that discrepancy signals can be integrated into classification tasks by making use of transfer learning methods, to automatically classify faults when historical failure data is not available for the monitored asset. The proposed fault detection and tracking methodology can also easily be extended to an open set recognition methodology, as proposed by Schmidt & Heyns (2019).

1.3 Condition monitoring under time-varying operating conditions

Condition monitoring under time-varying operating conditions is much more challenging than under stationary operating conditions. For rotating machinery, the frequencies of the vibrations caused by faults on shafts, gears and bearings will vary in proportion to the operating speed of the machine. Fluctuations in the operating speed will cause smearing in the frequency spectrum of the vibration signal, making it difficult to identify fault frequencies (Lin & Zhao 2015). In many cases, the clearest indication of a fault developing and becoming more critical is an increase in the vibration amplitude. For this reason, many approaches monitor simple statistical features of the vibration signals, with RMS values being the most widely used health indicator in prognostic tasks (Lei et al. 2018). Time-varying operating conditions also cause amplitude modulation in vibration signals, making it difficult

to distinguish whether changes in the vibration amplitude are caused by the varying operating conditions, or by changes in the health state of the machine (Schmidt & Heyns 2020). The deterministic and random components of the signal are amplitude modulated, making features such as the RMS correlated with the operating speed. This modulation also makes the data heteroscedastic, with the variance of the noise being larger when the machine is operated at higher speeds.

Fig. 2 shows how the correlation between the monitored features and operating speed leads to poor fault detection and tracking results. When the features are correlated with the operating speeds, the damage is detected (specified in this case as the sample from where no false negatives are classified further onward) much later than when the features are not correlated with the operating speeds. The fault can also be tracked with much more certainty when the features are not correlated with the operating speeds.



(a) Uncorrelated

(b) Correlated

Figure 2: Fault detection and tracking with features (a) uncorrelated to the operating speed vs. (b) features correlated to the operating speed .

The condition monitoring task, under varying speed conditions, can be significantly simplified by incorporating the shaft rotational speed and phase information into the analysis. Order tracking (OT) can be implemented to remove the spectral smearing caused by varying speed conditions. With OT, the signal is sampled at a rate proportional to the shaft rotational speed. Instead of sampling at constant time intervals (constant number of samples per second), sampling is done at constant shaft angular displacement intervals (constant number of samples per shaft revolution). OT can either be performed using analogue hardware that samples at a rate proportional to the shaft speed, or the signal can be sampled at constant time intervals, and then be re-sampled afterwards based on shaft speed measurements. The latter case is known as Computed Order Tracking (COT) (Fyfe & Munck 1997). For COT, the shaft speed is recorded with a tacho signal. After order tracking, a Fourier Transform can be applied to get an order spectrum, where the vibration orders caused by a certain fault will line up in the spectrum, with minimal smearing, making it easier to identify faults. For cases in industry where the shaft speed is not recorded, tacholess order tracking methods have been proposed, where the shaft speed and phase are estimated directly from the vibration signal so that order tracking can still be applied (Schmidt et al. 2018*b*, Lu et al. 2019).

Another advantage of order tracking the data is that it allows one to perform synchronous averaging on the input data, or the discrepancy signals. The synchronous average is obtained by point-wise averaging over several signal segments with the same length. The signal segments are usually chosen to correspond with one shaft rotation period. By doing this, non-synchronous components, such as noise, get averaged out, and components synchronous to the shaft rotation remain. Shaft localised faults, such as gear faults, can be detected in this manner. Heyns et al. (2012) order tracked raw vi-

bration signals, and fit a GMM to the data. Discrepancy analysis was performed, and the discrepancy signals were then synchronously averaged to enhance the components caused by the gear faults.

The problem of amplitude modulation can be addressed from different angles. Some approaches try to learn the relationship between a diagnostic metric, such as the RMS, and the shaft speed, using a regression model (Bartelmus & Zimroz 2009, Zimroz et al. 2014, Schmidt et al. 2019*a*). For a certain shaft speed, the expected value for the diagnostic metric at a healthy condition can be calculated with the regression model, and then compared to the actual diagnostic metric, to detect changes in the condition of the machine. These methods effectively learn the conditional distribution $p(\boldsymbol{x}^*_{healthy}|\boldsymbol{c})$, where $\boldsymbol{c} \in \mathbb{R}^{D_{\boldsymbol{c}}}$ is the associated operating condition information. Other approaches remove the effect of amplitude modulation by applying normalisation techniques to the signals with varying operating conditions. Schmidt & Heyns (2020) proposed the Normalisation of the Amplitude Modulation caused by Varying Operating Conditions (NAMVOC) procedure, where it was shown that, if this technique is used as a pre-processing step, the RMS can still be used to track fault severity in prognostic tasks.

In this work, it is investigated to what extent the unsupervised learning task presented to latent variable models, can be simplified by incorporating operating condition information in the process, specifically the shaft speed and phase information. This can be done during pre-processing, by order tracking the data, or during modelling, by learning the conditional distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c})$. If the availability of the shaft speed and phase information significantly improves the models' performances, a case can be made to invest time into tacholess order tracking methods to extract the shaft speed and phase information, and then combine the information in a semi-supervised learning setting, rather than attempting an unsupervised approach.

1.4 Discrepancy analysis from raw signals

1.4.1 Generative factors of a condition monitoring signal

To perform discrepancy analysis, we first have to model the healthy data. Consider a vibration signal $\boldsymbol{x}_{healthy}$, recorded from a healthy machine. The signal is sampled at 25 kHz, for a time window of 1 second. This makes $\boldsymbol{x}_{healthy}$ a 25000-dimensional vector, meaning that $p(\boldsymbol{x}_{healthy})$ is a distribution in the high-dimensional $\mathbb{R}^{25000}$ space, which makes it infeasible to model this distribution. Let's consider generative factors that could have influenced the 25000 values captured:

- Operating speed: The frequencies present in the signal will mostly be harmonics of the shaft rotational frequency.
- Shaft rotation angle when the recording button was pressed: If the vibration signal was recorded slightly later, the shaft would have been at a different rotation angle, causing the vibration signal to differ by a phase shift from the one we are currently considering.
- Operating load: If the loads on the machine vary significantly, the loads can change the stiffness of some parts, changing the natural frequencies of the system and influencing the measured vibrations.
- Environmental conditions: The environmental conditions (e.g temperature, pressure) can cause some parts to undergo thermal expansion, causing parts to press harder against each other, which influences the measured vibrations.
- Other significant factors that we are not aware of.

Along with the generative factors, there is also measurement noise present in the 25000 values captured. The operating and environmental conditions mentioned above are denoted by $\boldsymbol{c}$. Note that the physical properties of the machine are not included in this list of $\boldsymbol{c}$, as these properties are assumed to either be constant or to be complex functions of the variables included in $\boldsymbol{c}$ (e.g. the stiffness of a material might change with a change in temperature). The operating and environmental conditions $\boldsymbol{c}$ can be divided into those that we have measured, $\boldsymbol{c}_{known}$, and those that we have not measured or are not aware of, $\boldsymbol{c}_{unknown}$.

Considering this short list of generative factors, it is clear that the 25000 values measured are only influenced by a handful of underlying generative factors. It should therefore be possible to model the vibration signal $\boldsymbol{x}_{healthy}$ with a model taking fewer than 25000 inputs. These generative factors apply constraints to the observed 25000 variables, constraining them to a small area in the $\mathbb{R}^{25000}$ space. This assumption, that there exists a set of latent variables $\boldsymbol{z}$ in a lower dimensional subspace, that captures all the details of the observed variables $\boldsymbol{x}$, is known as the manifold hypothesis (Fefferman et al. 2016).

A manifold is a smooth n-dimensional topological space, that locally resembles Euclidean space near each point on the manifold. The manifold exists in some higher dimensional space. Lower-dimensional manifolds can be found in a variety of high-dimensional data, such as images and speech data. These manifolds are formed by the constraints arising from physical laws. Lin et al. (2016) explain this concept by using the example of classifying images containing cats or dogs, explaining how all natural images fall on a manifold that covers only a small region in the high-dimensional input space of an image, which allows models with parameters far less than the dimensionality of the input space, to successfully classify the images.

Returning to the vibration signal, if it is possible to accurately model vibration signals recorded from the machine under all possible values for $\boldsymbol{c}$ with a vector-valued function $\boldsymbol{x}_{healthy} = \boldsymbol{f}(\boldsymbol{c})$, it would mean that the vibration signals all fall on a $D_{\boldsymbol{c}}$-dimensional manifold in the $\mathbb{R}^{25000}$ input space. Vibration signals from similar conditions $\boldsymbol{c}$ (e.g. similar shaft speeds) will fall close to one another on this $D_{\boldsymbol{c}}$-dimensional manifold. When the vibration signals start to capture the effect of damage in the machine, the data will fall off this $D_{\boldsymbol{c}}$-dimensional manifold, and the function $\boldsymbol{x}_{healthy} = \boldsymbol{f}(\boldsymbol{c})$ will not be able to accurately generate the data.

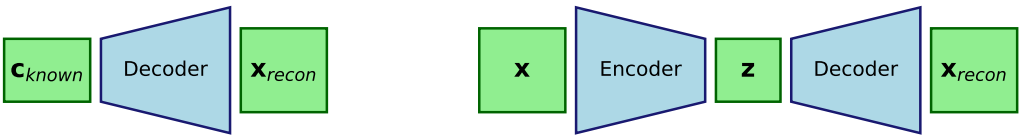
1.4.2 Modelling the healthy data using observed variables vs. using latent variables

When performing discrepancy analysis, one asks how likely it is that the measured $\boldsymbol{x}$ is from a healthy machine operated at the current set of operating conditions $\boldsymbol{c}$, so we are not interested in modelling a deterministic function $\boldsymbol{x}_{healthy} = \boldsymbol{f}(\boldsymbol{c}_{known}, \boldsymbol{c}_{unknown})$, but rather in capturing the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known}, \boldsymbol{c}_{unknown})$. By conditioning on $\boldsymbol{c}_{known}$ and $\boldsymbol{c}_{unknown}$, one points to a specific location on the manifold where one expects to find healthy data for that specific set of operating and environmental conditions.

If all of the generative factors included in $\boldsymbol{c}_{unknown}$ play only a minor role in what $\boldsymbol{x}_{healthy}$ looks like, the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known}, \boldsymbol{c}_{unknown})$ will resemble the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known})$. It will be possible to model the healthy data distribution in a completely supervised manner, by simply mapping each set $\boldsymbol{c}_{known}$ to the corresponding $\boldsymbol{x}_{healthy}$, to learn the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known})$. Fig. 3(a) shows a schematic representation of this supervised model, taking $\boldsymbol{c}_{known}$ as input, and then learning to reconstruct $\boldsymbol{x}_{healthy}$ as accurately as possible by decoding $\boldsymbol{c}_{known}$. This model architecture is referred to in this work as a C-decoder.

If some of the generative factors included in $\boldsymbol{c}_{unknown}$ play a major role in determining what $\boldsymbol{x}_{healthy}$

looks like, one will expect that a supervised approach will not work well, since $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known})$ and $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known}, \boldsymbol{c}_{unknown})$ will be vastly different. Unsupervised latent variable models can then be used to model the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known}, \boldsymbol{c}_{unknown})$. All the latent variable models considered in this work have some form of encoder and decoder function, as shown in Fig. 3(b). The models are trained to project the high-dimensional input data $\boldsymbol{x}_{healthy}$ to a lower-dimensional latent representation $\boldsymbol{z}$ using the encoder function, and then reconstruct the input data as accurately as possible by only considering $\boldsymbol{z}$, using the decoder function. This forces the model to learn how to capture only the most important information present in $\boldsymbol{x}_{healthy}$ in the latent space representation $\boldsymbol{z}$. The latent representation will be an abstract representation of all the significant generative factors of $\boldsymbol{x}_{healthy}$, so $\boldsymbol{z}$ will be a representation of $\boldsymbol{c}_{known}$ and $\boldsymbol{c}_{unknown}$. Latent variable models allow one to model the healthy data distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known}, \boldsymbol{c}_{unknown})$, without knowing $\boldsymbol{c}_{unknown}$, by approximating it with the density $p(\boldsymbol{x}_{healthy}|\boldsymbol{z})$. This is the main reason why we use latent variable models.



(a) Supervised C-decoder model

(b) Unsupervised latent variable model

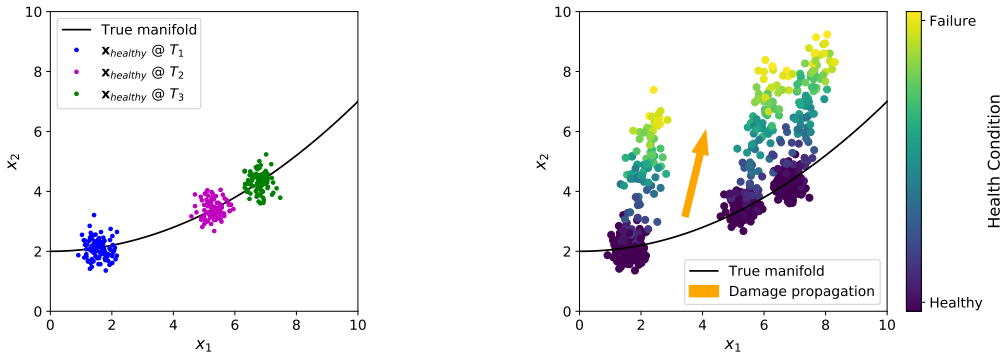
Figure 3: Schematic representation of the supervised decoder model and the unsupervised latent variable model.

To illustrate this concept, let's consider the following toy problem: we are interested in monitoring the condition of a machine operated at constant operating conditions (constant shaft speed and loads), but under varying environmental temperatures, denoted by T. The sensors used during the task are an accelerometer that measures the vibrations, and a shaft encoder that records the shaft speed. The temperatures play a large role in what each vibration sample $\boldsymbol{x}$ looks like, but are not recorded. To simplify the discussion, we also assume that the temperature is the only significant generative factor that causes variance in $\boldsymbol{x}_{healthy}$, since the other notable generative factors such as operating speed are kept constant. So $\boldsymbol{c}_{known}$ only includes the constant shaft speed, which is irrelevant to modelling $\boldsymbol{x}_{healthy}$, since it is kept constant and therefore does not cause any variance in the vibration samples. $\boldsymbol{c}_{unknown}$ consists of the environmental temperature T.

Vibration samples from a healthy machine $\boldsymbol{x}_{healthy}$ are recorded on two different days, where the environmental temperature differs substantially between the two days. The temperatures are denoted as T_1 and T_2. To simplify visualization, $\boldsymbol{x}$ is represented as a 2-dimensional vector, but in reality, it will be a high-dimensional vector that is difficult to model. Fig. 4 shows the true manifold on which all healthy data samples lie for different environmental temperature values. Fig. 4(a) shows the healthy samples $\boldsymbol{x}_{healthy}$ recorded at T_1 and T_2 that represent the training data. Healthy samples $\boldsymbol{x}_{healthy}$ recorded at T_3, are also included in Fig. 4(a), which will be part of the discussion later on, but for now, it is important to understand that only the healthy samples at T_1 and T_2 are available to train the models with, and we do not have temperature measurements available. Fig. 4(b) shows how damage presents itself in the vibration data, moving further from the true manifold as the damage level increases.

We now consider that a new vibration signal is measured at T_1, and we want to determine the likelihood of the sample being from the healthy data recorded at T_1 to determine the current health state of the machine. The machine is in a healthy state, but we do not know this. The lower the likelihood, the more likely it is that the machine is damaged. To do this, we have to model the distribution

$p(\boldsymbol{x}_{healthy}|T_1, \boldsymbol{c}_{known})$. Note that because $\boldsymbol{c}_{known}$ plays no significant role on $\boldsymbol{x}_{healthy}$, it can also be approximated as $p(\boldsymbol{x}_{healthy}|T_1)$. Fig. 5 shows the distribution $p(\boldsymbol{x}_{healthy}|T_1)$. It can be seen that the correct likelihood for the new sample would be 1.77.



(a) Healthy data positioned on the manifold

(b) Unhealthy data positioned off the manifold

Figure 4: Healthy data manifold, along with (a) healthy samples at T_1, T_2 and T_3, and (b) unhealthy samples at T_1, T_2 and T_3 moving off the manifold as the damage level increases.

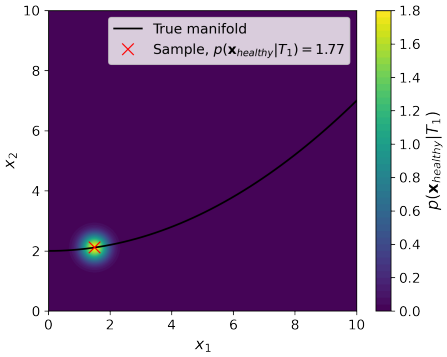


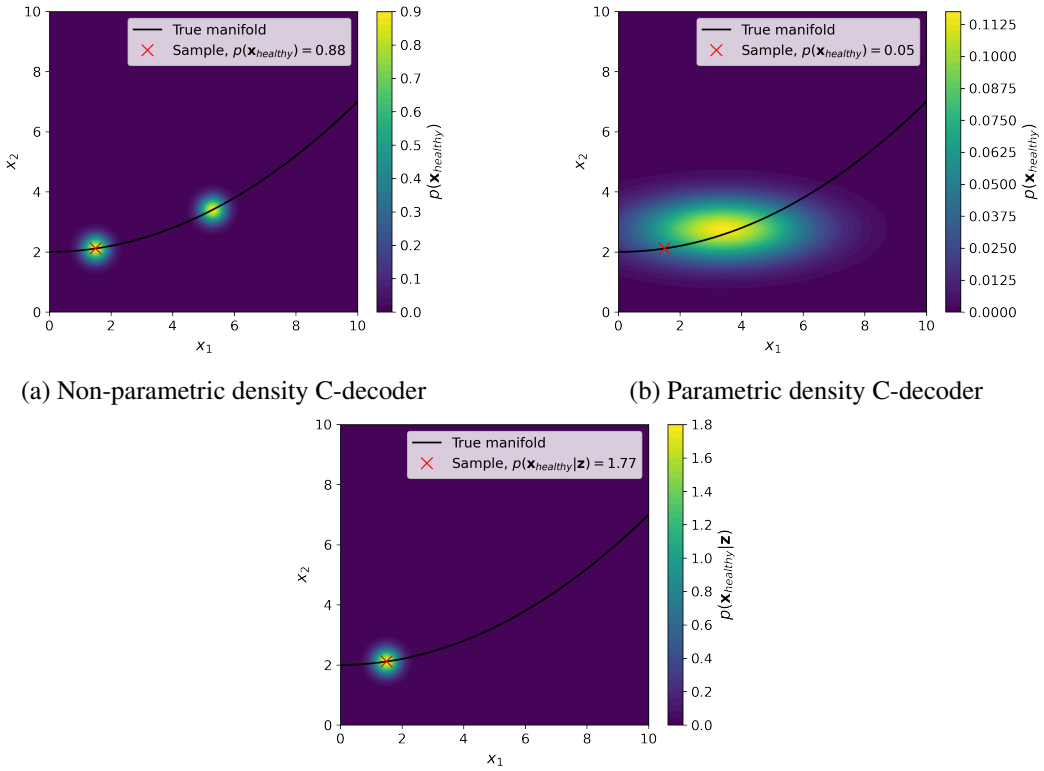
Figure 5: $p(\boldsymbol{x}_{healthy}|T_1)$, along with the new measured sample.

We are interested in capturing the distribution $p(\boldsymbol{x}_{healthy}|T, \boldsymbol{c}_{known})$, but since T was not recorded, we will have to model the distribution without knowing T. We have four options to approximate $p(\boldsymbol{x}_{healthy}|T, \boldsymbol{c}_{known})$ with:

1. Using a supervised C-decoder model, we learn the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known})$. We use a non-parametric model, such as a kernel density estimator to fit the distribution.
2. Using a supervised C-decoder model, we learn the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known})$. We use a parametric model, assuming that the distribution has some parametric form e.g. Gaussian.
3. Using a latent variable model, we can learn $p(\boldsymbol{x}_{healthy}|\boldsymbol{z})$ in an unsupervised setting.
4. Using a latent variable model, we can learn $p(\boldsymbol{x}_{healthy}|\boldsymbol{z}, \boldsymbol{c}_{known})$ in a semi-supervised setting. This can potentially simplify the learning process when the latent representations $\boldsymbol{z}$ are mostly abstract representations of generative factors included in $\boldsymbol{c}_{known}$. Since there is no useful information recorded in $\boldsymbol{c}_{known}$ in this scenario, this approach is equivalent to Option 3, and is therefore omitted from the rest of this discussion.

Fig. 6 shows the approximations for $p(\boldsymbol{x}_{healthy}|T_1, \boldsymbol{c}_{known})$ learned by the three options. Option 1 returns the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known})$ (or $p(\boldsymbol{x}_{healthy})$), recovering the two modes at T_1 and

T_2. The predicted likelihood for the new sample is 0.88 (compared to the correct 1.77). Option 2 performs even worse, by capturing the mean and variance of the training data samples from both T_1 and T_2, since the distribution is assumed to be Gaussian. The predicted likelihood for the new sample is 0.05 (compared to the correct 1.77). Not knowing T, when it plays a large role in what $\boldsymbol{x}_{healthy}$ looks like, makes it impossible to model $p(\boldsymbol{x}_{healthy}|T, \boldsymbol{c}_{known})$ in a supervised manner. This is where unsupervised latent variable models become useful. Without knowing T, latent variable models can be used to project $\boldsymbol{x}_{healthy}$ to a lower dimensional representation $\boldsymbol{z}$, that encodes all the generative factors that play a large role in what $\boldsymbol{x}_{healthy}$ looks like (i.e. T), and then learn to reconstruct $\boldsymbol{x}_{healthy}$ from $\boldsymbol{z}$. The encoder models the density $p(\boldsymbol{z}|\boldsymbol{x}_{healthy})$, and the decoder the density $p(\boldsymbol{x}_{healthy}|\boldsymbol{z})$. When new data is now recorded at T_1, $\boldsymbol{x}$ will be projected to a latent representation $\boldsymbol{z}$ corresponding to T_1, resulting in the correct distribution being learned, shown in Fig. 6(c), giving the new sample the correct likelihood of 1.77.



(a) Non-parametric density C-decoder

(b) Parametric density C-decoder

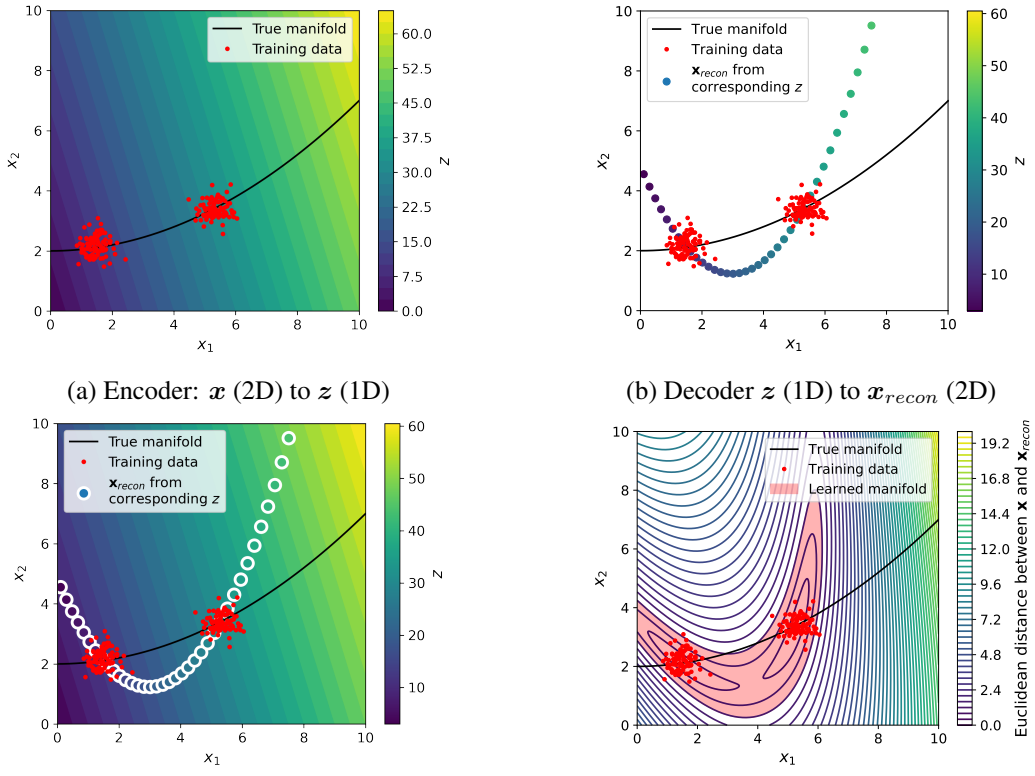
(c) Latent variable model

Figure 6: Approximated densities for $p(\boldsymbol{x}|T_1, \boldsymbol{c}_{known})$, using (a) a non-parametric density supervised C-decoder model, (b) a parametric density supervised C-decoder model, and (c) an unsupervised latent variable model.

If the temperatures were recorded in this scenario, the supervised C-decoder models would also have been able to accurately capture the distribution $p(\boldsymbol{x}_{healthy}|T_1, \boldsymbol{c}_{known})$, making latent variable models redundant for this application. This work explores this concept in detail, to see to what extent supervised models can be used to map available operating condition information $\boldsymbol{c}_{known}$ to the healthy data $\boldsymbol{x}_{healthy}$ to perform discrepancy analysis. It is also done to identify when it becomes beneficial to rather use latent variable models to model the healthy data distribution, and to what extent $\boldsymbol{c}_{known}$ can be incorporated into latent variable models in a semi-supervised setting, to simplify the learning process. We specifically investigate the case where $\boldsymbol{c}_{known}$ includes the shaft speed and phase information, as it is expected that these generative factors are responsible for most of the variance in the healthy data.

1.4.3 Discrepancies in the reconstruction space vs. in the latent space

Latent variable models with encoder-decoder structures have been applied to a wide range of anomaly detection applications, operating on the idea that the models will reconstruct out-of-distribution (OOD) data poorly. OOD samples in this context refer to samples that are not from the same distribution as the training data samples $\boldsymbol{x}_{healthy}$. The encoder and decoder are trained together with the goal of reconstructing $\boldsymbol{x}_{healthy}$ as accurately as possible. By setting the latent space dimensionality $D_{\boldsymbol{z}}$ lower than the input space dimensionality $D_{\boldsymbol{x}}$, the latent variable model is forced to learn a lower-dimensional manifold in the high-dimensional input space, where only samples that fall on the learned manifold are reconstructed accurately. This concept is illustrated in Fig. 7, still using the example of the vibration signals recorded at T_1 and T_2. The latent space is 1-dimensional, so all possible input data in the 2-dimensional input space cannot be captured in the latent space. Fig. 7(a) shows how the encoder maps the 2-dimensional input space to a 1-dimensional latent space. Fig. 7(b) shows how the decoder then maps the 1-dimensional latent representations back to a 2-dimensional space, which is referred to in this work as the reconstruction space. Fig. 7(c) visualizes the encoder and decoder functions combined. Input samples that fall close to points where the colour of the encoded space matches the colour of the decoded space, will be reconstructed well, while input samples that are positioned further from these points will be reconstructed poorly. This is illustrated in Fig. 7(d), where a contour map of the reconstruction error (the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_{recon}$) are shown. Note how the areas closest to the training data have the lowest reconstruction error. The areas with reconstruction errors smaller than 1 are highlighted in red, to show the learned manifold. Input data falling on or close to this manifold will be reconstructed accurately, and less accurate the further away it is from the manifold.



(a) Encoder: $\boldsymbol{x}$ (2D) to $\boldsymbol{z}$ (1D)

(b) Decoder $\boldsymbol{z}$ (1D) to $\boldsymbol{x}_{recon}$ (2D)

(c) Encoder-Decoder combined

(d) Reconstruction error map

Figure 7: Visualization of how the encoder and decoder work together to learn a manifold, where all values of $\boldsymbol{x}$ on or close to the manifold will have a small reconstruction error. Reconstruction errors (Euclidean distance) smaller than 1 are highlighted in red to visualize the learned manifold.

Note that the learned manifold does not necessarily cover the whole true manifold. Only the regions where the training data samples are from, are guaranteed to be covered by the learned manifold. Also, note that the learned manifold covers some regions that are not close to the true manifold. The encoder and decoder have no way to know that healthy samples recorded at T_3 for instance should also be reconstructed accurately, since healthy samples recorded at T_3 were not included in the training data. If samples were now recorded at a different temperature T_3, the signals will be OOD samples. The encoder will map the data to an unknown location in the latent space, most likely off from the learned manifold, and the decoder will then reconstruct the healthy samples at T_3 poorly. The same idea holds for samples recorded from a damaged machine, as these signals will also be OOD samples.

The reconstruction space is used to calculate anomaly scores using measures such as the likelihood and the Mahalanobis distance. Recent literature has however shown that these models can in some cases reconstruct certain OOD samples very accurately, making measures such as the likelihood and the Mahalanobis distance in the reconstruction space unreliable for anomaly detection (Nalisnick et al. 2018, Xiao et al. 2020).

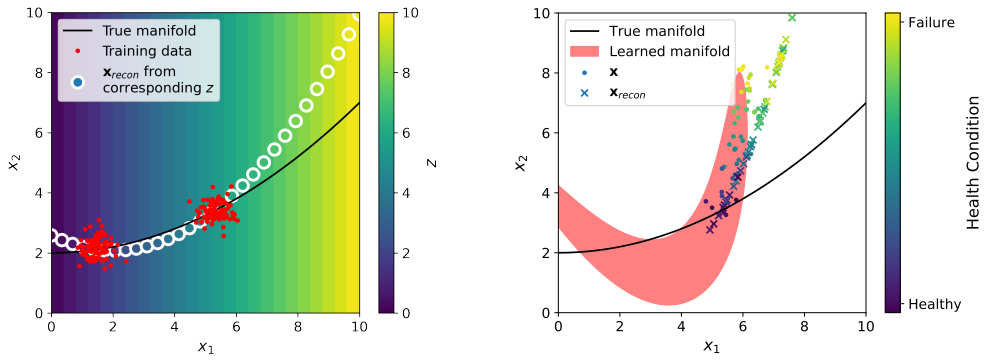
There are two concerns when monitoring the reconstructions of latent variable models for anomaly detection:

1. The model may overfit by learning an identity function transformation for some of the data points in the input samples. The model will then reconstruct these points accurately whether the samples are OOD or not (Steck 2020, Denouden et al. 2018). The information bottleneck provided by the encoder-decoder structure is thus not narrow enough to prevent the model from just memorizing parts of the input data. This concept is visualized in Fig. 8(a), where it is shown what the encoded and decoded space would look like if the model learns an identity function for x_1. Note that in Fig. 8(a), the model does not learn to store any information in x_2 in the latent space, with x_2 not influencing the encoded z value. The encoding-decoding transformation however acts as an identity function transformation for x_1, with the transformation from x_1 to z being completely invertible. If damage presents itself as a sudden spike in x_1, these spikes will be reconstructed perfectly, and the damage associated with the spike in x_1 won't be detected. It becomes more likely for this to happen when $D_{\boldsymbol{z}}$ is chosen to be close to $D_{\boldsymbol{x}}$. When $D_{\boldsymbol{z}}$ is equal to $D_{\boldsymbol{x}}$, the model can potentially learn to memorize all the input data values in the latent space and reconstruct any input perfectly.

2. The model may learn a latent manifold, where the training data falls on only a small part of this learned manifold. Any OOD sample that falls close to the regions on the learned manifold far from the training data, will also be reconstructed accurately (Denouden et al. 2018). This concept is illustrated in Fig. 8(b), where the reconstructions $\boldsymbol{x}_{recon}$ are compared with $\boldsymbol{x}$ recorded from a damaged machine at T_2, as was originally depicted in Fig. 4(b). Even though the damaged samples at T_2 are OOD samples, they also fall close to the learned manifold and are therefore reconstructed fairly accurately, making it difficult to detect the damage in the reconstruction space.

In these cases, the OOD samples cannot be reliably detected in the reconstruction space. The latent space representations of OOD samples at a certain temperature T_i will however differ from the latent space representations of the healthy training data at T_i, making it possible to detect the OOD samples in the latent space. Denouden et al. (2018) fitted a Gaussian distribution to the latent representations of the training data, and used the Mahalanobis distance between this distribution and the latent representation of new samples as an anomaly score. This will however only work if the training data latent representations are grouped so that a Gaussian distribution will fit the data well. This will not work when the training data samples are from multiple classes/operating conditions, as the latent

space representations for each class will be grouped together, but the latent space representations for different classes will not necessarily fall close together. Lee et al. (2018) detected OOD samples in a deep classification model, by using the Mahalanobis distance on latent representations. Gaussian distributions that were conditioned on the class labels, were fit to low- and high-level features (latent representations at different layers in the classification network). Because the input data samples were from different classes, the latent representations between classes would differ significantly, making it necessary to fit distributions conditioned on the class labels to the latent representations.



(a) Identity function learned for x_1. Any value of x_1 is reconstructed accurately.

(b) OOD samples ($\boldsymbol{x}_{unhealthy}$ @ T_2) falling on learned manifold are reconstructed accurately.

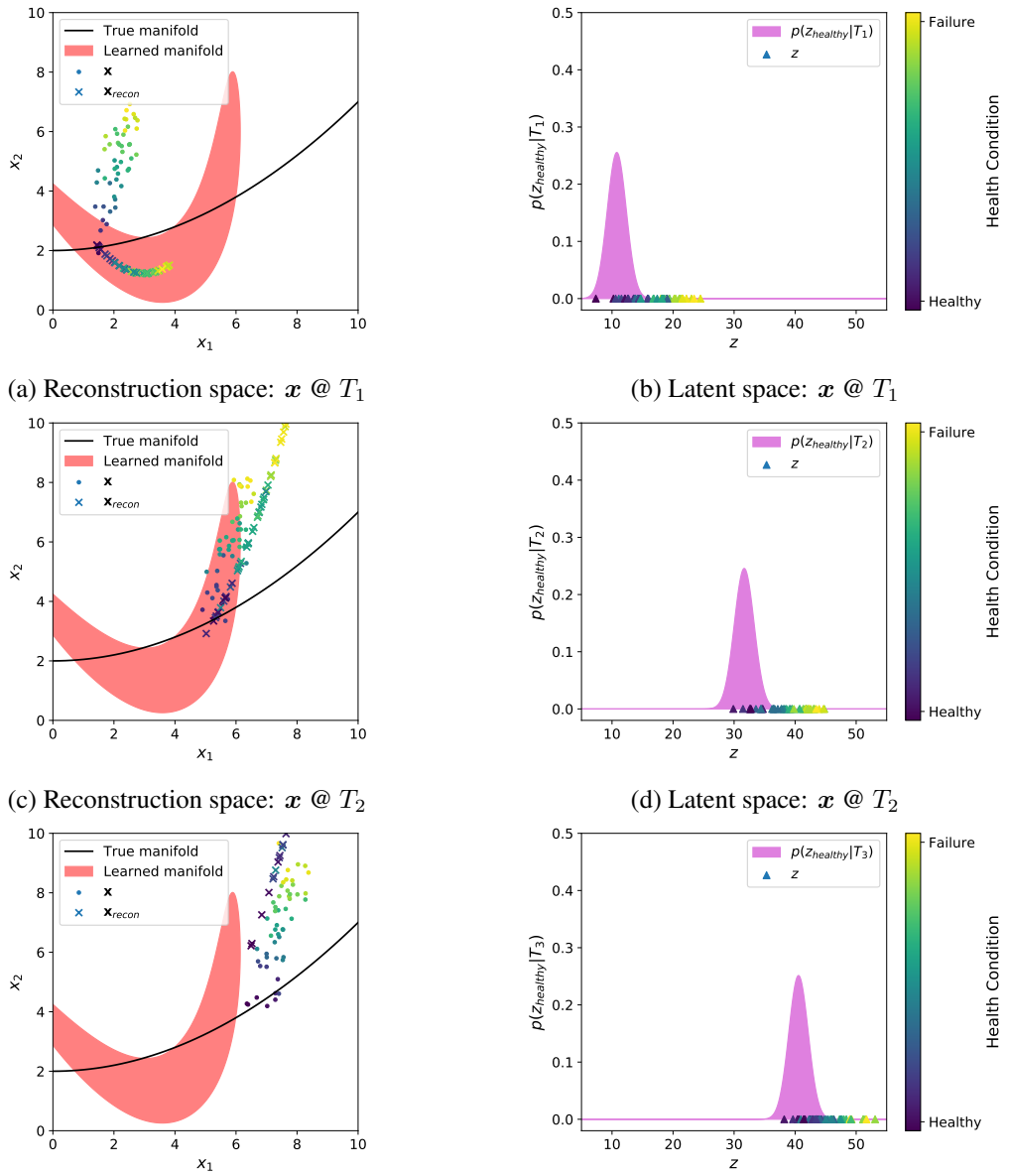
Figure 8: Concerns when monitoring the reconstruction space of LVMs for anomaly detection.

For the application of condition monitoring, it is thus crucial to not only monitor the reconstruction space to detect damage, but also the latent space. Balshaw et al. (2022) split the vibration signals into shorter segments by shifting a window one point at a time to create the input data matrix. By presenting the segments in this order to the models, it can be tracked how the latent representations change over time. Metrics were proposed that track the movement in the latent space over time between these consecutive samples with the idea that damaged data will cause unexpected jumps in the latent space that can then be detected.

In this work, the latent space is monitored using an approach similar to that proposed by Lee et al. (2018). When operating condition information is available, it can be used similarly as the class labels were used by Lee et al. (2018). The latent representations of the healthy training data are fitted with distributions conditioned on the available operating condition information. For the example discussed in this section, this will mean that the distribution $p(z_{healthy}|T)$ is learned, if the temperature measurements were available. If the temperature measurements were available, one would be able to learn what the expected latent representation z for a healthy sample would be at a given environmental temperature T_i. The Mahalanobis distance can then be used as a discrepancy metric for new samples. It is however not possible to monitor the latent space for discrepancies when the generative factor T that is responsible for the variance in the latent representations is not known.

Fig. 9 shows the reconstructions and latent space representations for the damaged samples at T_1, T_2 and T_3 depicted in Fig. 4(b). For the samples at T_1, the damage is easily detected in the reconstruction space, with the damaged samples falling off the learned manifold, causing poor reconstructions (Fig. 9(a)). The damaged samples at T_2 still fall close or on the learned manifold, resulting in the damaged samples being reconstructed fairly well (Fig. 9(c)). The reconstruction space is thus not ideal to detect damage at T_2. Fig. 9(d) shows how $p(z_{healthy}|T_2)$ could have been learned if measurements for T were available. The latent representations move further from the $p(z_{healthy}|T_2)$ probability mass as the damage increases, so if $p(z_{healthy}|T_2)$ was learned, discrepancy analysis could have been

performed on the latent space representations, and damage would have been detected. Fig. 9(e) shows that because samples at T_3 were not included in the training set, the model cannot be used to reliably detect damage at T_3. In this case, the model reconstructs both the healthy and damaged samples at T_3 poorly. The density $p(z_{healthy}|T_3)$ depicted in Fig. 9(f) can also not be learned accurately, even when measurements of T are available, since there were no samples at T_3 in the training set.



(a) Reconstruction space: $\boldsymbol{x}$ @ T_1

(b) Latent space: $\boldsymbol{x}$ @ T_1

(c) Reconstruction space: $\boldsymbol{x}$ @ T_2

(d) Latent space: $\boldsymbol{x}$ @ T_2

(e) Reconstruction space: $\boldsymbol{x}$ @ T_3

(f) Latent space: $\boldsymbol{x}$ @ T_3

Figure 9: $\boldsymbol{x}$ @ T_1, T_2 and T_3: Comparison of $\boldsymbol{x}$ vs. $\boldsymbol{x}_{recon}$ in the reconstruction space (left) and $\boldsymbol{z}$ vs. $p(\boldsymbol{z}_{healthy}|T_i)$ (right)

Note that the probability mass for $p(z_{healthy}|T_1)$, $p(z_{healthy}|T_2)$ and $p(z_{healthy}|T_3)$ are in three different regions in the latent space. At T_1, the expected z for healthy data is $z = 11$, at T_2 it is $z = 32$, and at T_3 it is $z = 41$. When the temperature values are not available, one will have no context of where the latent representation is expected to be at healthy conditions, making it difficult to monitor the latent space for OOD samples. Having operating condition information available enables one to easily monitor the latent space. In this dissertation, it is investigated how effectively the latent space

can be monitored using shaft speed and phase information. If the results are greatly improved by the availability of the shaft speed and phase information, then it should be considered to integrate tacholess order tracking methods with the proposed methodology when a tacho signal is not available.

It is also investigated how the latent space dimensionality affects where the damage is detected, as it is expected that a model with a larger latent space is more likely to learn an identity function between points in the inputs and outputs, which would make damage only detectable in the latent space.

1.5 Scope of Research

Two commonly used latent variable frameworks are implemented to perform discrepancy analysis on vibration signals: Principal Component Analysis (PCA), and Variational Auto-encoders (VAEs).

This work addresses the following three questions:

1. **When the operating conditions (shaft speed and phase information) are known or can be estimated, is it necessary to use latent variable models to model the healthy data distribution, or can the operating conditions simply be mapped to the healthy data in a supervised setting?**

   We investigate the case where the instantaneous shaft speed and phase information are available (obtained with a tachometer, or tacholess order tracking methods). Because we expect that the shaft speed and phase information are the generative factors behind most of the variance in the healthy data distribution, it is investigated to what extent the healthy data can be modelled in a supervised manner, mapping the shaft speed and phase information to the corresponding vibration signals using a C-decoder as explained in Section 1.4.2. In other words, we investigate how close the target distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known}, \boldsymbol{c}_{unknown})$ can be modelled by modelling $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known})$, where $\boldsymbol{c}_{known}$ is the shaft speed and phase information. This is done to get an idea of the complexity of a dataset, to see if there are other generative factors that are responsible for large amounts of variance in the healthy data distribution.

   If the C-decoder performs just as well as latent variable models on the discrepancy analysis tasks, then it would mean that the variance in the healthy data distribution can be sufficiently explained with only the shaft speed and phase information. This would make it redundant to use latent variable models to capture the healthy data distribution when the shaft speed and phase information are available, since the latent representations will only capture abstract representations of the shaft speed and phase information that we already know, while also being difficult to interpret.

   On the other hand, if the latent variable models perform significantly better than the C-decoder model, it will be a clear indication that the data samples are also influenced by other underlying generative factors that we are not aware of, or that are difficult to measure or extract. In this case, latent variable models will be very useful, as the models can learn to capture abstract representations of these unknown generative factors in the latent space, to successfully capture the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c}_{known}, \boldsymbol{c}_{unknown})$, by approximating it with $p(\boldsymbol{x}_{healthy}|\boldsymbol{z})$.

2. **To what extent can the availability of the instantaneous shaft speed and phase information improve the performance of latent variable models on fault detection and tracking tasks, when it is used in a semi-supervised setting?**

   In this work, the shaft speed and phase information are used to improve the performance of the latent variable models in the following ways:

- The data is order-tracked to simplify the learning process for latent variable models during pre-processing. Booyse et al. (2020) assumed that the vibration signals can be order-tracked before they are presented to the latent variable model. Balshaw et al. (2022) presented the raw vibration signals to the latent variable models without order tracking. To the author's knowledge, no one has systematically investigated whether the healthy data distributions are captured more accurately or not when order tracking is applied.
- The learning process is simplified during modelling by learning distributions conditioned on the latent variables and the known operating condition information. The instantaneous shaft speed and phase information are used during modelling in PCA to capture the heteroscedastic noise in the data, and for VAEs to get a more informed prior, by conditioning the prior on the available operating condition information.
- The operating condition information is used to monitor the latent space for discrepancies, as was explained in Section 1.4.3.

This dissertation systematically compares how the latent variable models perform in an unsupervised setting without order tracking, a semi-supervised setting where the shaft speed and phase information are used only for order tracking, a semi-supervised setting where the shaft speed and phase information are used only during modelling, and a semi-supervised setting where the shaft speed and phase information are used for order tracking and during modelling.

If the availability of the shaft speed and phase information significantly improves the models' performances, a case can be made to use tacholess order tracking methods to extract the shaft speed and phase information when it is not readily available, and then combine the information in a semi-supervised learning setting, rather than attempting an unsupervised approach.

For the cases where the unsupervised models perform much worse than the semi-supervised models, we try to address why this is the case, which leads to other minor contributions from this investigation. It is shown how linear and nonlinear latent variable models fit vibration data differently, and why the models fit order-tracked samples more accurately. It is also shown how the latent space dimensionality affects a latent variable model's performance on fault detection and tracking tasks, and that the latent space dimensionality plays a role in whether damage gets detected in the latent space or in the reconstruction space. The importance of using models that can capture the heteroscedastic noise associated with varying operating speeds is also highlighted.

3. **How do the discrepancy analysis methods perform compared to traditional envelope analysis methods?**

   The performances of the learning-based methods are compared with traditional signal processing (envelope analysis) approaches, to see if the learning-based methods can perform at the same level without any domain knowledge being built into them.

The potential benefits of combining discrepancy analysis frameworks with fault classification tasks using transfer learning techniques are also highlighted for future work. No transfer learning methods or fault classification models are implemented in this work.

The proposed methodology is applied to two datasets in this work. For the first dataset, a phenomenological model is used to create a gearbox vibration dataset under varying operating speeds with bearing inner and outer race faults. A phenomenological model is chosen, as this allows one to explicitly increase the damage level by known increments between samples, making it easier to evaluate a model's performance on fault detection and tracking tasks. The noise levels for this dataset are also chosen to be relatively low, so that the models' abilities to capture large noise distributions

are not the main focus in this investigation, but rather how well the models capture the generative factors such as the shaft speed and phase information in the latent space.

For the second dataset, the C-AIM experimental gearbox dataset is used (Schmidt et al. 2018*a*). The dataset consists of vibration data from a gearbox operated at varying operating conditions, with and without a gear-tooth fault. This dataset is chosen, as it is known to be a difficult dataset to analyse with latent variable models and traditional signal processing techniques. The vibration samples contain impulsive non-synchronous components at high frequencies in both the healthy and damaged data, which makes the healthy distribution difficult to learn. The difficulty of the learning task is adjusted by low-pass filtering the dataset, to get an indication of how much the impulsive components affect a model's performance.

1.6 Document Overview

Chapter 2 presents the proposed fault detection, classification and tracking methodology that is implemented in this study. **Chapter 3** introduces the latent variable models implemented in this work. Limitations in the PCA and VAE frameworks for unsupervised condition monitoring are discussed. It is shown how the available operating condition information is used in this work to overcome these limitations. In **Chapter 4**, a phenomenological model dataset is analysed using the proposed methodology. In **Chapter 5**, the experimental gearbox dataset is analysed using the proposed methodology. Conclusions and recommendations for future work are given in **Chapter 6**.

2 Proposed Methodology

The proposed fault detection, -tracking and -classification framework is discussed in this section. This framework is used for both the dataset analyses done in this work (documented in Chapters 4 and 5).

2.1 Overview

The proposed methodology can be divided into five steps, with Fig. 10 showing how the five steps flow from one to the other:

1. Pre-processing: The input data, consisting of the condition monitoring signals, and the operating condition information (if available), are processed before it is fed into a model.
2. Modelling: A model is trained to capture the distribution of the pre-processed training data $\boldsymbol{x}_{healthy}$. After training, the model can be used for the evaluation of unseen data.
3. Post-processing: Discrepancy signals are calculated and processed. These discrepancy signals capture how much the unseen input data differs from the distributions associated with the healthy samples. This includes reconstruction space discrepancy signals, as well as latent space discrepancy signals (when operating condition information is available to monitor the latent representations).
4. Fault detection and tracking: The processed discrepancy signals are analysed to detect a change from the healthy condition, and to track how the fault develops.
5. Fault classification: The processed discrepancy signals are analysed to determine the type of fault present.

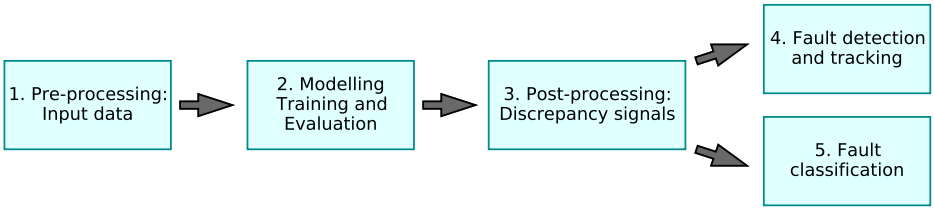


Figure 10: Schematic of the proposed methodology

Steps 1 to 4 can all be implemented without any operating condition information available (so in an unsupervised setting). While the proposed methodology allows one to detect and track faults without operating condition information available, this study aims to show how much the availability of the operating condition information can simplify the task, improving robustness and performance. Steps 1 to 4 will therefore also be implemented while making use of the available operating condition information (so in a semi-supervised setting). The proposed fault classification framework (step 5) requires that the shaft speed and phase information are known. This step is not implemented in this work (no classification models are trained), but it is shown how this discrepancy analysis framework can be useful to classification approaches when no labelled historical failure dataset on that particular machine is available.

A simple toy problem is used in this chapter to visualize the five steps of the proposed method. To make it simple, the operating conditions are kept constant. The measured signal consists of three components:

- A healthy component, represented by a sine wave with frequency f_{shaft}, so one shaft rotation creates one oscillation. f_{shaft} was chosen as 1.6 Hz.
- A noise component, represented by zero-mean Gaussian noise with a fixed variance.
- A damage component, represented by impulse responses for a viscously under-damped system, with a natural frequency of 51.2 Hz (32 times f_{shaft}). The impulses occur at a fixed frequency f_{damage}. Two damage modes are investigated:
  - Mode 1: f_{damage} equals f_{shaft}, so each shaft rotation leads to one impulse caused by the damage. The fault is thus correlated with the rotation angle of the shaft (as would be seen for a gear-tooth fault). The fault occurs at 270° from the reference shaft rotation position.
  - Mode 2: f_{damage} is not divisible by f_{shaft}, so the fault is uncorrelated with the rotation angle of the shaft (as would typically be seen in bearing faults). f_{damage} was chosen to be equal to 1.8 times f_{shaft}.

The signal components are shown in Fig. 11, for three shaft rotations. Each rotation is plotted in a different colour.

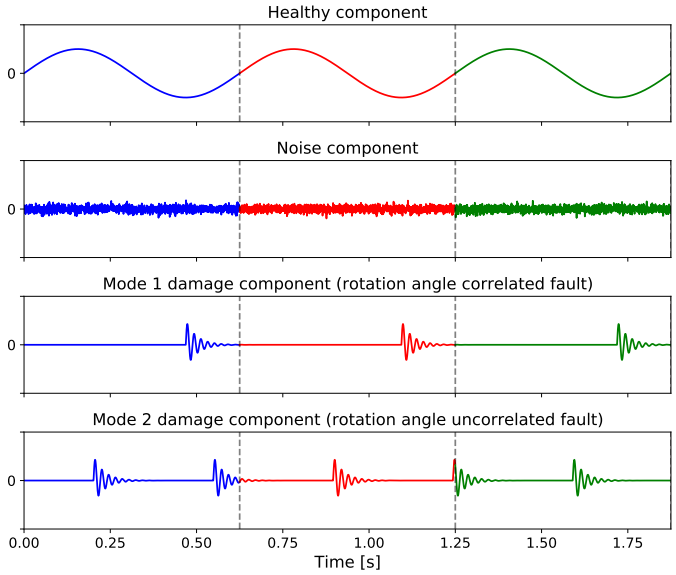


Figure 11: Toy problem: signal components

The measured signals are then generated by scaling the damage component with a factor α between 0 and 1, where 0 represents a healthy signal, and 1 represents the largest damage, and then adding the components together, as shown in Eq. (2-1):

$$\text{Measured signal} = \text{Healthy component} + \text{Noise component} + \alpha \times \text{Damage component} \tag{2-1}$$

Fig. 12 shows a healthy signal, a signal with a 'rotation angle uncorrelated'-fault ($f_{damage} = 1.8$ Hz, $\alpha = 1$) and a signal with a 'rotation angle correlated'-fault ($f_{damage} = 1$ Hz, $\alpha = 1$).

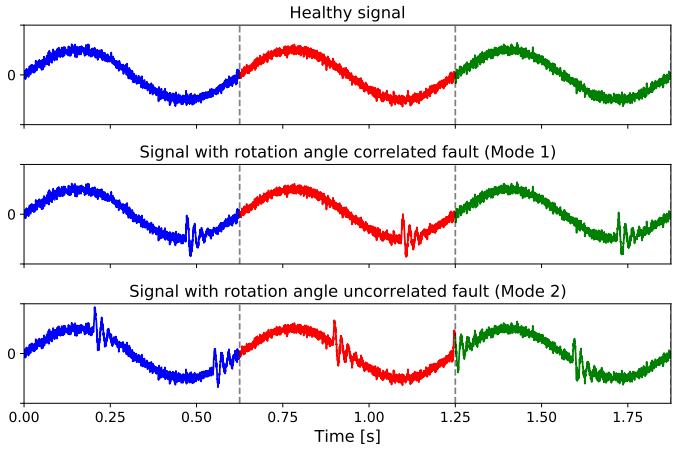


Figure 12: Toy problem: healthy and damaged signals

A dataset was generated consisting of healthy signals, and signals from both damage modes for different values of α. This dataset was then analysed using the proposed five steps.

The five steps are discussed in detail in the following subsections, except for the modelling step (step 2), which is only discussed briefly. The latent variable models implemented in this work are presented in Chapter 3. This allows the reader to get a full overview of the proposed methodology, before looking at the more technical formulations of the latent variable models. For each step, it is shown how it can be implemented in a completely unsupervised setting, as well as in a semi-supervised setting (except for the fault classification step, which requires operating condition information).

2.2 Pre-processing: input data

The input data fed into the model consists of the condition monitoring signals (vibration signals) and the operating condition information (if available). Data that are known to be from healthy machine conditions are used to train the model, and thereafter data from unknown machine conditions can be fed into the model for evaluation. Before the dataset is fed into the model, it is processed as follows:

- The healthy dataset is split into a training-, validation-, and test set, each having different roles during the training of the model.
- The condition monitoring signals are order tracked (if the shaft speed and phase information are available) and split into shorter windows with the same length (referred to as partitioning).
- All the data are normalized or centred, depending on the type of model used, using the mean and standard deviation of the training data.
- The training and validation sets are balanced based on operating conditions so that all operating conditions are equally represented in the training data. This is only done if the operating conditions are known.

2.2.1 Train-, validation- and test set split

The healthy dataset is split into a training-, validation-, and test set. The training set is used to train the model parameters, and also to set the normalizing parameters for the data. The validation set is used to prevent the model from overfitting on the training data. The test set is never used during training of the model. It is used to calculate the expected discrepancy signals of unseen healthy data, from which discrepancy thresholds can then be set. In this work, the Mahalanobis distance is used as

a discrepancy score. If discrepancy scores pass the threshold, the samples are classified to be from a damaged machine.

2.2.2 Partitioning: Section Only, Order Tracking, Order Tracking and Aligning

Condition monitoring signals such as vibration signals are sampled at very high frequencies. It is not feasible to use the whole signal as input to the models, as the inputs would have very large dimensionalities. For this reason, the common approach when using raw signals as inputs to a machine learning method is to use a direct partitioning scheme, where the input signal is split into several segments of the same length L_w. This is either done by randomly moving a window with length L_w over the signal or shifting the window a certain number of points at a time to obtain segments with a certain percentage of overlap (Baggerohr 2019, Booyse et al. 2020).

Balshaw (2020) proposed that one should partition and store the data in such a way that the information of time is preserved in between consecutive segments, by using a continuous time-analysis approach. It was shown that making this small change to the data preparation step, leads to many new possibilities, such as utilizing movements in the latent representations to detect damage. Balshaw et al. (2022) partitioned and stored each raw measurement $\boldsymbol{x}_i$ from $\boldsymbol{X}$ in a Hankel matrix in the following manner:

$$\boldsymbol{H}(\boldsymbol{x}_i) = \begin{bmatrix} \boldsymbol{x}_i(0) & \boldsymbol{x}_i(1) & \dots & \boldsymbol{x}_i(L_w) \\ \boldsymbol{x}_i(L_{sft}) & \boldsymbol{x}_i(L_{sft}+1) & \dots & \boldsymbol{x}_i(L_{sft}+L_w) \\ \boldsymbol{x}_i(2\times L_{sft}) & \boldsymbol{x}_i(2\times L_{sft}+1) & \dots & \boldsymbol{x}_i(2\times L_{sft}+L_w) \\ \vdots & \vdots & \dots & \vdots \\ \boldsymbol{x}_i(N_w\times L_{sft}) & \boldsymbol{x}_i(N_w\times L_{sft}+1) & \dots & \boldsymbol{x}_i(N_w\times L_{sft}+L_w) \end{bmatrix} \tag{2-2}$$

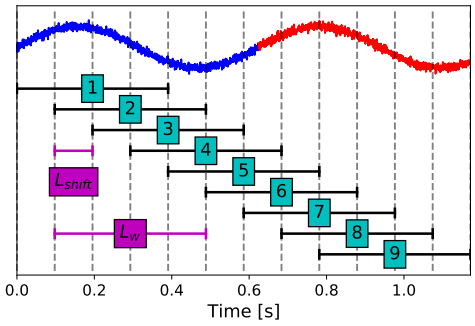
where L_w determines the window length, and L_{sft} is a shifting parameter that determines the number of sample points that overlap between windows. N_w is the number of L_{sft} jumps that fit into $(N_p - L_w)$, where N_p is the number of points in the measurement $\boldsymbol{x}_i$. Note that when L_{sft} is chosen to be equal to L_w, no overlap between consecutive windows occurs. Different lengths of L_w are investigated in this work, and L_{sft} is chosen based on the size of the dataset. If only a few samples are available for training, L_{sft} can be decreased to increase the size of the training dataset.

The Hankel matrices for all the measurements are then concatenated to form the input data matrix for the condition monitoring signals $\boldsymbol{X}_{data}$:

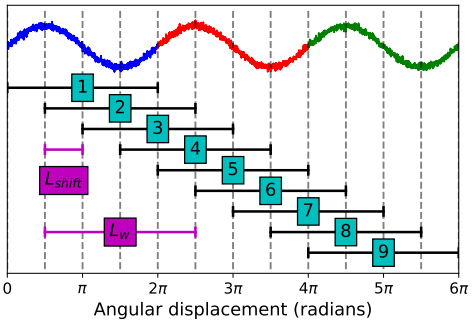
$$\boldsymbol{X}_{data} = \begin{bmatrix} \boldsymbol{H}(\boldsymbol{x}_1) \\ \boldsymbol{H}(\boldsymbol{x}_2) \\ \vdots \\ \boldsymbol{H}(\boldsymbol{x}_n) \end{bmatrix} \tag{2-3}$$

where $\boldsymbol{X}_{data} \in \mathbb{R}^{N_{tw}\times L_w}$, and N_{tw} refers to the total number of signal window segments included in $\boldsymbol{X}_{data}$. This approach of partitioning the raw signal is referred to in this work as Section Only (SO) pre-processing. Two other pre-processing approaches are proposed, that can be implemented when order tracking can be implemented (so when shaft speed and phase information are available). By order tracking the raw signal before feeding it into the model, the model will no longer have to capture the changes in frequency in the data, but only the amplitude modulation associated with varying speed conditions. The order-tracked signal can then be partitioned in the same manner as shown in Eq. (2-2). This approach is referred to as Order Tracking (OT) pre-processing. While this OT approach removes the varying frequencies from the measured data, the model will still need to learn to capture the phase shift between signal segments, which is caused by moving the partitioning window by

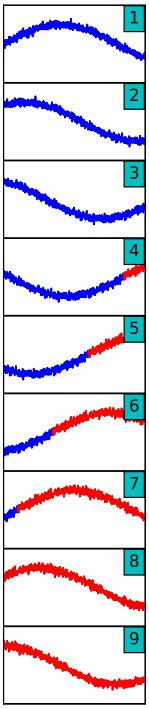
L_{sft}. To remove this phase shift from the data, a third pre-processing approach is proposed, referred to as Order Tracking and Aligning (OTA) pre-processing. This is done by aligning all the points corresponding to a certain shaft angle with each other.



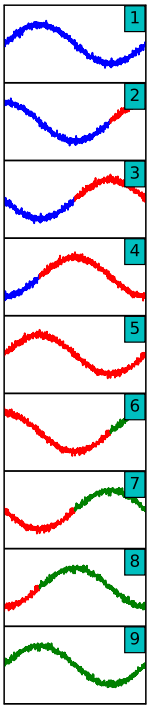
(a) Partitioned raw vibration signal



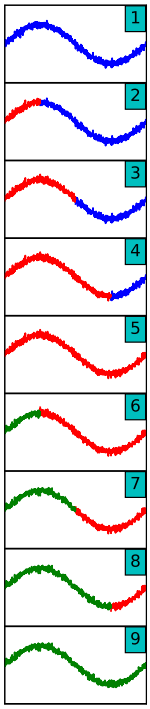
(b) Partitioned order tracked vibration signal



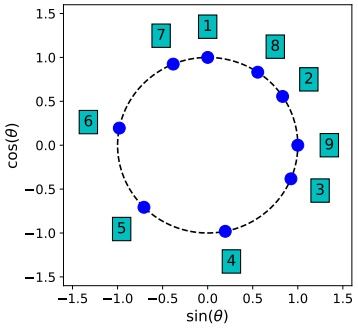
(c) SO: segments array



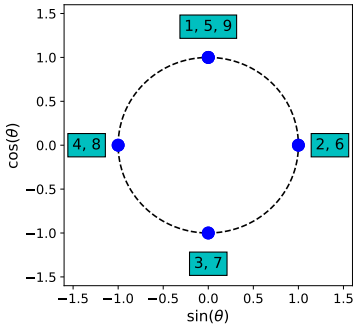
(d) OT: segments array



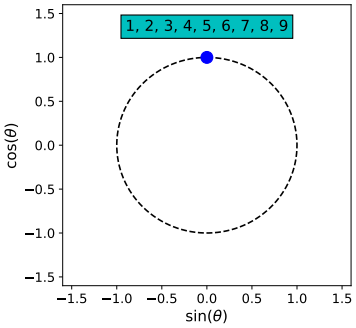
(e) OTA: segments array



(f) SO: phase of segments



(g) OT: phase of segments



(h) OTA: phase of segments

Figure 13: Differences between SO, OT and OTA pre-processing approaches

The differences between the three approaches are shown visually in Fig. 13, using the toy problem. A

raw, healthy signal shown in Fig. 13(a) was sampled at 1600 Hz, and order-tracked at 1000 samples per revolution, in Fig. 13(b). L_w is chosen as 1000 points (so the length of one rotation of the order tracked signal), and L_{sft} as 250 points. The first 9 segments from the raw signal and order tracked signal are taken, and stacked on top of one another, as was done in Eq. (2-2), using each of the three pre-processing techniques. The resulting arrays are shown visually in Fig. 13(c) for SO, Fig. 13(d) for OT, and Fig. 13(e) for OTA. The phase of the segments for each of the three approaches are shown in Fig.13(f)-(h), by plotting the sine and cosine of the phase on the x- and y-axis respectively.

The advantage of using OT or OTA can be seen in these phase plots. Because the shaft frequency in the toy problem is kept constant, each signal segment can be described with only the phase information. To learn the distribution of the healthy data, the model needs to learn a mapping from the signal segments to a latent representation of the phase information. To do this, the model must be presented with signal segments from all possible phase values, and ideally, the different phase values should be equally represented in the training data, as the model will otherwise be biased to capture data from certain phase values better than others. When implementing SO, the values for L_w and L_{sft} are chosen blindly, without a clear understanding of what the phase of each signal segment will be. The phases of the segments are scattered around the circle in Fig.13(f), and many more raw signals will have to be processed before phase information all around the circle is present in the processed data. There is also no way of knowing what phase values are more present in the data. The safest approach would be to make L_{sft} as small as possible, as this would ensure that there are data for all the phase values around the circle. The disadvantage of doing this is that the number of segments in the input data increases drastically, so training and evaluation of models will take much longer. The mapping from the signal segments, to a latent representation of the phase information, will also be highly nonlinear, so linear models might not accurately capture the phase shifts.

When using OT or OTA, and choosing L_{sft} so that L_w is divisible by L_{sft}, this problem is avoided. For OT, $\frac{L_w}{L_{sft}}$ phase values will be present in all the data, so the model needs to only learn a mapping from the signal segments to these $\frac{L_w}{L_{sft}}$ phase values. For OTA, the points in the segments are aligned based on the shaft angle, so the model needs to only learn a mapping from the signal segments to one phase value. This is a much simpler mapping between the data space and the latent space and might be easily captured by simple linear models such as PCA. L_{sft} can be chosen based on the amount of training data available. If limited training data is available, L_{sft} can be chosen to be smaller, but if healthy training data is freely available, L_{sft} can be chosen to be equal to L_w, which will speed up training and evaluation, without decreasing model performance. Note that when L_{sft} is chosen to be equal to L_w, the OT and OTA methods return the same signal segments.

Keep in mind that this toy problem is for constant shaft speed. If one then considers a problem with varying shaft frequency, one can easily imagine how much the complexity of the mapping from the signal segments to a latent representation of the phase and frequency information, will increase when using the SO method. When using the OT or OTA method, the mapping will only have to account for the amplitude modulation caused by varying speed conditions.

In the example shown, the window lengths L_w for the order-tracked approaches were chosen to be equal to one shaft rotation, so that the shaft rotations are aligned by the OTA approach, causing the signal segments to look relatively the same, without a lot of variance between them. For machinery containing gears or sprockets, the largest deterministic component in the signal is usually caused by the gear teeth meshing, with the component being periodic at order N_{teeth}, where N_{teeth} is the number of teeth on the gear or sprocket. In these cases, it might also make sense to use shorter window lengths associated with the period of one gear tooth completing a meshing cycle, and then align these shorter

windows, since the vibrations associated with each tooth will look relatively the same at healthy operating conditions. Heyns et al. (2012) followed a similar approach for gear diagnostics, where the window lengths were chosen to correspond with one pinion-gear meshing period, instead of a whole shaft rotation. Both these approaches are investigated in Chapters 4 and 5.

2.2.3 Normalization/ centering of data

For all the models, except for the PCA models, the condition monitoring signal data are normalized using the mean and standard deviation of the training set:

$$\boldsymbol{X}_{data} = \frac{\boldsymbol{X}_{data} - \mu_{\boldsymbol{x},train}}{\sigma_{\boldsymbol{x},train}} \tag{2-4}$$

where $\mu_{\boldsymbol{x},train}$ and $\sigma_{\boldsymbol{x},train}$ are the mean and standard deviation of the whole training set. Note that the normalization is not done per input dimension in the $\mathbb{R}^{L_w}$ input space, as this would bias the model to reconstruct more accurately on the input dimensions that have the smallest variance. When the inputs are normalized, the neural network-based models converge faster during training and are less likely to get stuck at local minima. For the PCA models, the condition monitoring signal sections are centred per input dimension, as will be discussed in Section 3.2.

The operating condition information corresponding to each of the N_{tw} signal segments stored in $\boldsymbol{X}_{data}$, is stored in an array $\boldsymbol{C} \in \mathbb{R}^{N_{tw} \times D_{\boldsymbol{c}}}$ where $D_{\boldsymbol{c}}$ is the number of operating conditions known. In the investigations performed in this work, only the shaft speed and phase information are available. The sine and cosine of the shaft phase θ are fed into models so that the models can learn that there is not a discontinuity in the input space at $\theta = 0$ and $\theta = 2\pi$, but that these values represent the same phase. Along with the shaft speed, this gives $\boldsymbol{c}$ a dimensionality $D_{\boldsymbol{c}} = 3$:

$$\boldsymbol{c}_i = \begin{bmatrix} f_{shaft,i} & \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \tag{2-5}$$

The operating condition array $\boldsymbol{C}$ is normalized per input dimension in the $\mathbb{R}^{D_C}$ input space, before being fed into the models:

$$\boldsymbol{C} = \frac{\boldsymbol{C} - \boldsymbol{\mu}_{\boldsymbol{c},train}}{\boldsymbol{\sigma}_{\boldsymbol{c},train}} \tag{2-6}$$

where $\boldsymbol{\mu}_{\boldsymbol{c},train} \in \mathbb{R}^{D_{\boldsymbol{c}}}$ and $\boldsymbol{\sigma}_{\boldsymbol{c},train} \in \mathbb{R}^{D_{\boldsymbol{c}}}$ is the per dimension mean and standard deviation of the training set.

If the operating conditions are known, the training and validation sets can be chosen so that the different possible operating condition combinations are all presented relatively equally in the training data. If this is not done, the model will be biased to reconstruct data from certain operating conditions better.

2.3 Modelling: Training and Evaluation

After pre-processing, the healthy input data are used to train the models. The models can be categorized as unsupervised or semi-supervised (where operating condition information is used during training).

The distributions that the models learn that can be used to generate discrepancy signals are now listed. The unsupervised models all return the reconstruction distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{z})$, the semi-supervised models return reconstruction distributions of the form $p(\boldsymbol{x}_{healthy}|\boldsymbol{z}, \boldsymbol{c})$ and the C-decoder

model returns $p(\boldsymbol{x}_{healthy}|\boldsymbol{c})$. These distributions can be used to get reconstruction space discrepancy signals. The semi-supervised latent variable models also learn the distribution $p(\boldsymbol{z}_{healthy}|\boldsymbol{c})$, which is used to get latent space discrepancy signals. For the unsupervised latent variable models, the latent space is not monitored in this work.

All the distributions listed are modelled with Gaussian distributions with diagonal covariance matrices. Each Gaussian is thus parameterized by a mean vector $\boldsymbol{\mu}$ and a diagonal variance vector $\boldsymbol{\sigma}^2$. Balshaw et al. (2022) highlighted that Gaussian distributions will be less effective when the healthy signals include impulsive components. In these cases the data distributions will be heavy-tailed, and Balshaw et al. (2022) proposed that implicit density estimation techniques can be used to overcome this challenge. This work however only considers the explicit density estimation technique where all distributions are assumed to be Gaussian. A dataset that includes impulsive components is investigated in Chapter 5, to see how this Gaussian assumption holds.

After training, new data from unknown health conditions can be fed into the model, with the models then returning sets of $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$ vectors that represent the above-mentioned distributions, for each input signal $\boldsymbol{x}_i$. The intricacies of how the models learn to capture the above-mentioned distributions are not presented in this section so that the reader can first get an overview of the whole discrepancy analysis framework. The model formulations and implementations are documented in detail in Chapter 3.

2.4 Post-processing: discrepancy signals

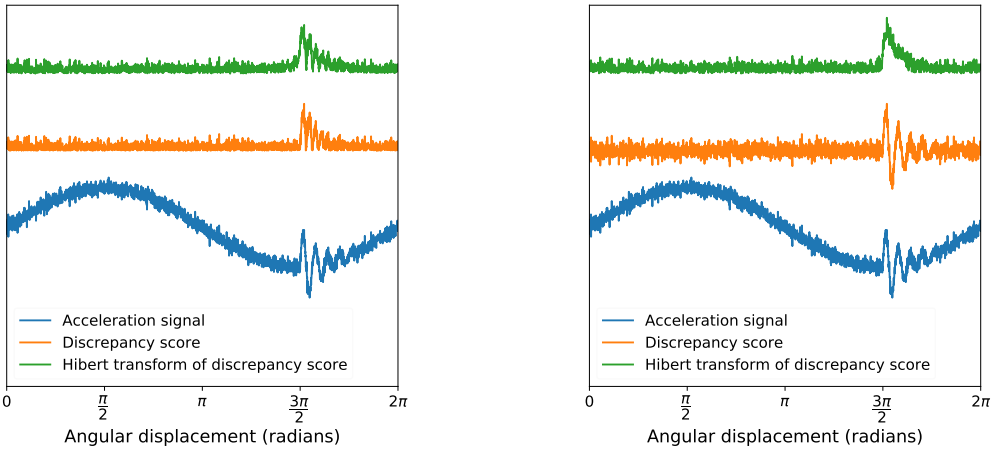
2.4.1 Discrepancy signals from the reconstruction space

To get discrepancy signals from the reconstruction space, it is evaluated how likely it is that a sample $\boldsymbol{x}_i$ is from one of the reconstruction distributions $p(\boldsymbol{x}_{healthy}|\boldsymbol{z})$, $p(\boldsymbol{x}_{healthy}|\boldsymbol{z},\boldsymbol{c})$ or $p(\boldsymbol{x}_{healthy}|\boldsymbol{c})$. With the model only being trained on healthy data, the idea is that the reconstruction of $\boldsymbol{x}_i$ will be accurate when $\boldsymbol{x}_i$ is recorded from a healthy machine, and less accurate if $\boldsymbol{x}_i$ is from a damaged machine. The Mahalanobis distance works well as a discrepancy measure in this case, as it captures how many standard deviations the point is from the mean. This ensures that the discrepancy score is not correlated with the size of the reconstruction variance $\boldsymbol{\sigma}_i^2$, as is the case if one uses the mean squared error (MSE) or likelihood as a discrepancy measure. Balshaw et al. (2022) calculated the Mahalanobis distance for the whole signal segment $\boldsymbol{x}_i$, viewing the reconstruction mean and variance as forming a multivariate distribution. While this makes discrepancy scores less sensitive to noise, the downside is that the resolution of the discrepancy signal is influenced by the lengths chosen for L_w and L_{sft}. In this work, it is proposed that a point-wise discrepancy signal for $\boldsymbol{x}_i$ is calculated from the reconstructed vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$, giving a discrepancy signal $\boldsymbol{d}_{recon,i}$. Each point in the discrepancy signal can be calculated as:

$$d_{recon,i,j} = \frac{x_{i,j} - \mu_{i,j}}{\sigma_{i,j}} \tag{2-7}$$

with $j \in [1, 2, ..., L_w]$. Note that the sign of the difference between the predicted mean and the sample is kept (taking the absolute value of Eq. (2-7) will give the Mahalanobis distance). This is done so that the frequency information of the discrepancy is captured. Since most faults cause cyclo-stationary impulses in the measured signal, it is common in signal processing approaches to take the absolute value of the Hilbert transform of the signal as an envelope analysis technique (Randall & Antoni 2011, Hou et al. 2022), so that the fault frequencies can be extracted from the signal, not the high natural frequencies of the structure that vibrate because of these impulses. Using Eq. (2-7) to calculate discrepancy signals, will also result in the high natural frequencies of the structure to be

captured in the signal. By then taking the absolute value of the Hilbert transform of the discrepancy signal $\boldsymbol{d}_{recon,i}$, the fault frequencies can be identified in the discrepancy signal. Fig. 14 shows why the sign of the difference between the predicted mean and the sample is kept when calculating the point-wise discrepancy, as this results in the Hilbert transform capturing the fault frequencies more cleanly during the enveloping process.



(a) Mahalanobis distance

(b) Proposed discrepancy measure

Figure 14: Difference in the envelope of the reconstruction space discrepancy signal when using the Mahalanobis distance vs. the proposed discrepancy measure

2.4.2 Discrepancy signals from the latent space

To get discrepancy signals from the latent space, it is evaluated how likely it is that the latent representation $\boldsymbol{z}_i$ of a sample $\boldsymbol{x}_i$ is from the distribution $p(\boldsymbol{z}_{healthy}|\boldsymbol{c})$. The Mahalanobis distance is again used to calculate a discrepancy score, however this time it doesn't make sense to calculate the discrepancy point-wise, as it is not known which latent variables describe a specific point in the signal segment. The effect of time over the signal segment $\boldsymbol{x}_i$ is lost in the latent representation $\boldsymbol{z}_i$. Every point in the discrepancy signal $\boldsymbol{d}_{latent,i}$ for that specific segment $\boldsymbol{x}_i$ has the same value:

$$d_{latent,i,j} = \sum_{j}^{L_w} \frac{(x_{i,j} - \mu_{i,j})^2}{\sigma_{i,j}^2} \tag{2-8}$$

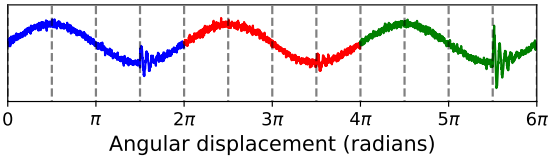
which is the squared Mahalanobis distance for that signal segment. The resolution in a latent space discrepancy signal is thus lower than for the reconstruction space discrepancy signal. It also does not make sense to take the Hilbert transform of $\boldsymbol{d}_{latent,i}$, since the Mahalanobis distance is always positive, and already acts as an enveloping technique on the calculated discrepancy signal.

2.4.3 Combining discrepancy signals from consecutive signal segments

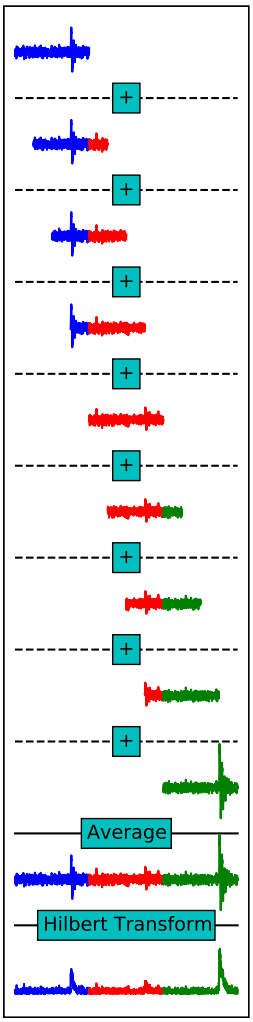
After calculating a discrepancy signal of length L_w for each input signal segment, the discrepancy signals are combined to form one long discrepancy signal that corresponds with the original raw signal before it was partitioned. The advantage of doing this is simply that working with longer discrepancy signals will make any statistical features extracted from these signals more robust to the presence of noise. Fig. 15 shows how consecutive discrepancy signals are combined, using the toy problem. The discrepancy signal points corresponding with a specific point in the raw signal are simply averaged. After averaging, the Hilbert transform is applied to the reconstruction space dis-

crepancy signals. Note that the Hilbert transform is not applied to the latent space discrepancy signals.

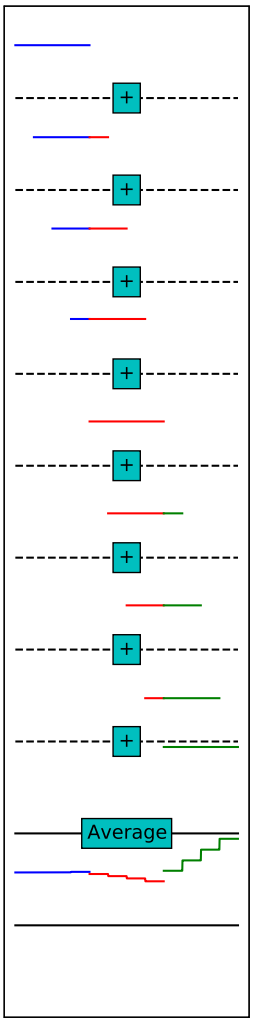
For illustration purposes, the damage impulses in the raw signal were scaled, with the second impulse being smaller, and the third impulse larger. Note how the resolution of the reconstruction space discrepancy signal is much higher than for the latent space discrepancy signal.



(a) Raw signal from damaged state



(b) Recon space



(c) Latent space

Figure 15: Combining discrepancy signals from consecutive signal segments (reconstruction space and latent space discrepancy signals)

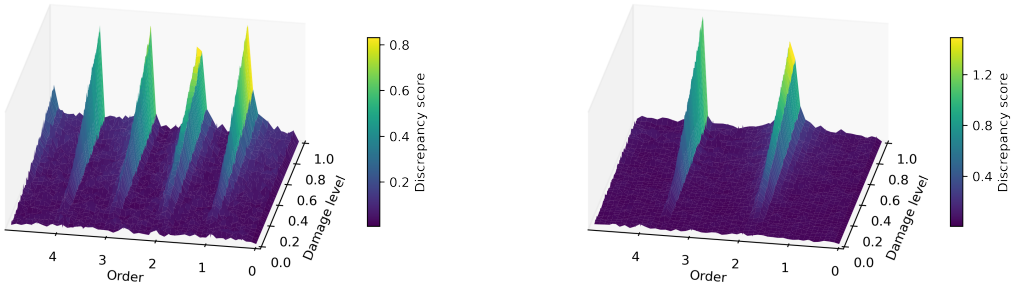
2.5 Fault classification

After post-processing of the discrepancy signals, one has point-wise discrepancy signals corresponding to each of the raw condition monitoring signals before partitioning, which can now be used to

perform fault detection, -tracking and classification tasks. Fault classification is not an easy task in the time domain, as the signals are noisy, and it is difficult to see what fault frequencies are present. It is also difficult to classify faults in the frequency domain under varying shaft speeds, as the frequency spectra get smeared when the signals are not order-tracked. Therefore, it is necessary to order track the discrepancy signals to be able to classify faults, so the shaft speed and phase information are required.

With the OTA and OT pre-processing methods, these discrepancy signals will already be order-tracked. With the SO pre-processing method, the discrepancy signals can be order-tracked as a post-processing step, as was done by Balshaw et al. (2022).

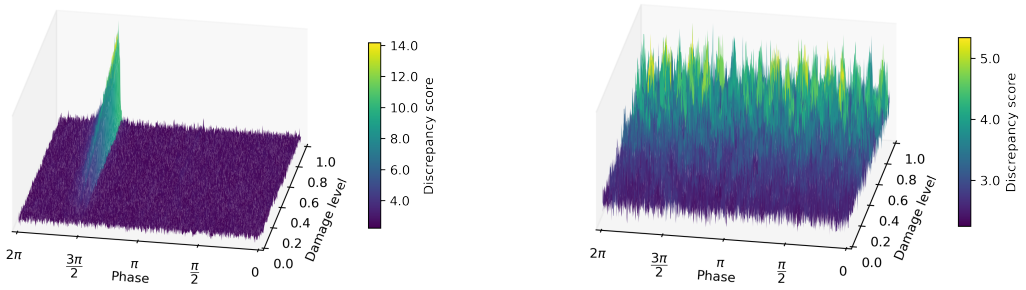
By plotting the order spectra and synchronous averages of the discrepancy signals, as they develop over time, one can clearly see which signals are from a machine with a 'rotation angle correlated'-fault (at 1 Hz/f_{shaft}), and which are from a machine with a 'rotation angle uncorrelated'-fault (at 1.8 Hz/f_{shaft}). Fig. 16 shows how the order spectra of the discrepancy signals for both fault types develop as the damage level increases. The fault frequencies and their harmonics can be clearly seen. To confirm that the order spectra content at 1 order is for a localized fault relative to the shaft angle, one can look at the synchronous average of the discrepancy signal. Fig. 17 shows how the synchronous averages of the discrepancy signals for both fault types develop as the damage level increases. For the fault at 1 Hz/f_{shaft}, one can clearly identify the fault at $\frac{3\pi}{2}$ radians. For the fault at 1.8 Hz/f_{shaft}, the noise level in the synchronous average only increases, showing that there is no fault present that is synchronous with the shaft rotation.



(a) 1 Hz/f_{shaft}

(b) 1.8 Hz/f_{shaft}

Figure 16: Order spectra of the reconstruction space discrepancy signals at increasing damage levels for faults at 1 Hz/f_{shaft} and 1.8 Hz/f_{shaft}

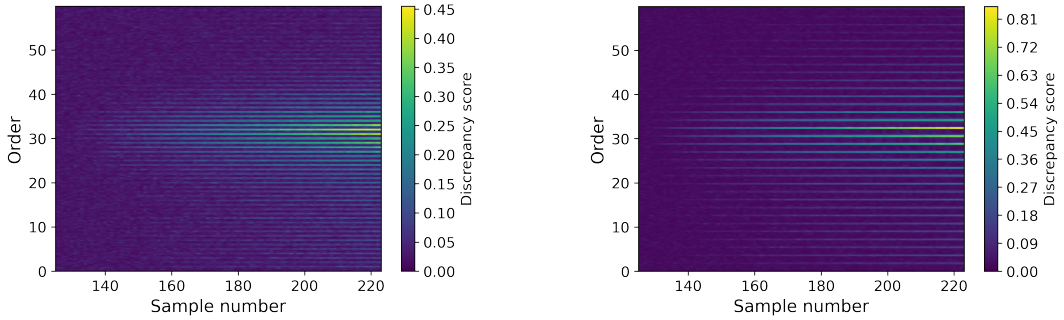


(a) 1 Hz/f_{shaft} (b) 1.8 Hz/f_{shaft}

Figure 17: Synchronous average of the reconstruction space discrepancy signals at increasing damage levels for faults at 1 Hz/f_{shaft} and 1.8 Hz/f_{shaft}

In both the analyses performed in this work, these order spectra and synchronous average plots are presented, to show how the different faults can be identified from these two plots. Note that no classification models were implemented in this work.

It is also useful to investigate the order spectra plots of the reconstruction space discrepancy signals before the envelopes are extracted. While this is not really useful for classification or fault tracking, it indicates which frequencies the model doesn't reconstruct well. These plots can be investigated to see if the model struggles to capture some of the frequencies present in the healthy data, which will lead to poor model performance. While the healthy component is reconstructed well for the toy problem, one can see in Fig. 18 that the model used does not reconstruct the frequency content caused by the fault at order 32.



(a) 1 Hz/f_{shaft} (b) 1.8 Hz/f_{shaft}

Figure 18: Order spectra of the reconstruction space discrepancy signals (without the Hilbert transform) at increasing damage levels for faults at 1 Hz/f_{shaft} and 1.8 Hz/f_{shaft}

2.6 Fault detection and tracking

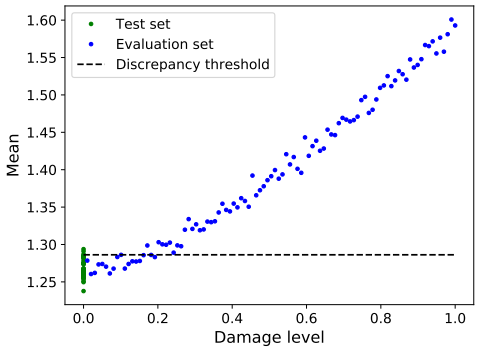
To detect and track the development of faults, three simple statistical features from the discrepancy signals are used: the mean, variance and kurtosis. It is expected that the kurtosis will be the most responsive to impulsive faults, but also the least robust to noise or any healthy frequency content that the model struggles to capture.

Along with these three statistical features, two other features are used in the case where the shaft speed and phase information are known:

- Order Spectra Tracking (OST) feature: If the fault frequency is known, the order-tracked discrepancy signal can be monitored in the frequency domain, monitoring only the order associated with the fault. For a 'rotation angle correlated'-fault, order 1 can be monitored.
- Synchronous Average Tracking (SAT) feature: For a 'rotation angle correlated'-fault, the angle corresponding to the fault can be monitored on the synchronous average.

Note that in this work it is assumed that the fault frequencies, and the angle corresponding to localized faults, are known. In reality, one would have to monitor all orders and all angles, and isolate regions of interest once a fault starts to develop and the discrepancy threshold is crossed at some specific order (for OST) or angle (for SAT).

The test set is used to set the discrepancy thresholds. Throughout this work, the 95^{th} percentile of a discrepancy signal feature of the healthy test set, is used as the discrepancy threshold. It is thus expected that the models will classify about 5% of healthy data incorrectly as damaged data.
Fig. 19 shows how the mean of the discrepancy signals can be tracked. The other discrepancy signal features can be tracked in the same manner.



(a) Mean

Figure 19: Fault tracking: Discrepancy signal mean vs. damage level

For the models where the latent space discrepancies and reconstruction space discrepancies are monitored, the two approaches are combined by simply classifying a sample as damaged data if one or both of the two discrepancy thresholds are exceeded.

This methodology is used throughout this work. The success of this approach depends greatly on how well the distributions $p(\boldsymbol{x}_{healthy}|\boldsymbol{z})$, $p(\boldsymbol{x}_{healthy}|\boldsymbol{z},\boldsymbol{c})$, $p(\boldsymbol{x}_{healthy}|\boldsymbol{c})$ and $p(\boldsymbol{z}_{healthy}|\boldsymbol{c})$ can be modelled with the PCA and VAE frameworks used in this work. The following chapter shows how these distributions were learned.

3 Latent Variable Models: Formulations, Training Challenges and Implementation

Six models are implemented in this work: a C-decoder, three variations of PCA, a VAE model and an Identifiable Variational-Autoencoder (iVAE) model. All the models, except the C-decoder model, have an encoder-decoder structure, where the input data are first mapped to a lower dimensional latent space, before being reconstructed back to the data space. The C-decoder model is the only model that is not a latent variable model, taking only $\boldsymbol{c}$ as input and reconstructing $\boldsymbol{x}_{healthy}$, making it a supervised model. The PCA models differ in how the standard deviation vector $\boldsymbol{\sigma}_{\boldsymbol{x}}$ is reconstructed from the input data. The VAE is trained in an unsupervised setting without operating condition information. The iVAE model is used to incorporate the available operating condition information into the VAE framework, making it a semi-supervised model.

This chapter presents the mathematical formulations and implementations of the models implemented in this work. The chapter also highlights limitations to the PCA and VAE frameworks for capturing vibration data measured at varying operating conditions in an unsupervised setting, and shows how these limitations can be addressed by incorporating the available operating condition information into the modelling process. The chapter ends with a discussion of how the latent space sizes were chosen in this work, along with a summary of how the models fit into the unsupervised and semi-supervised frameworks respectively.

The model architectures are described in this chapter, and a schematic representation of each model is provided (i.e Fig. 20 for the C-decoder model). In these schematics, the green boxes represent data arrays, with the dimensionality of the array specified above the green box. The number of signal segments is given as N, the dimensionality of the operating conditions as $D_{\boldsymbol{c}}$, the dimensionality of the condition monitoring segments as $D_{\boldsymbol{x}}$ (which is equal to L_w), and the dimensionality of the latent space as $D_{\boldsymbol{z}}$. The blue trapezium boxes represent models with either encoder or decoder structures, depending on whether the dimensionality of the array fed into the model is decreased or increased.

3.1 Supervised C-decoder

The C-decoder model is used as a baseline for the semi-supervised setting, showing how much of the variance in the healthy condition monitoring signals can be explained with only the known operating condition information, without using latent variables. The C-decoder model is a simple nonlinear model, that learns to reconstruct $\boldsymbol{x}_{healthy}$ by taking only $\boldsymbol{c}$ as input. The model learns the distribution $p(\boldsymbol{x}_{healthy}|\boldsymbol{c})$, which is parameterized to be Gaussian. The model is trained to return the reconstruction mean $\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{c}}$ and log-variance $\log \boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{c}}$ that will maximize the likelihood of $p(\boldsymbol{x}_{healthy}|\boldsymbol{c})$. The reconstruction log-variance is learned, to prevent the model from returning negative variance terms. The decoder is a deconvolutional neural network and has the same architecture as used for the decoder in the VAE and iVAE models. The decoder architecture is documented in Appendix C. Fig. 20 shows a schematic representation of the C-decoder model. The reconstruction space is used to calculate discrepancy signals.

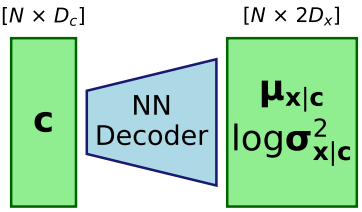


Figure 20: Schematic representation of C-decoder model

3.2 Principal Component Analysis

PCA is a linear latent variable model. Each sample in the input data matrix $\boldsymbol{X}$ is orthogonally projected onto a lower dimensional linear latent space, that maximizes the variance of the projected data. The maximum variance formulation of PCA as given in Bishop (2006), is summarized in this subsection. The deterministic form of PCA is presented first, and thereafter it is extended to a probabilistic framework. The challenge of capturing data with heteroscedastic noise using a probabilistic PCA framework is explained, followed by the documentation of the PCA models that are implemented in this work.

3.2.1 Deterministic PCA

The goal is to project the input data samples $\boldsymbol{X}$ (with size $N \times D_{\boldsymbol{x}}$) to a lower dimensional latent representation $\boldsymbol{Z}$ (with size $N \times D_{\boldsymbol{z}}$), using only linear transformations.

First consider projecting the input $\boldsymbol{X}$ onto a one-dimensional latent space ($D_{\boldsymbol{z}} = 1$), using a vector $\boldsymbol{u}_1$. Each input vector $\boldsymbol{x}_n$ is projected to a single point in the latent dimension, $z_n = \boldsymbol{u}_1^T \boldsymbol{x}_n$. The variance of the projected data $\boldsymbol{z}$ must now be maximized. The variance can be written as:

$$\sigma_{\boldsymbol{z}}^2 = \frac{1}{N}\sum_{n=1}^{N}(z_n - \mu_z)^2 \tag{3-1}$$

$$= \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{u}_1^T \boldsymbol{x}_n - \boldsymbol{u}_1^T \boldsymbol{\mu}_{\boldsymbol{x}})^2 \tag{3-2}$$

$$= \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 \tag{3-3}$$

where μ_z is the mean of the scalar z values, $\boldsymbol{\mu}_{\boldsymbol{x}}$ is the feature mean of the input data, and $\boldsymbol{S}$ is the covariance matrix of the input data, given as:

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu}_{\boldsymbol{x}})(\boldsymbol{x}_n - \boldsymbol{\mu}_{\boldsymbol{x}})^T \tag{3-4}$$

Simply maximizing the projected variance $\sigma_{\boldsymbol{z}}^2 = \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1$ with respect to $\boldsymbol{u}_1$, will cause the norm of $\boldsymbol{u}_1$ to tend to infinity. To prevent this, the transformation vector is constrained to be a unit vector, so that $\boldsymbol{u}_1^T \boldsymbol{u}_1 = 1$. To add this constraint to the maximization of the variance, a Lagrange multiplier λ_1 is used to get an unconstrained objective function, given as:

$$\boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^T \boldsymbol{u}_1) \tag{3-5}$$

The critical points can be found by taking the derivative of the objective function with respect to $\boldsymbol{u}_1$, and setting it equal to zero:

$$\boldsymbol{S}\boldsymbol{u}_1 - \lambda_1 \boldsymbol{u}_1 = 0 \tag{3-6}$$

$$\boldsymbol{S}\boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1 \tag{3-7}$$

This shows that $\boldsymbol{u}_1$ must be an eigenvector of $\boldsymbol{S}$. Left-multiplying by $\boldsymbol{u}_1^T$ gives:

$$\boldsymbol{u}_1^T \boldsymbol{S}\boldsymbol{u}_1 = \lambda_1 = \sigma_{\boldsymbol{z}}^2 \tag{3-8}$$

showing that the variance will be a maximum when $\boldsymbol{u}_1$ is set equal to the eigenvector that has the largest eigenvalue λ_1. This eigenvector is known as the first principal component. Since the covariance matrix $\boldsymbol{S}$ is symmetric, the eigenvalues will all be real, and the corresponding eigenvectors are all orthogonal. To find the next principal component, the process can be repeated, finding the direction that maximizes the variance of the projected inputs, but each new principal component to be found has the additional constraint that it should be orthogonal to all the previously found principal components. The linear transformation matrix $\boldsymbol{U}$, with size $D_{\boldsymbol{x}} \times D_{\boldsymbol{z}}$ then consists of the first $D_{\boldsymbol{z}}$ principal components.

Using $\boldsymbol{U}$, the input data $\boldsymbol{X}$ can be encoded to a lower dimensional latent representation $\boldsymbol{Z}$:

$$\boldsymbol{z}_n = \boldsymbol{U}^T(\boldsymbol{x}_n - \boldsymbol{\mu}_{\boldsymbol{x}}) \tag{3-9}$$

The input data $\boldsymbol{X}$ can then be reconstructed/decoded from this latent representation:

$$(\boldsymbol{x}_n - \boldsymbol{\mu}_{\boldsymbol{x}})_{\text{recon}} = \boldsymbol{U}\boldsymbol{z}_n \tag{3-10}$$

where $(\boldsymbol{x}_n - \boldsymbol{\mu}_{\boldsymbol{x}})_{\text{recon}}$ is the reconstruction of the zero-centred input data.

Fig. 21 shows a schematic representation of the standard deterministic PCA model.

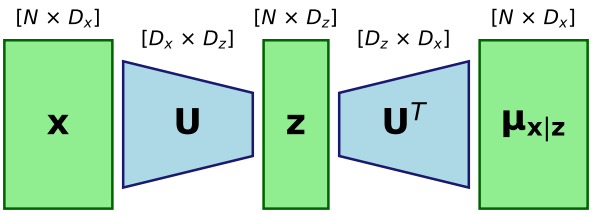


Figure 21: Schematic representation of the deterministic PCA model

The size of the latent space, $D_{\boldsymbol{z}}$, can be chosen based on the percentage of the variance that one wants to preserve during reconstruction, where the percentage of the variance captured is given by the cumulative contribution rate (CCR):

$$\text{CCR} = \frac{\sum_{i=1}^{D_{\boldsymbol{z}}} \lambda_i}{\sum_{i=1}^{D_{\boldsymbol{x}}} \lambda_i} \tag{3-11}$$

where λ_i is the eigenvalue associated with the i^{th} principal component of the covariance matrix of the input data.

The main limitation of PCA is that it uses a linear function to transform the input data to the latent space, as well as a linear function to reconstruct the input data from the latent representation. If the high-dimensional input data has a highly nonlinear relationship with the underlying generative factors of the data, many principal components will be necessary to capture most of the variance in the data, while a nonlinear mapping to and from the latent space might only require a small latent space dimensionality to capture most of the variance in the input data.

In this context of discrepancy analysis, we want to be able to compare how different models fit the data using discrepancy measures. To do this, we have to extend PCA to a probabilistic framework.

3.2.2 Probabilistic PCA

PCA can be formulated as a probabilistic model (Tipping & Bishop 1999), where the latent prior $p(\boldsymbol{z})$ and the generative distribution $p(\boldsymbol{x}|\boldsymbol{z})$ are both Gaussian. Each input data sample $\boldsymbol{x}$ is viewed as the linear transformation of the associated latent variables $\boldsymbol{z}$, along with additive Gaussian noise:

$$\boldsymbol{x} = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}_{\boldsymbol{x}} + \boldsymbol{\epsilon} \tag{3-12}$$

where the noise $\boldsymbol{\epsilon}$ is sampled from $\mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$. Note that the noise is sampled from a Gaussian with constant isotropic variance, so the noise is not a function of the latent representation $\boldsymbol{z}$.
The latent prior is given as a zero-mean unit variance Gaussian:

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I}) \tag{3-13}$$

The generative distribution is given as:

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}_{\boldsymbol{x}}, \sigma^2\boldsymbol{I}) \tag{3-14}$$

The maximum likelihood solutions for $\boldsymbol{\mu}_{\boldsymbol{x}}$, $\boldsymbol{W}$ and σ^2 can be obtained explicitly. The maximum likelihood estimate for $\boldsymbol{\mu}_{\boldsymbol{x}}$ is simply the mean of the data $\boldsymbol{X}$. The maximum likelihood estimate for σ^2 can be calculated as:

$$\sigma^2_{ML} = \frac{1}{D_{\boldsymbol{x}} - D_{\boldsymbol{z}}} \sum_{i=D_{\boldsymbol{z}}+1}^{D_{\boldsymbol{x}}} \lambda_i \tag{3-15}$$

where λ_i is the eigenvalue associated with the i^{th} principal component of the covariance matrix of the input data. σ^2_{ML} can thus be seen as the average variance that the discarded principal components between $D_{\boldsymbol{z}}$ and $D_{\boldsymbol{x}}$ would have captured. The maximum likelihood estimates for $\boldsymbol{W}$ can be calculated as:

$$\boldsymbol{W}_{ML} = \boldsymbol{U}_q(\boldsymbol{L}_q - \sigma^2\boldsymbol{I})^{1/2} \tag{3-16}$$

where q is the number of principal components used, so $q = D_{\boldsymbol{z}}$. $\boldsymbol{L}_q$ is a $q \times q$ diagonal matrix containing the eigenvalues $\lambda_1, ..., \lambda_q$ on the diagonal, and $\boldsymbol{U}_q$ the $D_{\boldsymbol{x}} \times q$ matrix containing the first q principal components.

The posterior for the latent variable $p(\boldsymbol{z}|\boldsymbol{x})$ is given as:

$$p(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{M}^{-1}\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}}), \sigma^2\boldsymbol{M}^{-1}) \tag{3-17}$$

where

$$\boldsymbol{M} = \boldsymbol{W}^T\boldsymbol{W} + \sigma^2\boldsymbol{I} \tag{3-18}$$

3.2.3 Capturing heteroscedastic noise

Having a model with zero-centred noise and constant isotropic variance presents a problem to the application of capturing vibration data from varying speed conditions. As was discussed in Section 1.3, the vibration data will have highly heteroscedastic noise, with the variance of the noise increasing as the operating speed increases. Using a constant variance σ^2 will result in data from slow operating speeds being reconstructed with a variance term that is too large, and data from fast operating speeds, with a variance term that is too small.

Another model closely related to PCA, namely factor analysis, accounts for heteroscedasticity between the data points in $\boldsymbol{x}$, by modelling the covariance in Eq. (3-14) with a diagonal matrix, rather than an isotropic covariance (Bishop 2006). This is however still a constant covariance matrix, applicable to the whole dataset, so the noise is still independent of the latent representation $\boldsymbol{z}$, and will not capture the heteroscedasticity of the vibration data accurately. Hong et al. (2019) proposed a PCA variant that incorporates heteroscedasticity into the model, which allows noise variances to vary across samples. The variance is however assumed to be constant over the sample, so the model cannot capture second-order cyclostationary components. The model also assumes that the variance per sample is known or can be easily estimated.

It would be ideal if the variance can be modelled as a function of the operating speed or as a function of $\boldsymbol{z}$, as it is expected that $\boldsymbol{z}$ will be a proxy for the operating conditions present in each vibration sample $\boldsymbol{x}$. To address this problem, the probabilistic PCA model is adapted in this work to learn the variance as a function of either $\boldsymbol{c}$ or $\boldsymbol{z}$, depending on whether the operating speed information is available or not.

3.2.4 PCA implementations

Three different PCA variations were implemented in this work:

- PPCA: This is the normal probabilistic PCA framework documented in Section 3.2.2, which assumes a constant variance across the whole dataset.
- PCA-Z-LR: The variance is learned per sample by fitting the conditional distribution $p(\boldsymbol{e}|\boldsymbol{z})$ using a linear regression model, where $\boldsymbol{e}$ is the difference between the input data sample $\boldsymbol{x}$ and the reconstruction $\boldsymbol{x}_{recon}$. Because the latent representation $\boldsymbol{z}$ is used, the model falls in the unsupervised setting.
- PCA-C-LR: The variance is learned per sample by fitting the conditional distribution $p(\boldsymbol{e}|\boldsymbol{c})$ using a linear regression model. Because the available operating condition information $\boldsymbol{c}$ is used, the model falls in the semi-supervised setting.

With the PPCA implementation, an input sample $\boldsymbol{x}$ is projected to a latent space representation, using the mean of the posterior latent distribution given in Eq. (3-17), and then reconstructed with Eq. (3-14) (as shown in Fig. 22). The variance in the input data is assumed to be isotropic, with the variance calculated using Eq. (3-15).

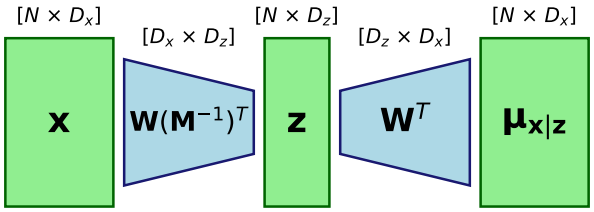


Figure 22: Schematic representation of PPCA model

For the PCA-Z-LR and PCA-C-LR models, the input data samples are first projected to the latent space, and then reconstructed, using the normal deterministic PCA framework (as shown in Fig. 21). The reconstruction difference $\boldsymbol{e}$ can then be calculated as:

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}} \tag{3-19}$$

To capture the variance in the data per sample, a conditional Gaussian distribution is then fit to the reconstruction differences, with mean $\boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{z}}$ or $\boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{c}}$ and standard deviation $\boldsymbol{\sigma}_{\boldsymbol{e}|\boldsymbol{z}}$ or $\boldsymbol{\sigma}_{\boldsymbol{e}|\boldsymbol{c}}$, depending on whether the Gaussian is conditioned on the latent representations (PCA-Z-LR) or the operating conditions (PCA-C-LR). The predicted reconstruction mean can then be improved as:

$$\boldsymbol{\mu}^*_{\boldsymbol{x}|\boldsymbol{z}} = \boldsymbol{\mu}_{\boldsymbol{x}} - \boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{z}} \qquad \text{or} \qquad \boldsymbol{\mu}^*_{\boldsymbol{x}|\boldsymbol{z}} = \boldsymbol{\mu}_{\boldsymbol{x}} - \boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{c}} \tag{3-20}$$

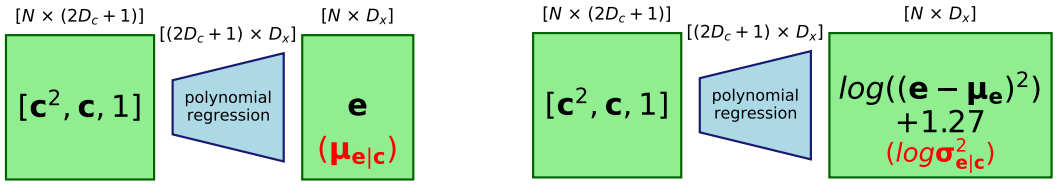
where $\boldsymbol{\mu}^*_{\boldsymbol{x}|\boldsymbol{z}}$ is the improved reconstruction mean, and the standard deviation vector is then given as:

$$\boldsymbol{\sigma}_{\boldsymbol{x}|\boldsymbol{z}} = \boldsymbol{\sigma}_{\boldsymbol{e}|\boldsymbol{z}} \qquad \text{or} \qquad \boldsymbol{\sigma}_{\boldsymbol{x}|\boldsymbol{z}} = \boldsymbol{\sigma}_{\boldsymbol{e}|\boldsymbol{c}} \tag{3-21}$$

Second-order polynomial regression models are used to fit the mean and variance of the reconstruction difference distribution and are trained using the least squares method. The reconstruction difference mean $\boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{z}}$ or $\boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{c}}$ can be predicted by simply mapping the inputs $\boldsymbol{z}$ or $\boldsymbol{c}$ to the difference $\boldsymbol{e}$, while the logarithm of the reconstruction difference variance $\log(\boldsymbol{\sigma}^2_{\boldsymbol{e}|\boldsymbol{z}})$ or $\log(\boldsymbol{\sigma}^2_{\boldsymbol{e}|\boldsymbol{c}})$ can be learned by mapping the inputs to $\log\left((\boldsymbol{e} - \boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{z}})^2\right) + 1.27$ or $\log\left((\boldsymbol{e} - \boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{c}})^2\right) + 1.27$ respectively, since

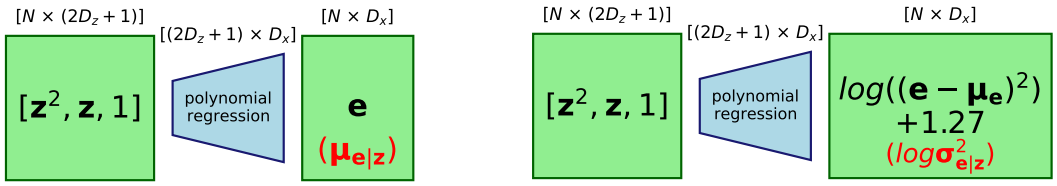
$$\log(\boldsymbol{\sigma}^2_{\boldsymbol{e}|\boldsymbol{k}}) \approx \mathbb{E}\left(\log\left((\boldsymbol{e} - \boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{k}})^2\right)\right) + 1.27 \tag{3-22}$$

where $\boldsymbol{k}$ is just a placeholder for $\boldsymbol{z}$ or $\boldsymbol{c}$. The derivation for Eq. (3-22) is presented in Appendix B. The log-variance is learned, to prevent the model from returning negative variances. Figs. 23 and 24 show schematic representations of the reconstruction difference fitting approaches for the PCA-C-LR and PCA-Z-LR models respectively.



(a) $\boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{c}}$ (b) $\log(\boldsymbol{\sigma}^2_{\boldsymbol{e}|\boldsymbol{c}})$

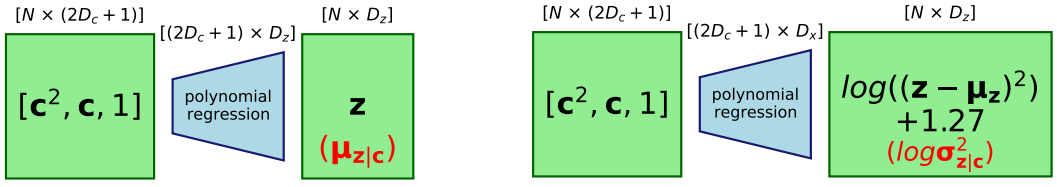
Figure 23: PCA-C-LR: Reconstruction difference fitting, for $\boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{c}}$ and $\log(\boldsymbol{\sigma}^2_{\boldsymbol{e}|\boldsymbol{c}})$



(a) $\boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{z}}$ (b) $\log(\boldsymbol{\sigma}^2_{\boldsymbol{e}|\boldsymbol{z}})$

Figure 24: PCA-Z-LR: Reconstruction difference fitting, for $\boldsymbol{\mu}_{\boldsymbol{e}|\boldsymbol{z}}$ and $\log(\boldsymbol{\sigma}^2_{\boldsymbol{e}|\boldsymbol{z}})$

For all three PCA models, the reconstruction space is used to calculate discrepancy signals. For the PCA-C-LR model, the latent space is also monitored, since the operating condition information is available. This is done by fitting a conditional Gaussian distribution $p(\boldsymbol{z}|\boldsymbol{c})$ in the same manner as the reconstruction difference distributions were fitted. The Mahalanobis distance between $\boldsymbol{z}$ and $p(\boldsymbol{z}|\boldsymbol{c})$ is calculated using Eq. (2-8). Fig. 25 shows a schematic representation of fitting $p(\boldsymbol{z}|\boldsymbol{c})$ for the PCA-C-LR.



(a) $\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{c}}$ (b) $\log(\boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{c}})$

Figure 25: PCA-C-LR: fitting $p(\boldsymbol{z}|\boldsymbol{c})$

3.3 Variational Auto-Encoders

VAEs are nonlinear latent variable models, with the encoder and decoder both parameterized using neural networks. The VAE uses variational inference to infer the values of $\boldsymbol{z}$ given a certain sample $\boldsymbol{x}$. The concept of variational inference is first explained, and thereafter it is explained how this concept is implemented in a VAE.

3.3.1 Variational inference

The following section is adapted from Kingma & Welling (2019), which provides a clear and concise introduction to variational inference and Variational Auto-Encoders (VAEs).

The observed variable $\boldsymbol{x}$, is a random sample from the underlying process, with distribution $p^*(\boldsymbol{x})$. We do not know this true distribution, and try to approximate it with a model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $\boldsymbol{\theta} \in \mathbb{R}^{D_{\boldsymbol{\theta}}}$ is the parameters of the model.

The first goal in the variational inference process is to find the values of $\boldsymbol{\theta}$ for which it is most likely that the samples from the underlying process $p^*(\boldsymbol{x})$ came from the modelled distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. $\boldsymbol{\theta}$ must thus be set to values that will maximize the log-likelihood of the samples coming from $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. This is equivalent to the minimization of the Kullback Leiber (KL) divergence between the samples and $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. If we assume that the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is influenced by a set of latent variables $\boldsymbol{z}$ as

previously discussed, $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ can be viewed as a marginal distribution over the observed variables:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z} \tag{3-23}$$

This is also known as the model evidence or marginal likelihood. A model for $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ can be viewed as the decoder, taking latent representations $\boldsymbol{z}$ as input, and reconstructing $\boldsymbol{x}$ from it.

Calculating the marginal distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is typically intractable, because there doesn't exist an analytical solution to the integral in Eq. (3-23), and the integral is usually over a relatively high-dimensional latent space, making it difficult to estimate. Because of this, $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ can not be differentiated, and θ can not be optimized to maximize the log-likelihood.

Using Bayes' theorem, the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ can be written as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})} \tag{3-24}$$

$p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ is tractable to compute, but because $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is intractable, the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is also intractable. Because the posterior is intractable, a parametric inference model $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is introduced, to approximate the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. This model acts as the encoder, mapping the input data $\boldsymbol{x}$ to a lower dimensional latent representation $\boldsymbol{z}$. ϕ is known as the variational parameters. The second goal of this process is to optimize ϕ so that $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ resembles $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$.

The log-likelihood can now be written as:

$$\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})] && (3\text{-}25)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right]\right] && (3\text{-}26)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right]\right] && (3\text{-}27)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right]\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right]\right] && (3\text{-}28)
\end{aligned}$$

where the first term in Eq.(3-28) is known as the evidence lower bound (ELBO), and the second term is the Kullback-Leibler (KL) divergence between the posterior and the encoder. The KL-divergence is always non-negative:

$$KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) - \log p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right] \geq 0 \tag{3-29}$$

Re-arranging Eq.(3-28), and considering that the KL-divergence is always positive, gives

$$\begin{aligned}
ELBO &= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) && (3\text{-}30)\\
&\leq \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) && (3\text{-}31)
\end{aligned}$$

This shows that the ELBO is a lower bound of the log-likelihood. Thus, by maximizing the ELBO with respect to ϕ and θ, the log-likelihood of the data is also maximized. This means that $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ approximates the true underlying distribution $p^*(\boldsymbol{x})$, which was the first goal. Looking at Eq. (3-30), it can be seen that maximizing the ELBO also minimizes the KL-divergence, so $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ approximates $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ better, and thus one has also managed to approximate the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$, which was the second goal.

Since all the terms are tractable, we can maximize the ELBO :

$$ELBO = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] \tag{3-32}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] \tag{3-33}$$

By maximizing the ELBO, one will now have a probabilistic encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, and a probabilistic decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, which together form the Variational Auto-Encoder (VAE) (Kingma & Welling 2014).

3.3.2 VAE formulation

In a VAE, both the probabilistic encoder and probabilistic decoder are parameterized using neural networks, making it a nonlinear latent variable model. The idea is to train the network using backpropagation, finding the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ that maximize the ELBO. The loss function is the negative of the ELBO (given in Eq. (3-33)).

To be able to train the network using backpropagation, the loss function must be differentiable by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Differentiating with respect to $\boldsymbol{\theta}$ is simple, but differentiating with respect to $\boldsymbol{\phi}$ is more difficult, as the ELBO's expected value with respect to $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is taken, which is a function of $\boldsymbol{\phi}$. To solve this, a reparameterization trick is used, where $\boldsymbol{z}$ is reparameterized in terms of another random variable $\boldsymbol{\epsilon}$, so that the loss function can be written as an expectation of $p(\boldsymbol{\epsilon})$, making it easily differentiable by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. $\boldsymbol{z}$ is reparameterized using a function denoted here as $h(\cdot)$, as follows:

$$\boldsymbol{z} = h(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \boldsymbol{x}) \tag{3-34}$$

The distribution of $\boldsymbol{\epsilon}$ is independent of $\boldsymbol{\phi}$ and $\boldsymbol{x}$.

The loss function can now be written as

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{x}) = -\mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] \tag{3-35}$$

$$= -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z}) + \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \tag{3-36}$$

$$= -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{3-37}$$

where $\boldsymbol{z} = h(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \boldsymbol{x})$. Simply put, the reparameterization trick removes the sampling component that is initially inherent to the model and makes it an input. Because of this we then no longer have to differentiate with respect to it.

Note that the loss function attempts to minimize the KL-divergence between $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$. The chosen prior $p(\boldsymbol{z})$ regularizes the latent space around the carefully selected prior. The prior is usually chosen as a zero mean, isotropic unit variance Gaussian, as this forces the latent variables to cluster around zero. This can be seen as trying to force the latent variables to fall on a certain manifold.

The distribution form of the probabilistic encoder and probabilistic decoder must be specified, as the parameterizations of the chosen distribution will be learned. The forms of both the probabilistic encoder and probabilistic decoder are usually chosen to be multivariate Gaussian distributions, with diagonal co-variances. The rest of the discussion assumes these forms for the encoder, decoder and prior, as it is also the forms implemented in the investigations in this work.

The Gaussian encoder then takes the form:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}}, diag(\boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{x}})) \tag{3-38}$$

where $\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}}$ and $\log \boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{x}}$ are the outputs of the neural network, parameterized with $\boldsymbol{\phi}$. The *diag* operator constructs a diagonal matrix, where the input vector represents the diagonal elements.

The reparameterization function for $\boldsymbol{z}$ is given by

$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}} + \boldsymbol{\sigma}_{\boldsymbol{z}|\boldsymbol{x}} \odot \boldsymbol{\epsilon} \tag{3-39}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\odot$ is the element-wise product.

The Gaussian decoder takes the form:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}}, diag(\boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}})) \tag{3-40}$$

where $\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}}$ and $\log \boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}}$ are the outputs of the neural network, parameterized with $\boldsymbol{\theta}$. The logarithms of the variances $\boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{x}}$ and $\boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}}$ are learned, to prevent the model from learning negative variances.

The latent variable prior takes the form:

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I}) \tag{3-41}$$

Each of the three terms in the loss function in Eq. (3-36) can now be calculated, using the following equation for the log-likelihood of a multivariate Gaussian with diagonal covariance:

$$\log p(\boldsymbol{a}) = -\frac{1}{2}\sum_{i}^{D_{\boldsymbol{a}}} \left(\log(2\pi\sigma_i^2) + \frac{(a_i - \mu_i)^2}{\sigma_i^2} \right) \tag{3-42}$$

where $\boldsymbol{a}$ is the variable with dimensionality $D_{\boldsymbol{a}}$, $\boldsymbol{\mu}$ is the mean of the distribution, and $\boldsymbol{\sigma^2}$ the diagonal elements of the covariance matrix.

Note that in this implementation of the VAE, the reconstruction variance $\boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}}$ is learned. This is referred to here as a stochastic VAE. Some implementations of the VAE assume the variance $\boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}}$ to be an identity covariance matrix, referred to here as a deterministic VAE. For the deterministic VAE, the first term in the loss function in Eq. (3-37) reduces to the mean square error loss function. Recalling the discussion at the end of Section 3.2.3, the deterministic VAE will not be able to capture the heteroscedasticity of vibration data under varying operating speed conditions, since the variance is assumed to be constant over the whole dataset. The stochastic VAE on the other hand should be able to capture the variance accurately since the reconstruction variance is modelled with the decoder that takes $\boldsymbol{z}$ as input. The reconstruction variance is thus a function of $\boldsymbol{z}$. Since $\boldsymbol{z}$ will be a proxy of the operating conditions $\boldsymbol{c}$ present in the input data, the decoder should be able to learn the relationship between the operating speed and the reconstruction variance.

3.3.3 VAE implementation

The VAE model is used when no operating conditions are available (so in the unsupervised setting). The encoder and decoder architectures are documented in Appendix C. Fig. 26 shows a schematic representation of the VAE model.

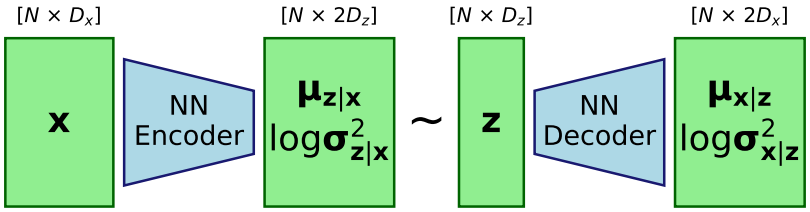


Figure 26: Schematic representation of VAE model

The reconstruction space is used to calculate discrepancy signals. The latent space is not monitored for discrepancies, since the model is used when no operating condition information is available.

3.4 Identifiable Variational Auto-Encoders

The Identifiable Variational Auto-Encoder (iVAE), proposed by Khemakhem et al. (2019), is a modification of the original VAE, with the goal of making the true joint distribution over the observed and latent variables $p(\boldsymbol{x}, \boldsymbol{z})$ identifiable. The idea of identifiability is that the learned latent variable prior and posterior distributions will then be representations of the true latent distributions that the observed data are generated from. So the learned latent representations can be found with a few simple transformations of the true latent distributions, helping with latent space disentanglement. The proposed model achieves this by conditioning the latent prior distribution on an additionally observed variable $\boldsymbol{u}$, so the prior changes from $p(\boldsymbol{z})$ to $p(\boldsymbol{z}|\boldsymbol{u})$. The prior is assumed to be conditionally factorial. $\boldsymbol{u}$ can be almost any other observation, such as a class label, or a time stamp in the case of time series. This approach to identifiability is similar to what recent advances in nonlinear Independent Component Analysis (ICA) follow (Hyvarinen et al. 2018).

The iVAE architecture provides a practical way of incorporating available operating condition information $\boldsymbol{c}$ into the modelling process of $\boldsymbol{x}$, by using $\boldsymbol{c}$ as the additionally observed variable $\boldsymbol{u}$.

3.4.1 iVAE formulation

This section will not provide the identifiability theory presented by Khemakhem et al. (2019), as it is not important to understand this for the application of condition monitoring. The rest of the section will only show how the encoder, decoder and prior distributions change from the original VAE and show the updated loss function.

Since the latent prior must be conditionally factorial, it was chosen to be multivariate Gaussian with a diagonal covariance, with the diagonal denoted by $\boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{u}}$. The latent prior takes the form:

$$p_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{u}}, diag(\boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{u}})) \tag{3-43}$$

and is parameterized by a neural network with weight parameters $\boldsymbol{\psi}$, which is trained along with the encoder and decoder. $\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{u}}$ and $\log \boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{u}}$ are the outputs of the neural network. The encoder and decoder are again chosen to be multivariate Gaussian distributions, with diagonal covariances. The Gaussian encoder then takes the form:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{u}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u}}, diag(\boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u}})) \tag{3-44}$$

where $\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u}}$ and $\log \boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u}}$ are the outputs of the neural network, parameterized with $\boldsymbol{\phi}$.

The reparameterization function for $\boldsymbol{z}$ is given by

$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u}} + \boldsymbol{\sigma}_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u}} \odot \boldsymbol{\epsilon} \tag{3-45}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and $\odot$ is the element-wise product.

The Gaussian decoder takes the same form as the VAE:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}}, diag(\boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}})) \tag{3-46}$$

where $\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}}$ and $\log \boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}}$ are the outputs of the neural network, parameterized with $\boldsymbol{\theta}$. The logarithms of the variances $\boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u}}$ and $\boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}}$ are learned, to prevent the model from learning negative variances.

The loss function can be written as

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{x}) = -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \log p_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{u}) + \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u}) \tag{3-47}$$

$$= -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u})||p_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{u})) \tag{3-48}$$

3.4.2 iVAE implementation

The iVAE model is used when the operating conditions are available (so in the semi-supervised setting). The available operating condition information $\boldsymbol{c}$ is used as the additional observed variable $\boldsymbol{u}$ that the latent prior is conditioned on. The iVAE is expected to perform better than the VAE, because the latent space prior is conditioned on the known operating conditions. As the latent representations are expected to mostly be a proxy for the known operating conditions, the prior conditioned on the operating conditions should act as a highly informed prior, or at least more informed than the zero mean, isotropic unit Gaussian prior used in the VAE. The neural network that parameterizes the conditional prior, is chosen as a simple feed-forward neural network, with two dense hidden layers of 32 neurons. The encoder and decoder architectures are the same ones used for the VAE model. The architectures are documented in Appendix C. Fig. 27 shows a schematic representation of the iVAE model.

The reconstruction space and latent space are used to calculate discrepancy signals. For the latent space discrepancy signal, the Mahalanobis distance between the mean of the latent posterior $\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{c}}$ and the learned prior $p(\boldsymbol{z}|\boldsymbol{c})$ is calculated using Eq. (2-8).

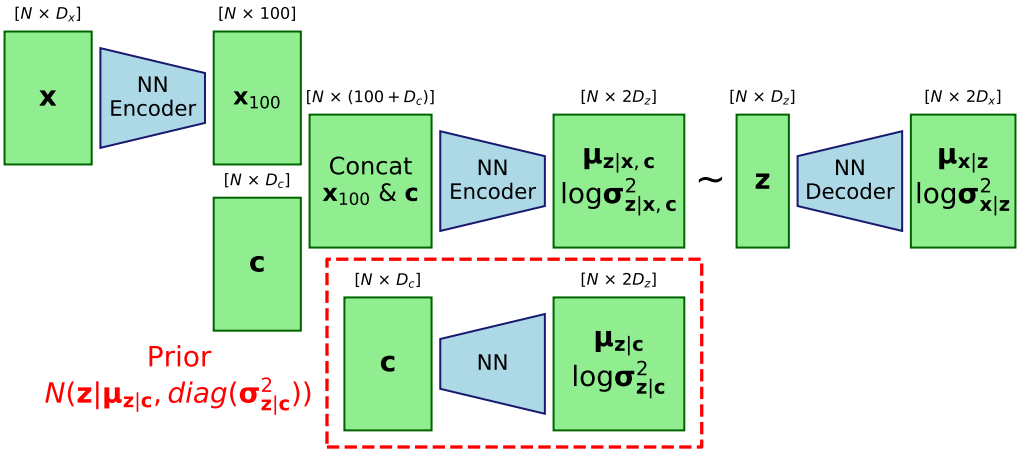


Figure 27: Schematic representation of iVAE model

3.5 VAE Training Challenges

The latent variable models described earlier can all map the high-dimensional input data to a lower-dimensional representation. It is important to understand that, with the nonlinear VAE models, there are an infinite number of possible encoder-decoder function combinations that can be learned, resulting in an infinite number of possible latent representations for the same input data. The weight parameters that are learned for the encoder and decoder, will be heavily influenced by how the weights are initialized. Not all lower-dimensional representations will however be equally good (Bengio et al. 2012). Two qualities to describe how good a latent representation is, are considered in this work:

- **Whether the model can accurately reconstruct the training data from the learned latent space**: If the latent variable model cannot learn to accurately reconstruct healthy training data, it will not be useful for damage detection, as poor reconstruction is used as an indicator of damage present in the signal. Poor reconstruction of the healthy data will lead to false alarms of healthy data being classified as unhealthy data.
- **Whether the latent space learned is disentangled or not**: For a disentangled representation, each of the generative factors can be isolated to certain latent variables in the representation.

This subsection explains how the VAE architecture can lead to entangled latent spaces and poor reconstructions, especially for this application of vibration signal discrepancy analysis. It is also explained how the iVAE architecture can be used to address these challenges when operating condition information is available.

3.5.1 Learning disentangled latent spaces

Let's first consider the idea of disentanglement. Bengio et al. (2012) argued that instead of just learning lower-dimensional representations of the data, one must try to learn representations that disentangle the factors of variation in the data. The advantage of this is that each generative factor can then be isolated to certain latent variables in the representation. There is not yet a widely accepted definition for disentanglement (Locatello et al. 2018), but the main idea is that a disentangled representation will allow one to change one of the latent variables in the representation, without any of the other latent variables changing. These latent space representations will then be more interpretable, as each latent variable will correspond with only one of the underlying sources of variation (generative factors) in the input data.

Wilke et al. (2022) explained the role of untangled latent representations in a condition monitoring context, stating that learning untangled latent spaces can help to isolate the effects of time-varying environmental and operating conditions on the condition monitoring signal. It might be possible to feed a raw vibration signal into the latent variable model, and then extract latent variables where one corresponds with the operating speed, another the load conditions, etc. The focus is thus on identifying sources in the data, and then the sources of interest can be isolated and analyzed, while the other sources are suppressed.

A few variations of the original VAE were proposed to encourage the VAE to learn disentangled representations, with the simplest variation considered here. The β-VAE (Higgins et al. 2017, Burgess et al. 2018) adds a simple extension to the original VAE, with the goal of learning disentangled latent spaces. The KL-divergence term in the original VAE loss function of the VAE (given in Eq. (3-37)) is scaled by a factor β:

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{x}) = -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \beta \times KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{3-49}$$

As was previously mentioned, the prior $p(\boldsymbol{z})$ is usually chosen as a zero mean, isotropic unit covariance Gaussian, so for the prior, the latent variables are independent and uncorrelated. By increasing β, the regularization of this prior $p(\boldsymbol{z})$ on the learned latent space is increased. This forces the VAE to learn a latent space closer to the proposed prior. With the added regularization, it can be expected that the reconstruction ability of the VAE will decrease. The KL-divergence between $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$ will thus decrease, at the cost of a larger reconstruction error. Burgess et al. (2018) proposed that β should initially be set to a small value, and then gradually be increased to the desired level so that the VAE can first learn to accurately reconstruct the input before the additional regularization is applied.

Locatello et al. (2018) challenged this idea to achieve disentangled representations, stating that without inductive biases on both the models and the data set, it is impossible to learn disentangled representations. Balshaw (2020) distinguished between implicit and explicit disentanglement of learned representations in the context of vibration-based condition monitoring. Implicit disentanglement is the form of disentanglement discussed so far, while for explicit disentanglement, the latent space components are explicitly prescribed to be a specific structure. Balshaw (2020) and Baggerohr (2019) used Generative Adversarial Network (GAN)-based models that aim to learn separable latent space components, where some capture the deterministic components of the vibration signals, and other the random components of the signals. This is particularly useful for bearing diagnostics, as it is known that bearing damage manifests in the random component of a signal. The idea is that if the damage only appears in the latent variables corresponding with the random components, the deterministic components of the signals can then be suppressed, and only the random components monitored for deviations from the healthy distribution. This approach is however tailored for bearing diagnostics. Damage will not necessarily be seen in the random component of the condition monitoring signal when other components are monitored.

The iVAE enables disentanglement by using the available operating conditions $\boldsymbol{c}$ as the additionally observed variable $\boldsymbol{u}$. One can manually change some of the operating condition values in $\boldsymbol{c}$, and then investigate how each operating condition (i.e. shaft speed) influences the latent representations and reconstructions.

For the proposed methodology presented in Chapter 2, it is not a requirement that disentangled latent spaces are learned. While it helps with the interpretability of the latent space, it is much more important for the proposed methodology that the model reconstructs the healthy samples well. Since there is usually a trade-off in the training objectives between disentangled latent spaces and the reconstruction quality of the input signal, the proposed methodology focuses more on addressing issues that can lead to poor reconstruction of the input signal.

3.5.2 Posterior collapse

A common cause of poor reconstructions in VAEs is posterior collapse (Wang et al. 2021). Posterior collapse is when the posterior of the latent variables $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ matches the uninformed latent prior $p(\boldsymbol{z})$. The encoder maps all of the input data to the same latent space representations. The latent representations then hold no useful information about $\boldsymbol{x}$, so the generative model (the decoder) learns to ignore $\boldsymbol{z}$ (Lucas et al. 2019, Wang et al. 2021). The decoder learns to reconstruct $\boldsymbol{x}$ as well as possible when no information about $\boldsymbol{x}$ is given, so would typically then just learn to return the mean and variance over the whole input dataset $\boldsymbol{X}$.

The exact causes of posterior collapse are still much debated. The most common approach to prevent posterior collapse is to use a β-VAE and slowly increase β from 0 to 1 during training. The regular-

ization effect on the latent space, caused by the KL term in the loss function, is thus reduced. Lucas et al. (2019) criticizes this idea of scaling the KL term to prevent posterior collapse, arguing that posterior collapse is caused by spurious local maxima in the training objective and that scaling the KL term only works when it reduces the stability of these local maxima. Wang et al. (2021) showed that posterior collapse occurs when latent variables are non-identifiable (when the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ does not depend on the latent variable $\boldsymbol{z}$). It is argued that the β-VAE prevents posterior collapse by preventing the model from learning non-identifiable latent variables.

The main things to note from this discussion are that posterior collapse will lead to poor reconstructions and that it can be prevented by learning identifiable latent variables, or by decreasing the effect of the KL term in the VAE loss function. Conditioning on the additionally observed variable $\boldsymbol{u}$ in the iVAE can help to learn identifiable latent variables, while the β-VAE can be used to scale the KL term in the VAE loss function.

3.5.3 Periodic latent variables

The vibration data samples recorded from rotating machinery are expected to be approximately periodic for each shaft rotation. During the investigations performed in this work, it was found that when learning periodic signals with VAEs, the challenges of learning disentangled latent spaces and dealing with posterior collapse arise during training. The majority of the literature dealing with disentanglement and posterior collapse investigates these phenomena using image recognition applications. Periodic signals are however less complex than images, which allows one to visualize these concepts more easily. In this subsection, it is discussed why it is difficult to learn disentangled latent spaces when dealing with periodic signals, and how trying to disentangle the latent space, can lead to posterior collapse.

Let's consider a dataset consisting of perfect sine wave vibration signals, with an amplitude of 1 and frequency of 1 Hz, sampled at a high sampling rate, with some measurement noise. A sample is shown in Fig. 28. We are interested in finding out how the VAE will capture the data in the latent space if we randomly took 1-second windows from these signals (shown by the green highlighted background in Fig. 28), and presented it to a VAE.

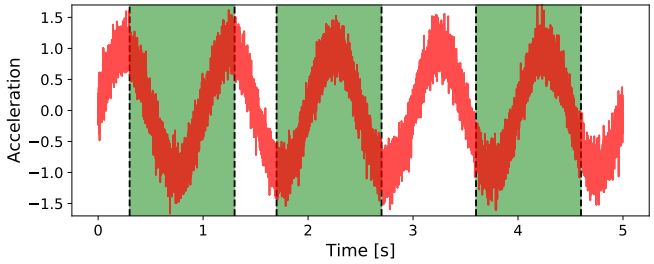


Figure 28: 1-second windows of a noisy sine wave.

Each of the windows can be accurately described by passing a time vector $\boldsymbol{t}$, containing values from 0 to 1 second, through the function:

$$f(p, t_i) = \sin(2\pi \times t_i + p) \tag{3-50}$$

where p represents the phase of the signal segment, and t_i is an element of the time vector $\boldsymbol{t}$. To accurately reconstruct the signals, the VAE model must only learn to capture a representation of p in the latent space. p is a scalar value, so one would expect that all the latent representations will fall on

a 1-dimensional latent manifold. It is however important to note that since the function is periodic, $p = 0$ and $p = 2\pi$ will result in the exact same input data, so input data for $p = 0$ and $p = 2\pi$ will be mapped to the same region in the latent space. All values of $p + k \times 2\pi$ will be mapped to the same space on the latent manifold, since the encoder will not be able to distinguish between samples with phases p and $p + 2\pi$. The VAE encoder typically encodes this information by mapping the input data to a lower dimensional circular manifold, indirectly learning a representation containing both $\sin(p)$ and $\cos(p)$, with the representations for $\sin(p)$ and $\cos(p)$ being highly correlated. The VAE will typically require at least two correlated latent variables to capture these representations of $\sin(p)$ and $\cos(p)$. While it is possible to extract one uncorrelated latent variable representing p using inverse trigonometric functions such as $\sin^{-1}$ or $\tan^{-1}$, a VAE will typically not learn to extract one uncorrelated variable, but rather just capture the information on a circular manifold with correlated latent variables. In Appendix A, it is shown how different factors of variation in periodic signals (such as varying amplitudes and frequencies) are projected to the latent space, to give the reader a better intuition behind how latent variable models capture the generative factors of the data in the latent space.

Other examples of data that can be represented in terms of these circular manifolds (or periodic latent variables) are data which captures some form of direction, such as photos of an object rotated along an axis. If the object is arbitrarily rotated, all the possible views of the object can be projected to a spherical manifold (Perez Rey et al. 2020).

When a VAE learns to project the input data to a circular manifold where the latent variables are entangled, a problem arises when trying to disentangle the learned representation. The VAE prior, when chosen as a zero mean, isotropic unit covariance Gaussian, will try to disentangle the latent variables that capture this circle, which will have to break this circular structure in the learned representation to make the variables uncorrelated. This is referred to as a 'manifold mismatch' (Davidson et al. 2018).

To investigate what the learned manifold and reconstructions will look like, three models were trained on the noisy sine wave data:

- A normal VAE ($\beta = 1$).
- A β-VAE with β set to 0.1, so that the regularization effect of the prior is reduced.
- An iVAE, where a prior conditioned on $\boldsymbol{u}$ is learned. $\boldsymbol{u}$ captures the phase information, with $\boldsymbol{u} = [\sin(p), \cos(p)]$.

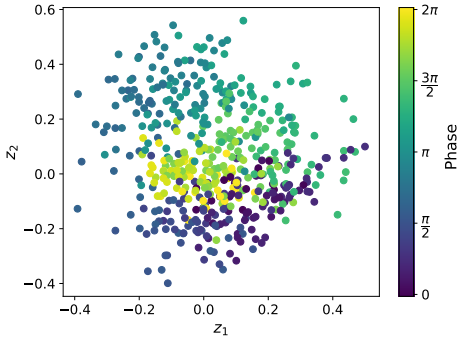
All three models are given a latent space size of 2, since this is large enough to capture a circular manifold, and allows for easy visualization. The latent space representations for all three models are plotted in Fig. 29, along with the reconstruction of one of the signal segments.

For the normal VAE, the prior is enforced too strong, with all the latent representations from the posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{u})$ matching the uninformed latent prior $p(\boldsymbol{z})$, as seen in Fig. 29(a). The KL-term regularization led to posterior collapse. This can also be seen in the reconstruction of the input data (Fig. 29(b)), with the model only returning the mean and variance of the whole dataset $\boldsymbol{X}$, with no phase information captured.

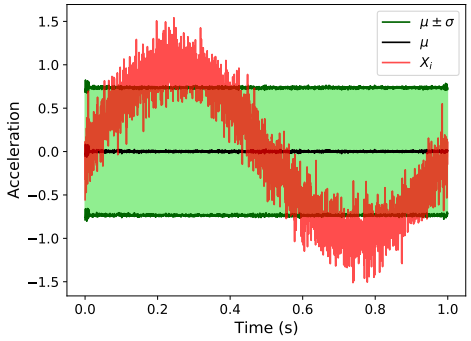
When the KL regularization is decreased by setting $\beta = 0.1$, the VAE is able to capture a circular manifold (Fig. 29(c)), and accurately reconstruct the data (Fig. 29(d)).

The iVAE also captures a circular manifold (Fig. 29(e)), and accurately reconstructs the data (Fig. 29(f)), even though the KL term in the loss function is not scaled to decrease the effect of the

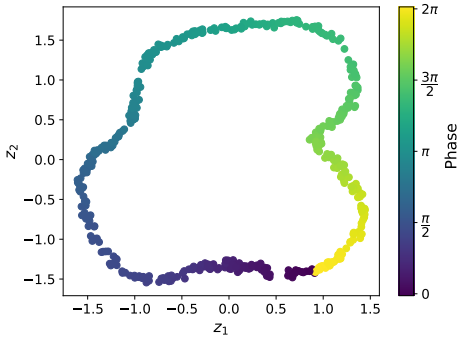
latent space regularization. Because the iVAE has a conditional prior $p_{\psi}(\boldsymbol{z}|\boldsymbol{u})$ that is learned, the regularization by the KL term in the loss function is only applied to localized parts on the latent manifold. By conditioning on $\boldsymbol{u}$, which contains the phase information, the enforced prior is pointed to the right local part on the manifold that the phase in the signal segment corresponds to. The differences between the VAE and iVAE priors are visualized in Fig. 30. While the VAE prior pulls all the latent representations to zero, the iVAE prior, when marginalizing $\boldsymbol{u}$ out, will cover a circular space in the latent space.



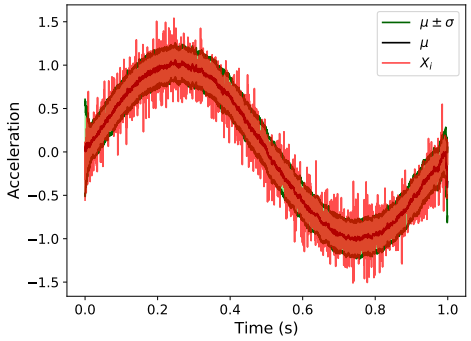
(a) Latent representations: VAE ($\beta = 1$)



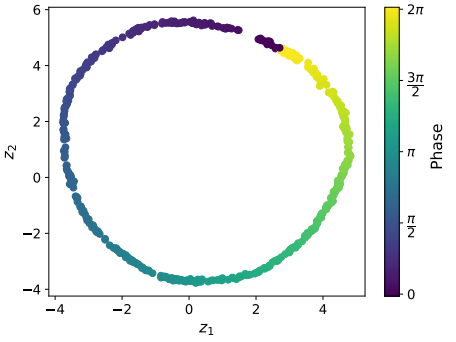
(b) Reconstruction: VAE ($\beta = 1$)



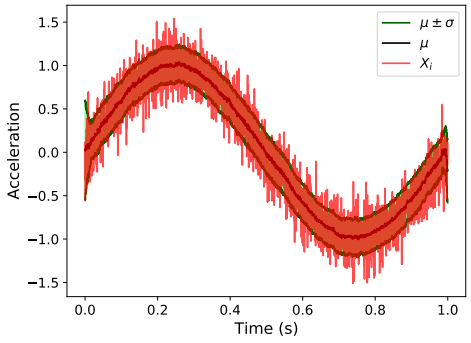
(c) Latent representations: VAE ($\beta = 0.1$)



(d) Reconstruction: VAE ($\beta = 0.1$)

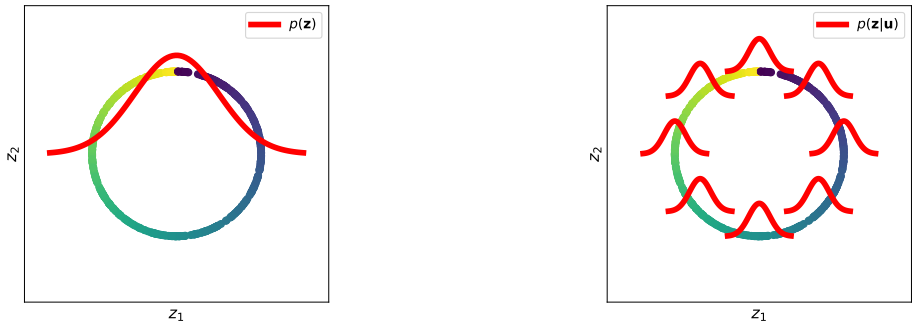


(e) Latent representations: iVAE ($\beta = 1$)



(f) Reconstruction: iVAE ($\beta = 1$)

Figure 29: Latent space representations and reconstruction of noisy sine wave segments with varying phase, modelled with VAE ($\beta = 1$), VAE ($\beta = 0.1$) and iVAE ($\beta = 1$).



(a) VAE prior

(b) iVAE prior

Figure 30: Comparison of how the VAE and iVAE prior acts as regularization on the latent space.

A Gaussian prior for the VAE has limitations when applied to a low dimensional latent space, as it pulls the latent representations to the origin. When the latent space is high-dimensional, the Gaussian prior has a different effect, referred to as the 'soap bubble' effect (Davidson et al. 2018). A standard Gaussian prior in high dimensions resembles a uniform distribution on a hypersphere. The prior thus forces the representations to lie on the surface of this hypersphere/'soap bubble' in the high-dimensional space. While the low-dimensional Gaussian prior will cause the learned latent circular manifold to collapse, the high-dimensional Gaussian prior might be able to capture this manifold. This will however mean that one would have to use a high-dimensional latent space to capture a simple one-dimensional circular manifold. Some approach this problem by explicitly changing the shape of the latent space from the normally used hyperplanar latent space to a hyperspherical latent space (Davidson et al. 2018, Perez Rey et al. 2020). These VAE variations are not implemented in this work, as not all the generative factors present in periodic signals and vibration data are periodic (see Appendix A).

This work shows the effect that Gaussian priors have on the learned representations for condition monitoring signals when using normal VAEs with low- and high-dimensional latent spaces. Because of the more informative prior in the iVAE model, it is expected to perform better than the VAE.

3.6 Choosing the size of the latent space

It is expected that the size of the latent space will play a big role in whether the damage is detected in the latent or reconstruction space, since a larger latent space should act as less of an information bottleneck, making it more likely that the model simply learns an identity function transformation from some of the input variables to the reconstruction space (as was discussed in Section 1.4.3). For this reason, it was decided to investigate each model's performance when small and large latent spaces are used.

For the PCA models, the large latent space size was chosen as the number of principal components that could capture 95% of the variance in the training data.

To choose a smaller latent space size for the PCA models, the goal was to choose the number of principal components $D_{\boldsymbol{z}}$, where adding more principal components will mostly just memorize the noise present, and not capture any more of the deterministic part of the signals. The process can be summarized as follows:

- Starting at $D_{\boldsymbol{z}} = 1$, the validation set signal segments $\boldsymbol{X}$ are fed through the model to find the reconstructions $\boldsymbol{X}_{recon}$. The element-wise reconstruction error mean and variance are

calculated to find a distribution for the reconstruction error. A fake validation set is then created by adding noise sampled from the reconstruction error distribution to $\boldsymbol{X}_{recon}$.

- A principal component is added, the true and fake validation sets are passed through the model, and the MSEs for the true and fake validation sets are compared. The process is repeated until the MSE for the fake set is equal to or below the MSE for the true validation set. This is determined as the point where adding more principal components will just capture more noise in the latent space.

For the VAE and iVAE models, it is more difficult to determine how much information the bottleneck will let through, so to investigate the effect of different latent space sizes, models with latent space sizes of 10 and 100 were investigated. This was chosen somewhat arbitrarily, with the only reasoning being that, in most cases, the model will only have to capture the shaft speed, and the phase information, so 10 latent variables should be enough to capture the operating conditions for a nonlinear model.

The models are labelled based on their latent space sizes, by adding an 'S' to models with a small latent space, and an 'L' to models with a large latent space. So a model with the name iVAE-L has $D_{\boldsymbol{z}} = 100$. The latent space sizes used in both investigations in Chapter 4 and Chapter 5 are documented in Appendix C.

3.7 Implementation Summary

As a conclusion to the chapter, a summary is given to show how the models and different pre-processing approaches fit into the unsupervised and semi-supervised frameworks. The different models and pre-processing approaches are grouped together in Table 3, and classified based on where the OC information is used during the process. All of these model and pre-processing combinations are implemented in the two analyses presented in Chapters 4 and 5.

Table 3: Model and pre-processing classification based on level of supervision

<table>
<tr><th rowspan="2">Model</th><th rowspan="2">Discrepancy space</th><th colspan="6">Pre-processing</th></tr>
<tr><th>SO
1-rev</th><th>SO
1-tooth</th><th>OT
1-rev</th><th>OT
1-tooth</th><th>OTA
1-rev</th><th>OTA
1-tooth</th></tr>
<tr><td>PPCA-S</td><td>Recon</td><td colspan="2" rowspan="6">Unsupervised approach

Monitored features:
mean, var, kurt</td><td colspan="4" rowspan="6">OC info used for pre-processing

Monitored features:
mean, var, kurt, OST, SAT</td></tr>
<tr><td>PPCA-L</td><td>Recon</td></tr>
<tr><td>PCA-Z-LR-S</td><td>Recon</td></tr>
<tr><td>PCA-Z-LR-L</td><td>Recon</td></tr>
<tr><td>VAE-S</td><td>Recon</td></tr>
<tr><td>VAE-L</td><td>Recon</td></tr>
<tr><td>C-decoder</td><td>Recon</td><td colspan="2" rowspan="5">OC info used for modelling

Monitored features:
mean, var, kurt, OST, SAT</td><td colspan="4" rowspan="5">OC info used for pre-processing and modelling

Monitored features:
mean, var, kurt, OST, SAT</td></tr>
<tr><td>PCA-C-LR-S</td><td>Recon + Latent</td></tr>
<tr><td>PCA-C-LR-L</td><td>Recon + Latent</td></tr>
<tr><td>iVAE-S</td><td>Recon + Latent</td></tr>
<tr><td>iVAE-L</td><td>Recon + Latent</td></tr>
</table>

4 Phenomenological Model Dataset Analysis

A phenomenological model was used to create a dataset that resembles measured vibration signals from a gearbox under time-varying operating conditions, with and without bearing faults. The advantage of using a phenomenological model dataset instead of an experimental dataset is that one can explicitly specify the noise level and damage level present in the data. This is ideal to investigate the performance of models on fault detection and tracking tasks, as the size of the damage component in the measured data, is known at all times. The noise levels for this dataset were also chosen to be relatively low, so that the models' abilities to capture large noise distributions are not the main focus in this investigation, but rather how well the models capture the generative factors such as shaft speed and phase information in the latent space. With a phenomenological dataset, all the components in the data can be easily explained, removing any uncertainty on the data side. This chapter is therefore used to get a benchmark idea of how the proposed methodology performs on fault detection and tracking tasks, and to get a better understanding of why some models perform better than others. The investigation in Chapter 5 focuses on real-world data.

This investigation builds on the following work, where datasets generated using the same phenomenological model were also analysed:

- Balshaw (2020) focused on how movements in the latent space between consecutive vibration signal segments, can be used as a health indicator. For the section using the phenomenological model, constant operating conditions were investigated.
- Schmidt et al. (2019*a*) investigated a discrepancy analysis approach for varying operating conditions. Features were manually extracted from the healthy data, and modelled with a Gaussian model. To remove the correlation between the discrepancy measures and the shaft speed, a Gaussian distribution was fit to capture the density $p(\eta|\omega)$, where η is the discrepancy measure and ω is the rotational speed. The values of η were then normalized using the parameters of the Gaussian distribution at each value of ω. This approach incorporates the available OC information during the post-processing of the discrepancy signals.

4.1 Dataset Overview

The phenomenological model used was proposed by Abboud et al. (2017). The model aims to reproduce measured vibration signals for a gearbox with gear and bearing faults, under time-varying operating conditions. The signals produced are a function of the rotational speed, with the effects of amplitude modulation included in the model.

The vibration signal consists of four components added together:

$$x(t) = x_{gd}(t) + x_{gr} + x_n(t) + x_b(t) \tag{4-1}$$

where $x_{gd}(t)$ is the deterministic gear component (caused by the gear teeth meshing), x_{gr} is the random gear component (caused by distributed gear damage), x_n is a noise component, and $x_b(t)$ is the bearing component (caused by either an inner race fault, given as $x_{bi}(t)$, or an outer race fault, given as $x_{bo}(t)$).

A vibration sensor will not measure the exact vibration present at the source of the vibration, as the vibration first propagates through the structure to the position where the sensor is mounted. This transmission path through the structure causes the vibration sensor to measure a modulated version

of the true excitation signal. The model accounts for this modulation, with the components (except for the noise component) given as:

$$x_{gd}(t) = h_{gd}(t) \circledast z_{gd}(t) \tag{4-2}$$
$$x_{gr}(t) = h_{gr}(t) \circledast z_{gr}(t) \tag{4-3}$$
$$x_b(t) = h_b(t) \circledast z_b(t) \tag{4-4}$$

where $h(t)$ is an impulse response function, $\circledast$ is the convolution operator, and $z(t)$ is the true excitation signal. Schmidt et al. (2019*a*) modelled the impulse response function as a single degree-of-freedom system with viscous damping:

$$h_i(t) = e^{-\zeta_i \omega_{n,i} t} \sin\left(\sqrt{1-\zeta_i^2}\omega_{n,i} t\right) \tag{4-5}$$

where ζ_i is the associated damping ratio of the component, and $\omega_{n,i}$ the natural frequency of the component in rad/s.

The true excitation signal for the deterministic gear component is given by Schmidt et al. (2019*a*) as:

$$z_{gd}(t) = M_{gd}(\omega_{shaft}(t)) \sum_{k=1}^{N_{gd}} A_{gd}^{(k)} \cos\left(k \cdot N_{t,g} \int_0^t \omega_{shaft}(\tau) d\tau + \phi_{gd}^{(k)}\right) \tag{4-6}$$

where $M_{gd}(\omega_{shaft}(t))$ is a function that simulates the effect of amplitude modulation, N_{gd} is the number of gear mesh components considered, $A_{gd}^{(k)}$ is the amplitude and $\phi_{gd}^{(k)}$ the phase of component k, $\omega_{shaft}(t)$ is the rotational speed of the shaft, and $N_{t,g}$ is the number of gear teeth.

The true excitation signal for the random gear component is given by Schmidt et al. (2019*a*) as:

$$z_{gr}(t) = M_{gr}(\omega_{shaft}(t))\epsilon_{gr}(t) \sum_{k=1}^{N_{gr}} A_{gr}^{(k)} \cos\left(k \int_0^t \omega_{shaft}(\tau) d\tau + \phi_{gr}^{(k)}\right) \tag{4-7}$$

with similar components to the deterministic gear signal, but with a random Gaussian component $\epsilon_{gr}(t)$, given as:

$$\epsilon_{gr}(t) \sim \mathcal{N}(0, \sigma_{gr}^2) \tag{4-8}$$

The distributed gear damage that this random gear component simulates, makes the bearing condition inference process more difficult (Schmidt et al. 2019*a*).

The true excitation signal for the bearing component in the case of an outer race fault is given by Schmidt et al. (2019*a*) as:

$$z_{bo}(t) = M_b(t) \sum_{i}^{N_{\mathcal{T}}} F_{dam_o}^{(i)} \delta(t - \mathcal{T}_i) \tag{4-9}$$

which is a series of Dirac functions centred at $\mathcal{T}_i$. $\mathcal{T}_i$ represents the time when a roller element strikes the fault on the race, and is a function of the characteristics of the bearing, as well as the rotational speed. Slip is introduced by adding zero-mean Gaussian noise with a standard deviation of 0.1 to $\mathcal{T}_i$, to shift the time of impact slightly forwards or backwards in time. F_{dam_o} is sampled from a Gaussian distribution with mean 1 and standard deviation $\sigma_{F_{dam_o}}$. This simulates the effect that each bearing

impulse does not have a constant magnitude.

The true excitation signal for the bearing component in the case of an inner race fault is given by Schmidt et al. (2019*b*) as:

$$z_{bi}(t) = q_{stribeck}\left(\int_0^t \omega_{shaft}(\tau)d\tau\right) M_b(t) \sum_i^{N_\mathcal{T}} F_{dam_i}^{(i)} \delta(t - \mathcal{T}_i) \tag{4-10}$$

which has the same components as the outer race fault, but with the signal being multiplied by $q_{stribeck}\left(\int_0^t \omega_{shaft}(\tau)d\tau\right)$, where $q_{stribeck}(\theta)$ is the Stribeck equation:

$$q_{stribeck}(\theta) = \begin{cases} q_0(1 - \frac{1}{2\epsilon}(1 - \cos(\theta)))^{c_{str}}, & \text{for } |\text{wrp}(\theta)| \leq \theta_{max} \\ 0, & \text{otherwise} \end{cases} \tag{4-11}$$

The wrap function $\text{wrp}(\theta)$ returns θ in the range $[-\pi, \pi]$. c_{str}, ϵ, q_0 and θ_{max} were set to the same values used by Schmidt et al. (2019*b*) ($c_{str} = 3/2$, $\epsilon = 0.49$, $\theta_{max} = 0.99\pi/2$ and $q_0 = 1$). The Stribeck equation simulates how the damaged area on the inner race moves in and out of the loading zone.

The noise component is given as:

$$x_n(t) = \epsilon_n(t) M_n(\omega_{shaft}(t)) \tag{4-12}$$

where $\epsilon_n(t)$ is sampled from a zero-mean Gaussian with standard deviation σ_n, and $M_n(\omega_{shaft}(t))$ models the amplitude modulation effect.

All of the components in Eq. (4-1) are now defined. The last part of the model is to scale the components based on signal contribution levels at constant operating conditions. The components are scaled relative to the noise component, by specifying the signal-to-noise ratio (SNR) of each component. The SNR in dB is given as:

$$SNR_{dB} = 10\log_{10}\left(\frac{A_{signal}}{A_{noise}}\right)^2 \tag{4-13}$$

where A is the RMS of the signal. By specifying the desired SNR, the necessary scaling component for each term, C can be calculated:

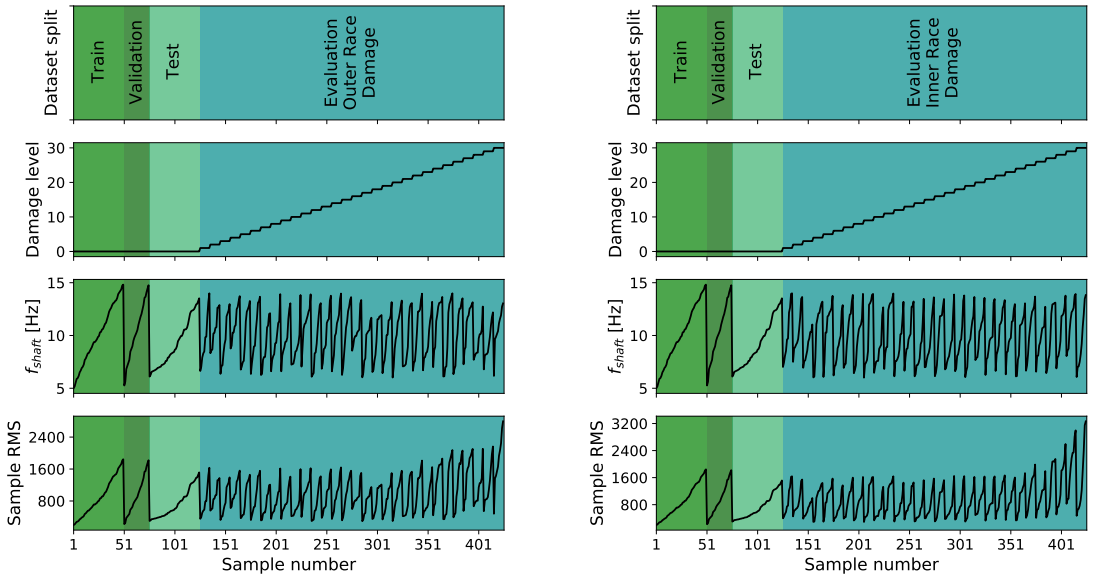
$$C = \frac{10^{\frac{SNR + 20\log_{10}(A_{noise})}{20}}}{A_{signal}} \tag{4-14}$$

This scaling coefficient C can then also be used to specify the damage level present for the bearing faults, as a larger fault will give a higher SNR.

The investigation was done for varying operating conditions. The shaft speed is kept constant per sample but varied from one sample to the next. This is done so that any correlation between discrepancy signal features and operating speed can be easily detected. The shaft speed for the training and validation sets were sampled from a uniform distribution ranging from 5 to 15 Hz, and the test set and evaluation set shaft speeds were sampled from a uniform distribution ranging from 6 to 14 Hz. This was done to ensure that there are no outliers in the evaluation set that are not represented in the training data. Outliers will affect the results, and the models' abilities to extrapolate outside

the training data range are not investigated in this work. 30 damage levels for the bearing faults were simulated. The SNR of the damage components corresponding to the 30 damage levels, was increased linearly from -30dB to 10 dB. To calculate the scaling parameters of the bearing damage components, the scaling coefficients necessary to achieve the specified SNR for each damage level for a shaft speed of 10 Hz were calculated, and these scaling coefficients were then applied to all samples. Each sample is recorded for 1 second, with a sampling frequency of 25000 Hz. 125 healthy samples were generated (50 for the training set, 25 for the validation set, and 50 for the test set), along with 10 samples for each damage level, for inner and outer race faults respectively. This gives 425 samples per dataset (125 healthy, 300 damaged), considering the inner and outer race faults as separate datasets. Note that the same healthy data is used for both the outer race and inner race fault datasets, so models were only trained once, and then used to evaluate both datasets.

Fig. 31 summarizes the damage level, f_{shaft} and RMS for each sample in the two datasets. The samples that are from the same damage level and same dataset split, were sorted based on operating speed, to simplify visualization. Note that the RMS is highly correlated with the shaft speed. Only a slight increase in the RMS caused by damage can be detected over the last few damage levels in both datasets.



(a) Outer race fault dataset

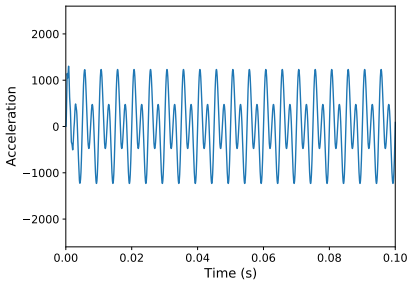
(b) Inner race fault dataset

Figure 31: Summary of damage level, f_{shaft} and RMS for each sample in both phenomenological model datasets.

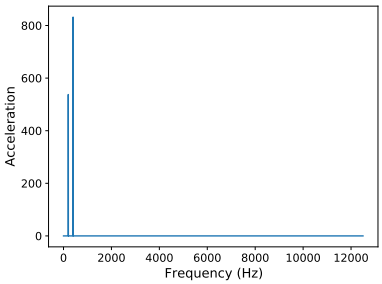
Similar model parameters to those used by Balshaw (2020) are used in this study. The parameters of the phenomenological model are listed in Appendix D. The following model parameters are important for the interpretation of the results:

- The gear mesh frequency is 20 Hz/f_{shaft}, so the deterministic gear component consists of the frequency 20 Hz/f_{shaft} and its harmonics.
- The Ball Pass Frequency Outer race (BPFO) is 4.12 Hz/f_{shaft}, and the Ball Pass Frequency Inner race (BPFI) is 5.88 Hz/f_{shaft}. The bearing inner and outer race natural frequencies are 7000 Hz.

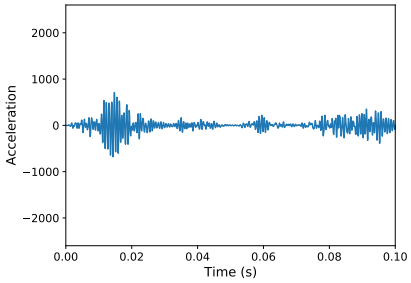
Fig. 32 shows the four signal components of Eq. (4-1) in the time domain (for one shaft rotation) and frequency domain, at a shaft speed of 10 Hz, and a bearing outer race damage level of 15.



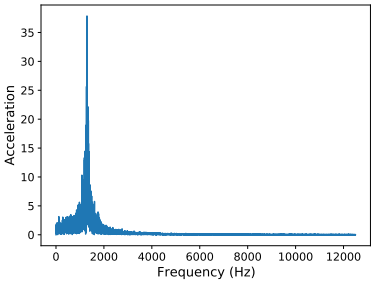
(a) x_{gd}: Time domain



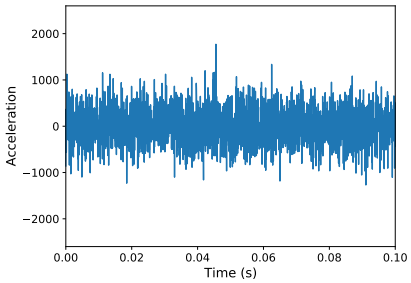
(b) x_{gd}: Frequency domain



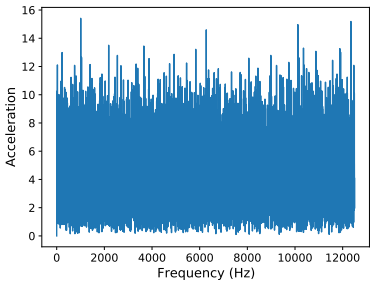
(c) x_{gr}: Time domain



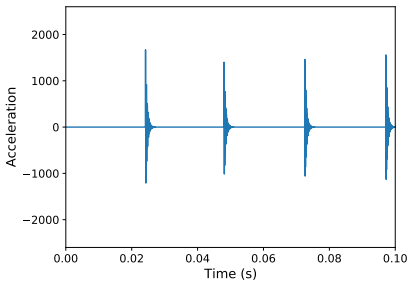
(d) x_{gr}: Frequency domain



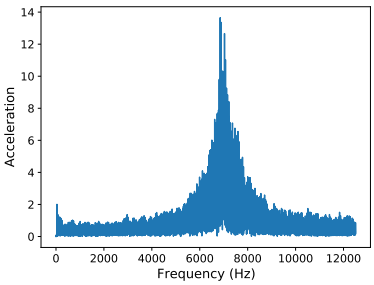
(e) x_n: Time domain



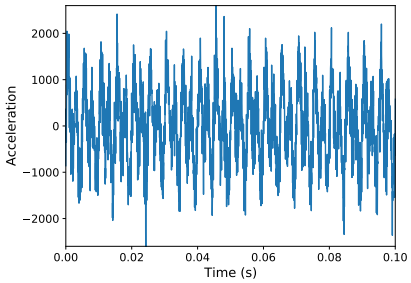
(f) x_n: Frequency domain



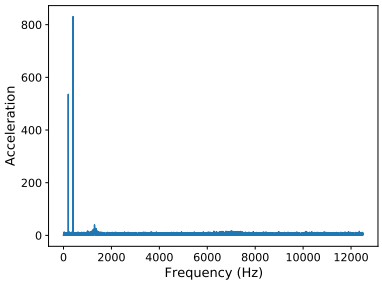
(g) x_{bo}: Time domain



(h) x_{bo}: Frequency domain



(i) $x_{measured}$: Time domain



(j) $x_{measured}$: Frequency domain

Figure 32: Phenomenological model signal components of Eq. (4-1) in the time and frequency domain at a shaft speed of 10 Hz, and a bearing outer race damage level of 15.

4.2 Experimental Setup

All 11 models, with the 6 different pre-processing approaches, as shown in Table 3 were implemented. Since the dataset is fairly small, the L_{sft} parameter was chosen as $L_w/4$. This allows for a somewhat finer resolution to the latent space discrepancy signals, without increasing the computational load too much. The pre-processing parameters and latent space sizes are documented in Appendix C.

The features of the discrepancy signals that are monitored in this investigation are the mean, variance, kurtosis, and OST at order 4.12 Hz/f_{shaft} for outer race faults, and order 5.88 Hz/f_{shaft} for inner race faults. SAT is not implemented, as bearing faults are not correlated with the shaft angular position. For the unsupervised approach, the OC information will not be available that allows one to perform order tracking and calculate the OST feature from the discrepancy signals. It is however still calculated and documented for the unsupervised approach, as it gives a good indication of a model's ability to capture the distribution of the healthy data.

Three metrics were used to evaluate how well models perform on fault detection and tracking tasks:

1. Damage detection level (DDL): This is the damage level (level 1 to level 30) from which all samples further on with larger damage levels fall above the discrepancy threshold, where the discrepancy threshold is set as the 95th percentile of the discrepancy signal features for the healthy test data. From this level on, no false negative classifications are seen. This metric is used to see how early a model can detect damage for the full range of operating conditions present in the data. Other studies usually set the threshold as the mean plus six times the standard deviation (Abboud et al. 2019, Balshaw et al. 2022), but it was chosen to use the 95th percentile instead, since the discrepancy signal features might not be normally distributed.

2. True positive rate (TPR): The percentage of the damaged samples (samples 126 to 425) that fall above the discrepancy threshold, being classified as faulty. By setting the discrepancy threshold as the 95th percentile in the healthy test data, the false positive rate (FPR) is expected to be around 0.05.

3. Area Under The Curve (AUC) of Receiver Operating Characteristics (ROC): The AUC was also added, since it is a metric commonly used in binary classification tasks, that is insensitive to the choice of the threshold. It is however expected that the DDL and TPR will give a better indicator of model performance in the context of condition monitoring, since one is not interested in knowing how the model performs at FPRs much higher than 0.05 (which the AUC also captures), as this would cause too many false alarms.

The monotonical increase of the discrepancy signal features as the fault develops is not directly monitored in this work. The metrics will not be able to detect if the discrepancy signal features stabilize at a certain value above the discrepancy threshold. The metrics will however detect if the discrepancy signal features fall back below the discrepancy threshold.

A signal processing benchmark is also set by analysing the data using traditional envelope analysis methods, to see how the learning-based approaches compare to commonly used envelope analysis approaches. The squared envelope spectrum (SES) and normalized squared-magnitude of the SES (NES) are tracked at the critical order of 4.12, for an unfiltered dataset as well as a bandpass filtered dataset. For the bandpass filtered dataset, the dataset is filtered at 6000-8000 Hz, since it is known from the phenomenological model that the bearing natural frequency is at 7000 Hz. Looking at the signal components in the frequency domain in Fig. 32, it can be seen that filtering the data around this frequency band effectively removes all components other than the bearing component, and part

of the noise component.

The SES has been used in a popular benchmark study done by Smith & Randall (2015), and the NES are used in state-of-the-art approaches (Abboud et al. 2019).

The SES is given by:

$$SES_{\boldsymbol{x}}(\alpha) = |DTFT_{n\to\alpha}\{|A\{\boldsymbol{x}[n]\}|^2\}| \tag{4-15}$$

where α represents the frequency variable of unit Hertz, $A\{\cdot\}$ the Hilbert transform, and $DTFT\{\cdot\}$ the discrete time Fourier transform (Abboud et al. 2019).

The NES is obtained by normalizing the SES with the DC offset of the spectrum:

$$NES_{\boldsymbol{x}}(\alpha) = \left(\frac{SES_{\boldsymbol{x}}(\alpha)}{SES_{\boldsymbol{x}}(0)}\right)^2 \tag{4-16}$$

4.3 Why do some latent variable models work better than others?

Before comparing the different models and pre-processing approaches with the metrics used to evaluate the performance on fault detection and tracking tasks, it is important to develop an intuition of why some of the model and pre-processing combinations will work better than others. The goal of this subsection is to show what factors influence the performance metrics so that when the performance metrics are presented in Section 4.4, the results are interpretable. The reader should have a good idea of which models are expected to perform better than others after reading this subsection.

It was seen that the main factor that influences the performance metrics, is how well the models fit the distribution of the healthy data. If the model doesn't fit some of the signal components associated with a healthy signal well, the discrepancy signals will be much noisier, which raises the discrepancy threshold, resulting in damage being detected at a later stage. Some models' performances are also heavily affected when the model overfits. Overfitting in this context refers to when feeding data through the encoder and decoder of the model starts to resemble an identity function transformation, where anything being fed into the model gets reconstructed accurately. The damage will then only be visible in the latent space discrepancy signals, not in the reconstruction space discrepancy signals, as was discussed in Section 1.4.3. Models that do not allow for latent space discrepancy monitoring, will not be able to detect damage in this case.

To investigate how the models fit healthy data, likelihood comparisons on the healthy test set can be made, as this gives an indication of how well the models reconstruct unseen healthy data. The likelihood comparisons on the healthy test data are presented first, and thereafter the different models' abilities to fit the healthy data well are compared based on two categories:

- Homoscedastic models (models that assume a constant variance over the dataset) vs. heteroscedastic models (models that capture the variance per sample).
- Linear PCA models vs. nonlinear VAE models. It is investigated why order tracking makes it easier for the models to fit the data. The effects of different latent space sizes and different input signal segment lengths on the linear and nonlinear models are also investigated.

This subsection ends with a short discussion on why some of the monitored discrepancy signal features will give better discrepancy analysis results than others.

Only results from the outer race fault dataset are presented in this subsection, as similar trends were seen for both inner and outer race fault monitoring. The full result tables for the outer and inner race faults are given in Appendix E.

4.3.1 Likelihood comparison of healthy test data

To analyse how well the models fit the healthy test data, we look at the reconstruction likelihood values. It is however a difficult task to infer the quality of the fit directly from the likelihood values, as the likelihood values are strongly correlated with the associated operating speeds. The data is expected to be heteroscedastic, with larger variances at faster operating speeds, resulting in lower likelihood values at faster operating speeds. Because a phenomenological model was used to create this dataset, the exact relationship between the variance terms and the shaft speed is known, allowing one to scale the likelihood values to remove the correlation with the shaft speed. This relationship between the variance in the data and the operating speeds is determined by the function chosen for the amplitude modulation functions (M_{gd}, M_{gr}, M_b, and M_n, as seen in Eqs. (4-6),(4-7), (4-9), (4-10) and (4-12)). The same quadratic modulation function was used for M_{gd}, M_{gr}, M_b, and M_n:

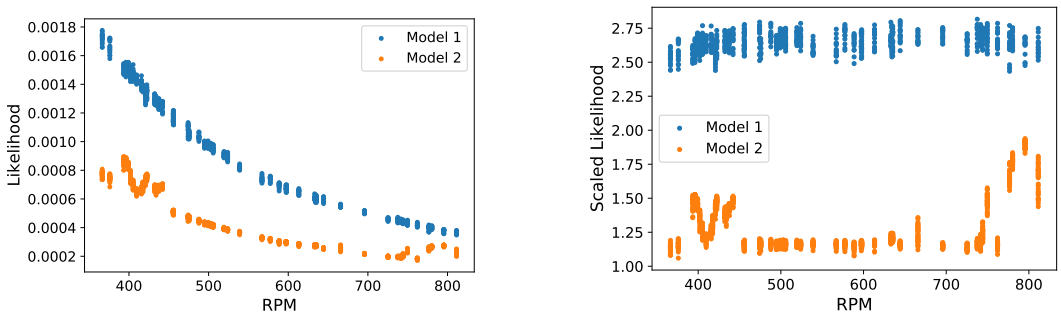
$$M(t) = \omega^2_{shaft}(t) \tag{4-17}$$

All the components that the model captures in the learned variance will therefore be amplitude modulated by this relationship in terms of the shaft speed, so the standard deviation will be a function of ω^2_{shaft}, and the variance a function of $(\omega^2_{shaft})^2$. The likelihood values were scaled as follows to make them comparable over all the operating speeds:

$$\log p(\boldsymbol{x}) = -0.5 \sum_{i}^{D_{\boldsymbol{x}}} \left(\log(2\pi) + \log(\sigma_i^2) + \frac{(x_i - \mu_i)^2}{\sigma_i^2} \right) \tag{4-18}$$

$$[\log p(\boldsymbol{x})]_{scaled} = -0.5 \sum_{i}^{D_{\boldsymbol{x}}} \left(\log(2\pi) + \log\left(\frac{\sigma_i^2}{(\omega^2_{shaft})^2} \right) + \frac{(x_i - \mu_i)^2}{\sigma_i^2} \right) \tag{4-19}$$

Fig. 33 shows how this scaling removes the correlation with the shaft speed, making it easier to compare different models' performances and to identify which operating speeds a model struggles to capture.



(a) Likelihood

(b) Scaled likelihood

Figure 33: Likelihood vs scaled likelihood

A model that fits all the healthy data well will have a high mean for the scaled likelihood of the healthy test set samples, and a low standard deviation for the scaled likelihood values (a high standard deviation will indicate that the model fits samples from a certain operating speed range much better than the other samples). The scaled likelihood means and standard deviations are listed in Tables 4 and

5 respectively. In Table 4, scaled likelihood values below 2 are classified as a poor fit (highlighted in red), above 2 a good fit (highlighted in green), and above 3.5 there is a risk of overfitting (highlighted in yellow). These classification limits were chosen based on visual inspection of the reconstructions that give likelihood values in these ranges. These classifications are only used as a rough indication of how the models fit the data. For Table 5, standard deviations larger than 0.2 times the associated mean were classified as large standard deviations, indicating poor fits (highlighted in red, with the rest highlighted in green).

Table 4: Scaled likelihood mean

Model	Pre-processing					
	SO 1-rev	SO 1-tooth	OT 1-rev	OT 1-tooth	OTA 1-rev	OTA 1-tooth
PPCA-S	1.09	1.64	2.18	2.26	2.17	2.24
PPCA-L	1.90	2.71	1.81	3.71	1.57	5.18
PCA-Z-LR-S	0.95	1.96	2.72	2.71	2.64	2.80
PCA-Z-LR-L	0.00	4.86	0.00	4.32	0.00	6.31
VAE-S	1.11	1.87	1.65	2.52	1.62	2.61
VAE-L	1.13	2.27	2.47	2.74	2.49	2.75
Decoder	1.16	1.18	2.71	1.28	2.70	2.74
PCA-C-LR-S	1.28	1.95	2.67	2.84	2.65	2.80
PCA-C-LR-L	2.47	5.10	3.00	4.55	2.83	6.62
iVAE-S	1.13	2.04	2.55	2.68	2.58	2.71
iVAE-L	1.16	2.27	2.65	2.74	2.61	2.74

Table 5: Scaled likelihood std

Model	Pre-processing					
	SO 1-rev	SO 1-tooth	OT 1-rev	OT 1-tooth	OTA 1-rev	OTA 1-tooth
PPCA-S	0.28	0.44	0.44	0.54	0.44	0.53
PPCA-L	0.72	1.09	1.19	1.24	1.16	2.15
PCA-Z-LR-S	0.35	0.33	0.07	0.36	0.07	0.31
PCA-Z-LR-L	0.00	0.78	0.00	0.75	0.00	1.69
VAE-S	0.03	0.16	0.57	0.25	0.57	0.25
VAE-L	0.03	0.18	0.10	0.27	0.09	0.27
Decoder	0.01	0.09	0.05	0.07	0.05	0.27
PCA-C-LR-S	0.18	0.27	0.07	0.30	0.07	0.29
PCA-C-LR-L	0.10	0.65	0.11	0.75	0.09	1.88
iVAE-S	0.02	0.17	0.08	0.28	0.08	0.27
iVAE-L	0.02	0.18	0.06	0.27	0.06	0.26

Table 6: Quality of fit based on scaled likelihood

Model	Pre-processing					
	SO 1-rev	SO 1-tooth	OT 1-rev	OT 1-tooth	OTA 1-rev	OTA 1-tooth
PPCA-S	Poor	Poor	Poor	Poor	Poor	Poor
PPCA-L	Poor	Poor	Poor	Over	Poor	Over
PCA-Z-LR-S	Poor	Poor	Good	Good	Good	Good
PCA-Z-LR-L	Poor	Over	Poor	Over	Poor	Over
VAE-S	Poor	Poor	Poor	Good	Poor	Good
VAE-L	Poor	Good	Good	Good	Good	Good
Decoder	Poor	Poor	Good	Poor	Good	Good
PCA-C-LR-S	Poor	Poor	Good	Good	Good	Good
PCA-C-LR-L	Good	Over	Good	Over	Good	Over
iVAE-S	Poor	Good	Good	Good	Good	Good
iVAE-L	Poor	Good	Good	Good	Good	Good

Table 6 combines the classifications in Tables 4 and 5, where a poor fit classification in any table results in a final classification of a poor fit. The overfitting classification was kept even if the standard deviation table indicates a poor fit. This was done since overfitting some of the data will cause a large standard deviation between the scaled likelihood values. Table 6 will be used throughout the rest of the chapter as a quick reference to how a model generalises to unseen healthy data. Keeping Table 3 in mind, it can already be seen that incorporating the operating condition information in the pre-processing and modelling steps improves how well the models fit the healthy data distributions, with the majority of these approaches falling into the 'Good fit' category, while the majority of the unsupervised approaches fall in the 'Poor fit' category. Also, note that the 1-tooth processed datasets are fitted better than the 1-revolution datasets. The 1-tooth processed datasets are also more prone to overfitting. This is attributed to the random gear and noise components. The longer signals contain more random information, making it more difficult for the models to simply project a large part of the random information to the latent space, and then reconstruct it accurately.

4.3.2 Homoscedastic vs. heteroscedastic models

To investigate the effect that the assumption of constant variance across the dataset has on a model's performance, the PPCA-S model and the PCA-C-LR-S model for OTA-1-revolution pre-processing are

compared. PPCA assumes a constant variance across the dataset and is referred to as a homoscedastic model. PCA-C-LR captures the variance per sample and is referred to as a heteroscedastic model. Because of the pre-processing method, the healthy data has a simple distribution. The variance can easily be captured by only one principal component since it is only influenced by the operating speed, so D_z is set to 1. This allows one to also easily plot the latent space, to see what information is captured in the latent space.

Fig. 34 plots the latent space scores of the PCA models against the associated operating speed for each healthy test set sample, in RPM. It can be seen that there is a clear correlation between the scores z_1 and the RPM, showing that the operating speed is the only information of importance to reconstruct the healthy data (the phase shift information corresponding to the shaft angular position, is removed from the data by the OTA pre-processing approach). Also note how the scores have more variance at the larger RPM values, as the variance in the input data is larger at these RPM values.

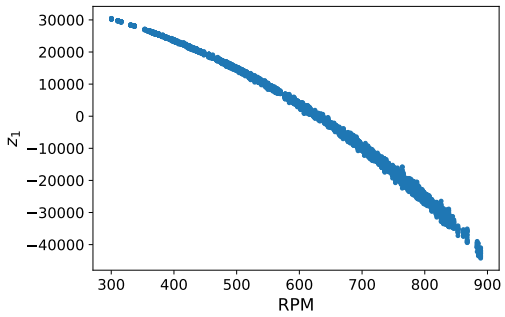


Figure 34: PCA latent space visualization for healthy test set (OTA-1-revolution, $D_z = 1$)

Now that we understand how the PCA models capture the important information in the latent space, we can look at the scaled LH for the two different models. The scaled LH values for the healthy test samples for both models are plotted against the associated RPM in Fig. 35. For the PCA-C-LR model, the scaled LH values are all in the same range, with no correlation seen with the RPM values, showing that the model successfully learned larger variances at the faster operating speeds and smaller variances at the slower operating speeds. For the PPCA model, a constant variance is assumed over the whole dataset, with the variance term being the one that would minimize the likelihood over the whole dataset. It is clear that the variance term fits the data well at around 670 RPM, but at slower speeds, the variance term is too large, decreasing the likelihood, and at faster speeds, the variance term is too small, also decreasing the likelihood.

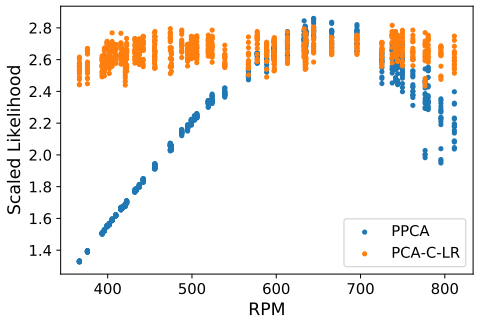


Figure 35: Scaled LH values against the associated RPM for each healthy test set sample: PPCA-S vs PCA-C-LR-S (OTA-1-revolution)

Fig. 36 shows the reconstructions and reconstruction space discrepancy signals for level 24 outer race

damage for the PCA-C-LR model. Slow and fast operating speed reconstructions and discrepancy signals are shown. Note that the variance is captured well for both the fast and slow operating speeds. Also note that the discrepancy peaks have similar magnitudes (13.5 compared to 14.5) for the fast and slow operating speeds, so the discrepancy signals are not correlated with the shaft speed, meaning that the damage level can be accurately tracked.

Fig. 37 shows the exact same plots as in Fig. 36, but this time for the PPCA model. Note how the reconstruction variance is clearly too large at the slow operating speed, and too small at the fast operating speed. Because of this, the discrepancy signal peaks are not of the same magnitude (5.5 compared to 31), even though the damage level is the same in both samples. This makes it difficult to accurately track how the damage develops, as the discrepancy signals are correlated with the operating speeds.

Fig. 38 shows how the discrepancy signal features, used for fault detection and tracking, develop as the damage level increases. The features of the PPCA-S and PCA-C-LR-S models are compared. The features are coloured based on the associated RPM, so that correlation with operating speeds can be clearly seen. Note how all the PPCA features are correlated with the RPM (except for kurtosis). This correlation shifts the discrepancy threshold to a point where only the damaged samples at fast operating speeds get classified as faulty when the damage starts to become visible in the signals. This results in a lower TPR, and the DDL occurs later on for all features (except kurtosis), as seen in Table 7.

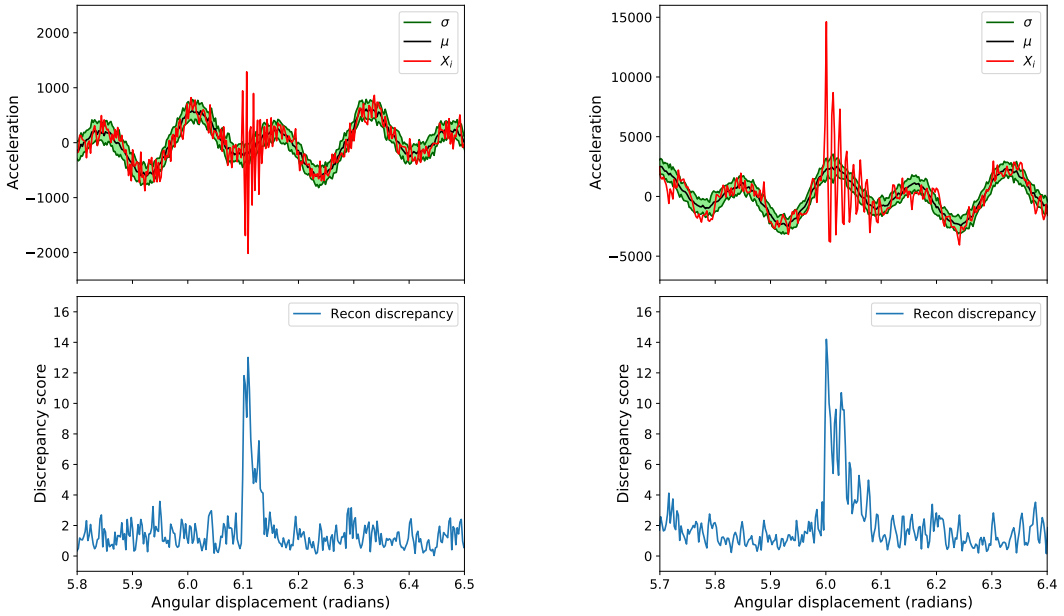
This shows that it is crucial to use heteroscedastic models, that can learn a reconstruction variance per sample, when dealing with data from varying operating conditions. This is the case for linear and nonlinear models. The VAE can be reformulated to be a homoscedastic model by only learning to reconstruct the mean, and assuming a constant reconstruction variance of 1. Even though a homoscedastic VAE model was not implemented in this work, the same correlation in the discrepancy signal features with the operating conditions can be expected, as was seen for the PPCA model.

It must also be addressed why the kurtosis feature performed just as well for the PPCA model as for the PCA-C-LR model. This is because kurtosis is scaled by the standard deviation σ of the distribution:

$$\text{Kurtosis} = \frac{1}{N}\sum_{i}\frac{(x_i - \mu_i)^4}{\sigma_i^4} \tag{4-20}$$

So even though the peaks in the discrepancy signals in Fig. 37 differ in amplitude at slow and fast operating speeds, both discrepancy signals will have the same kurtosis, as the fourth moment is scaled by the fourth power of the standard deviation. The effect of the incorrect variance is scaled away. It is however important to note that it only works in this case because the operating speed is kept constant per sample. If the operating speed fluctuates during a sample, the variance will not be relatively constant over the sample as is the case in this investigation. The scaling term σ^4 will then scale some of the points in the sample incorrectly, which will once again make the discrepancy signal feature correlated with the operating speed. So while it works in this case, it will not work when the speed varies during the sample. Heteroscedastic models should therefore be implemented.

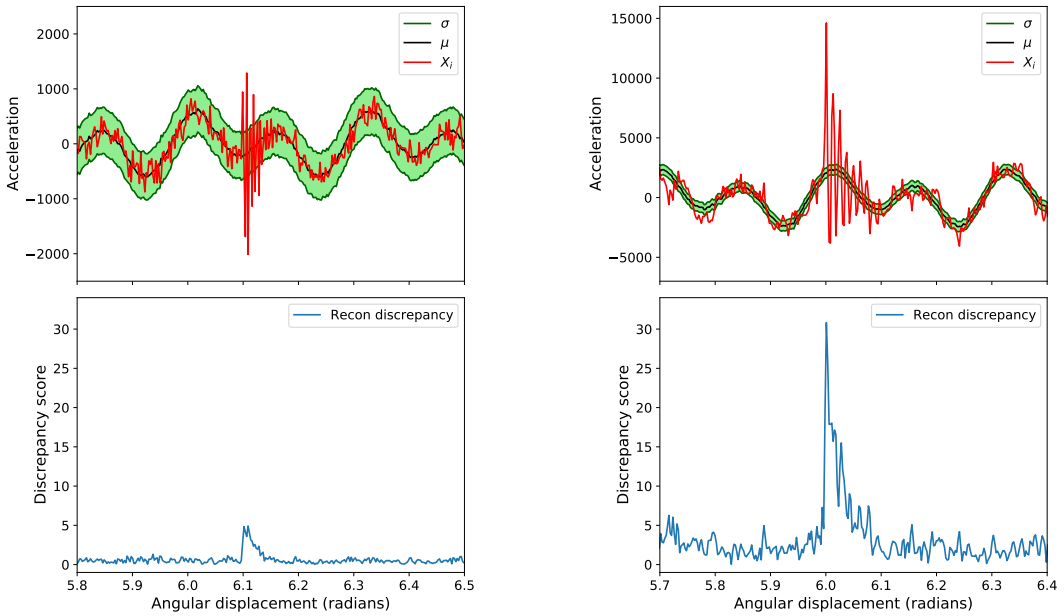
Another option not considered in this work is to make the discrepancy thresholds dependent on the operating conditions, so that the discrepancy threshold changes when the operating condition changes. In this case, it will not be a problem if the heteroscedastic noise is not captured successfully, since the varying threshold will be able to correct for it. This case is however not considered in this work, as this approach is not concerned with capturing the healthy data accurately, and will require that the operating conditions are always known, leaving no room for unsupervised approaches.



(a) Slow operating speed: 420 RPM

(b) Fast operating speed: 819 RPM

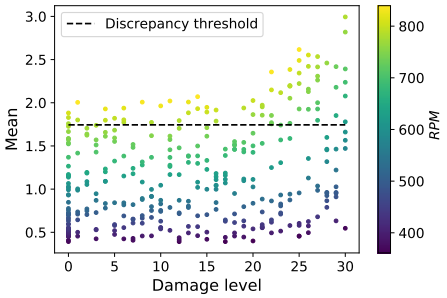
Figure 36: PCA-C-LR: Acceleration signal reconstruction and reconstruction space discrepancy signals at slow and fast operating speeds (outer race fault, damage level 24, OTA-1-revolution, $D_{\boldsymbol{z}} = 1$). Discrepancy peak ratio between samples at 420 RPM and 819 RPM: $\frac{13.5}{14.5} = 0.93$.



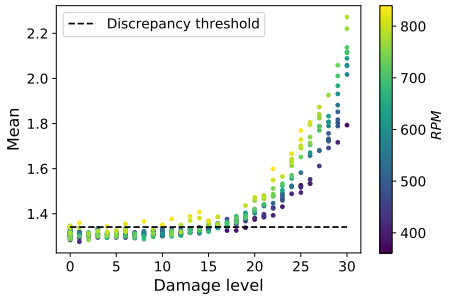
(a) Slow operating speed: 420 RPM

(b) Fast operating speed: 819 RPM

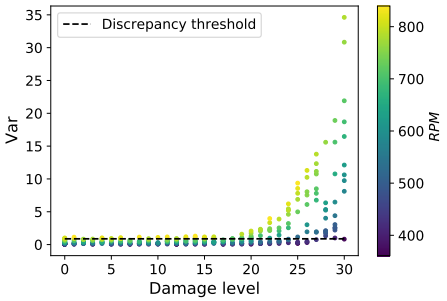
Figure 37: PPCA: Acceleration signal reconstruction and reconstruction space discrepancy signals at slow and fast operating speeds (outer race fault, damage level 24, OTA-1-revolution, $D_{\boldsymbol{z}} = 1$). Discrepancy peak ratio between samples at 420 RPM and 819 RPM: $\frac{5.5}{31} = 0.18$.



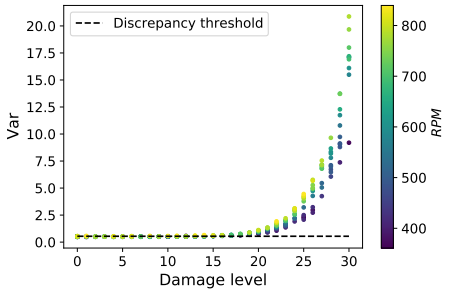
(a) PPCA: mean



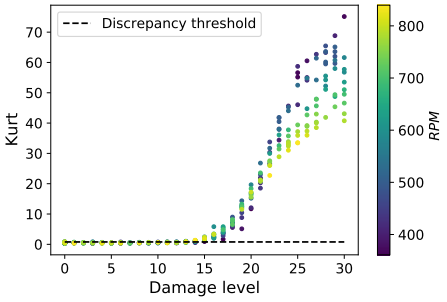
(b) PCA-C-LR: mean



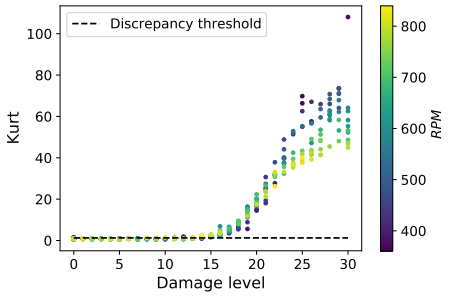
(c) PPCA: variance



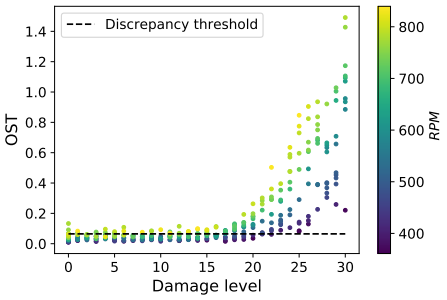
(d) PCA-C-LR: variance



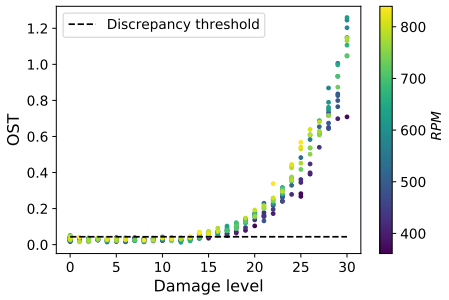
(e) PPCA: kurtosis



(f) PCA-C-LR: kurtosis



(g) PPCA: OST



(h) PCA-C-LR: OST

Figure 38: PPCA vs PCA-C-LR: discrepancy signal features for fault detection and tracking (outer race fault, OTA-1-revolution, $D_{\boldsymbol{z}} = 1$)

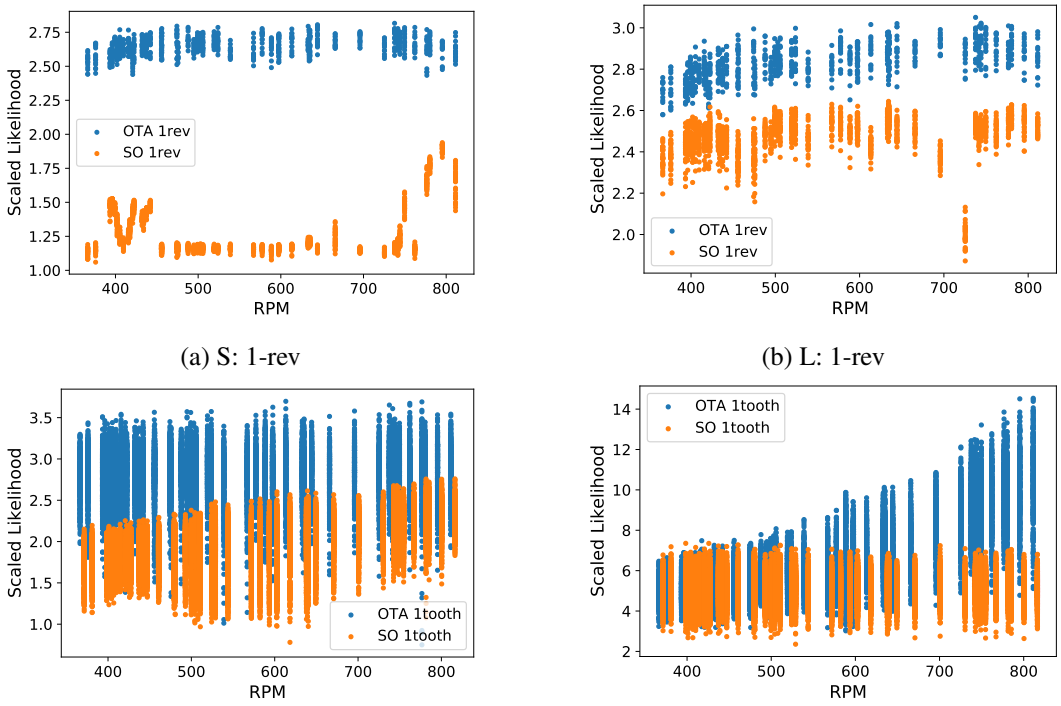
Table 7: PPCA-S vs. PCA-C-LR-S: TPR and DDL for reconstruction space discrepancy signal features (outer race fault, OTA-1-revolution, $D_{\boldsymbol{z}} = 1$)

Model	**Discrepancy space**	**TPR**				**DDL**			
		mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.22	0.35	**0.59**	0.52	31	31	**15**	23
PCA-C-LR-S	Recon	**0.53**	**0.58**	0.58	**0.55**	**20**	**15**	**15**	**16**

4.3.3 Linear PCA models

To investigate the linear PCA models, we consider the PCA-C-LR model, so that the latent space discrepancies can also be investigated. In Fig. 39 the scaled likelihoods on the healthy test samples for PCA-C-LR-S and PCA-C-LR-L models, with OTA and SO pre-processing for 1-revolution and 1-tooth, are compared. The scaled likelihood values for the healthy test set samples are plotted against the associated operating speeds.

The first thing to note is that the models fit the OTA data better in all four cases, with higher scaled likelihoods, showing that order tracking and aligning the input signal segments, make it significantly easier to accurately capture the distribution with linear models.



(a) S: 1-rev

(b) L: 1-rev

(c) S: 1-tooth

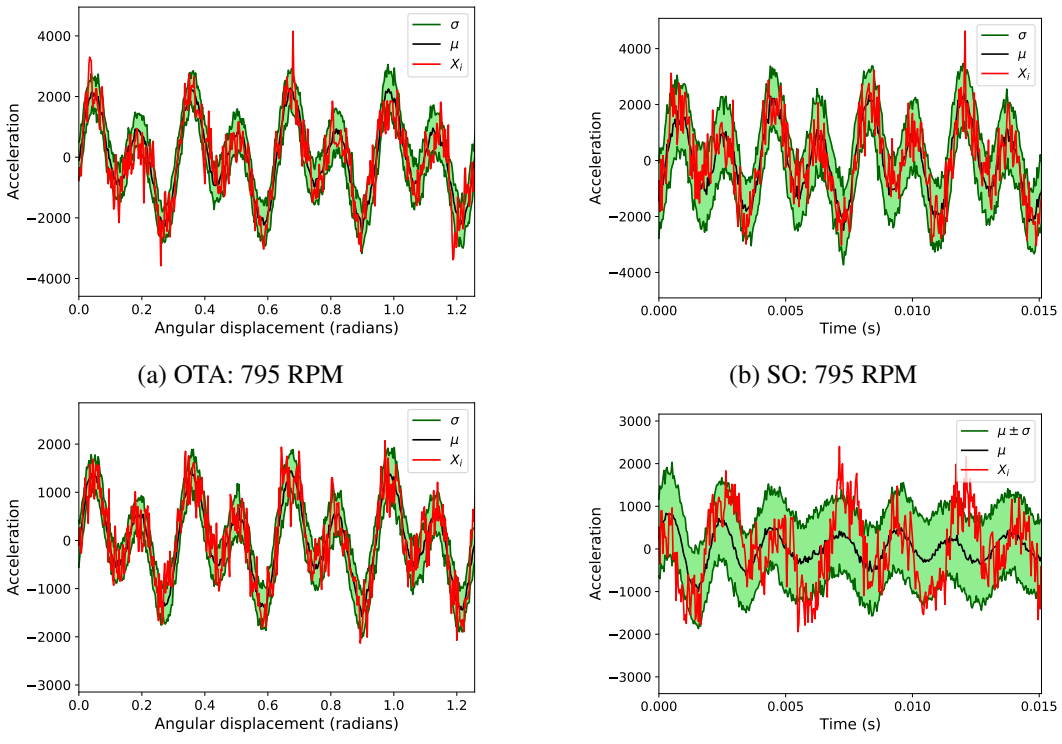
(d) L: 1-tooth

Figure 39: PCA-C-LR (S and L): Scaled likelihood values for the healthy test set samples, against the associated operating speeds, SO vs. OTA pre-processing (1-revolution and 1-tooth)

Let's first consider the case of a small latent space, for long input signal segments from 1-revolution pre-processing, shown in Fig. 39(a). The scaled likelihoods for the OTA-1-revolution data are all in the same range over the whole operating speed range, so the model fits all the data equally well. For the SO-1-revolution data, the scaled likelihoods are all lower than for the OTA-1-revolution data, so the model clearly does not fit this data as well. Notice that the model fits the samples in the 800 RPM region better than the rest of the operating speeds. The model also fits the data at around 400 RPM slightly better than the rest. This gives us important insight into how linear PCA models fit the data. The model does not have a big enough latent space (enough principal component projections) to capture data at all operating speeds represented in the training data. The model only reconstructs the samples with the highest associated variance (the samples at the fastest operating speed) accurately. So the model learns to reconstruct signals with certain frequencies, in this case, frequencies around 800 RPM. The model is also able to reconstruct the samples at 400 RPM since 800 RPM is a harmonic frequency of 400 RPM. The model is however not able to accurately capture the variance for these regions at 800 RPM and 400 RPM, as the variance now no longer monotonically increases as the

operating speed increases. The variance at 600 RPM is higher than at 800 RPM, because the model reconstructs the mean at 800 RPM more accurately than at 600 RPM.

This is confirmed when plotting reconstructions for samples in both the 800 RPM and 600 RPM ranges, as shown in Fig. 40. Because the OTA-1-revolution samples are order tracked, all the signals consist of the same frequency components. The model must only learn to capture these frequency components, and the PCA model does so very well, fitting both the samples in the 600 RPM and 800 RPM ranges. For the SO-1-revolution samples, the model learned to reconstruct the frequencies associated with the 800 RPM range, and it can be seen that it accurately reconstructs the mean for the 800 RPM sample. The sample in the 600 RPM range has different frequency components than the sample in the 800 RPM range, and the model fails to reconstruct this sample accurately. The poor mean reconstructions affect the linear regression model's ability to map the OC info to the reconstruction error variance. This is problematic, since all of the samples that are not close to 800 RPM or 400 RPM, will have discrepancy signals with a component of shaft order 20 (the order associated with the 20 gear teeth meshing) present, even for healthy samples, as this component is not captured by the model. This makes anomaly detection much more difficult, as healthy samples from the 800 RPM or 400 RPM operating speed region will have much smaller discrepancy scores than healthy samples from other operating speeds.



(a) OTA: 795 RPM

(b) SO: 795 RPM

(c) OTA: 635 RPM

(d) SO: 635 RPM

Figure 40: PCA-C-LR-S: OTA-1-revolution vs SO-1-revolution reconstructions

This shows that while PCA models have the potential to fit order-tracked data accurately with only a few principal components, it is not the case for data from varying speed conditions that are not order-tracked.

One should now ask the question: what if we just give the linear PCA model more principal components to work with? In Fig. 39(b) the latent space size is increased. The model now has enough

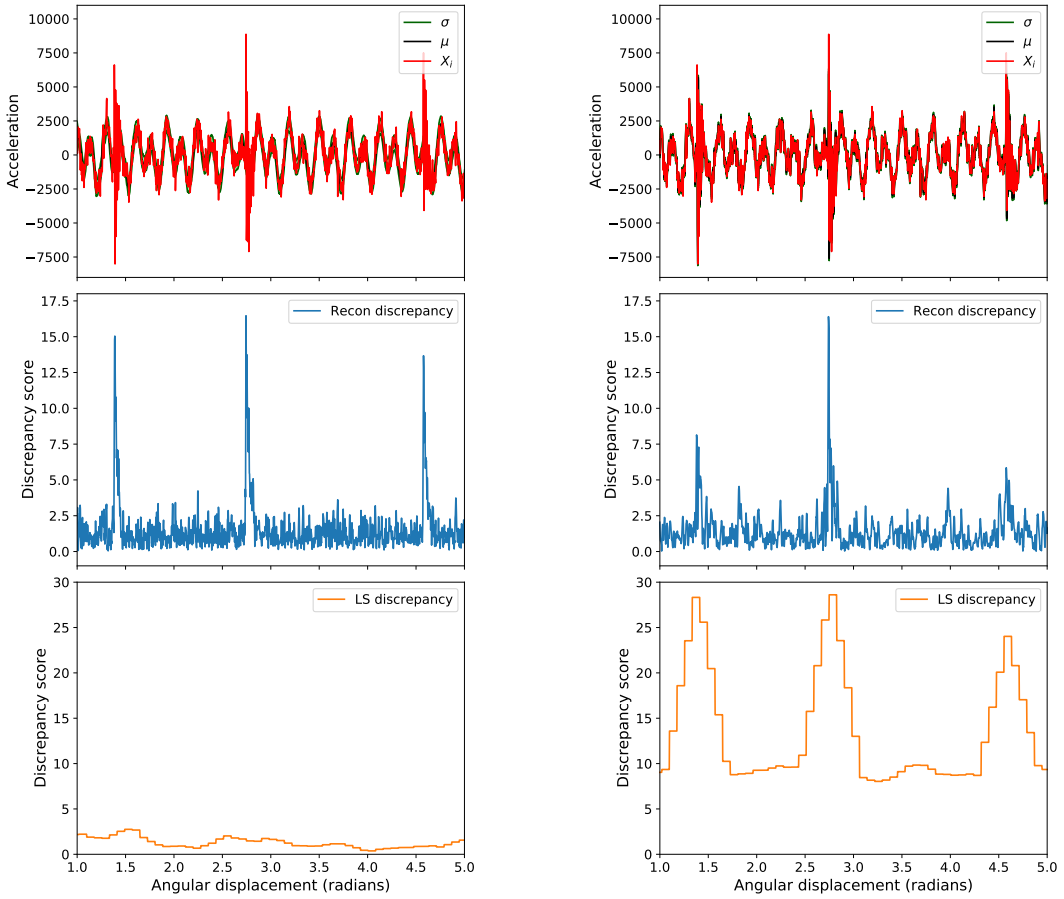
principal components to capture 95% of the variance in the training data. The model fits the SO-1-revolution data much better, with only samples around 720 RPM being reconstructed poorly. This however shows that the model is still only able to reconstruct certain frequencies, as it doesn't generalize to the frequencies in the 720 RPM region. The linear PCA model has no 'understanding' of the concept of frequency shifts, as this is a highly nonlinear operation. The model simply learns to reconstruct signal components at certain frequencies along with its harmonics, so the latent representation only indicates if these frequencies are present in the input signal or not. The frequencies that are not present in the training data are reconstructed poorly. By giving the model enough principal components to capture 95% of the variance in the training data, one also runs the risk of the model starting to overfit. The bearing faults might be reconstructed accurately with a combination of the frequencies that get recorded in the latent space. Therefore, it is crucial to also monitor the latent space for discrepancies when giving the PCA models so many principal components to work with, since discrepancies in the PCA scores will give a better indication of damage than the reconstruction space, as was discussed in Section 1.4.3.

For the shorter input signals from the 1-tooth pre-processing methods, the reconstruction of the SO samples improves. This is because the model can reconstruct a short signal segment consisting of unlearned frequency components, relatively accurately using a combination of other learned frequencies, but this becomes less accurate the longer the signal segment is. Fig. 39(c) and (d) show the improved fits on the shorter input signals. For the larger latent spaces, the models clearly overfit the data, especially on the OTA-1-tooth samples from higher operating speeds.

Seeing that overfitting will definitely play a role when choosing larger latent spaces for linear models, we now look at how the damage presents itself in the reconstruction space and latent space for different latent space sizes. Fig. 41 compares the reconstructions, reconstruction space discrepancy signals and latent space discrepancy signals obtained by using PCA-C-LR-S and PCA-C-LR-L models, for the same sample with an outer race fault at level 23. The data was pre-processed with the OTA-1-tooth method. The shorter input window lengths allow for a finer resolution in the latent space discrepancy signals.

With a small latent space, the PCA model does not reconstruct the three impulses caused by the outer race fault at all, and all three impulses are captured in the reconstruction discrepancy signal, with similar amplitudes for each impulse. The latent space discrepancy signal shows no indication of damage. With a large latent space, the model reconstructs most of the first and third impulses, resulting in smaller discrepancy amplitudes for these two impulses in the reconstruction discrepancy signal. This time the damage is detected better in the latent space discrepancy signal, showing three spikes with similar amplitudes.

Table 8 shows how combining the results of the reconstruction space discrepancy feature monitoring with the latent space discrepancy feature monitoring, improves both the TPR and the DDL. Note that for the small latent space, adding the latent space discrepancy signal only slightly improves the results obtained using only the reconstruction discrepancy signal. For the large latent space, adding the latent space discrepancy signal drastically improves the results obtained when using only the reconstruction discrepancy signal. If the latent space is not monitored with PCA models, overfitting (using too many principal components) will drastically worsen the model's performance.



(a) Small latent space

(b) Large latent space

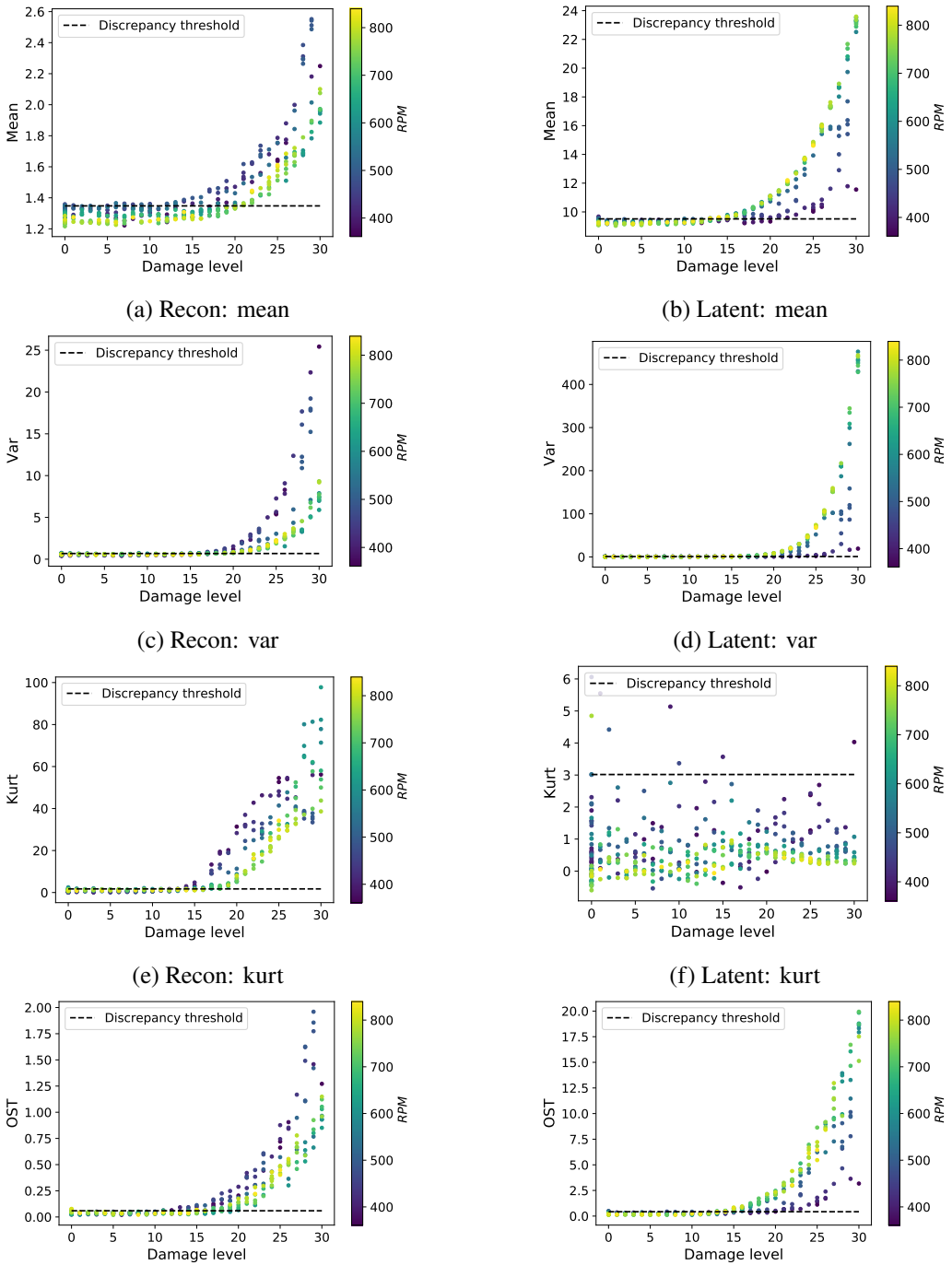
Figure 41: PCA-C-LR-S vs PCA-C-LR-L: reconstructions, reconstruction space discrepancy signals and latent space discrepancy signals (outer race fault, damage level 24, OTA-1-tooth)

Table 8: TPR and DDL for PCA-C-LR model with small and large latent spaces (Outer race fault, OTA-1-tooth)

Model	**Discrepancy space**	**TPR**				**DDL**			
		mean	var	kurt	OST	mean	var	kurt	OST
PCA-C-LR-S	Recon	0.48	0.57	0.60	**0.50**	**18**	**15**	**15**	**18**
	Latent	0.11	0.07	0.11	0.05	31	31	31	31
	Recon+Latent	**0.50**	**0.59**	**0.63**	**0.50**	**18**	**15**	**15**	**18**
PCA-C-LR-L	Recon	0.45	0.47	0.52	0.49	21	**18**	**17**	20
	Latent	0.53	0.50	0.02	0.54	21	21	31	20
	Recon+Latent	**0.60**	**0.54**	**0.53**	**0.59**	**18**	**18**	**17**	**15**

To understand why combining the reconstruction space discrepancy feature monitoring with the latent space discrepancy feature monitoring, improves both the TPR and the DDL so much for the PCA-C-LR-L model, the monitored features from both the reconstruction and latent space discrepancy signals, are plotted in Fig. 42. Note how in this specific case, the damaged samples corresponding to slower RPM values are detected first in the reconstruction space (purple dots cross the discrepancy line first), but in the latent space, the damaged samples corresponding to faster RPM values are detected first (yellow dots cross the discrepancy line first). With the reconstruction space being better for detecting damaged samples at slow operating speeds, and the latent space for damaged samples at fast operating speeds, combining the two leads to overall better performance. Also, note that monitoring

the kurtosis of the latent space discrepancy does not give a clear indication of the damage. This is due to the lower resolution of the latent space discrepancy signal (as was shown in Fig. 15(c)), which smooths out the impulsive components in the discrepancy signal to the point where the discrepancy signal values are almost normally distributed, making kurtosis less effective.



(a) Recon: mean

(b) Latent: mean

(c) Recon: var

(d) Latent: var

(e) Recon: kurt

(f) Latent: kurt

(g) Recon: OST

(h) Latent: OST

Figure 42: PCA-C-LR-L: monitored features from the reconstruction and latent space discrepancy signals

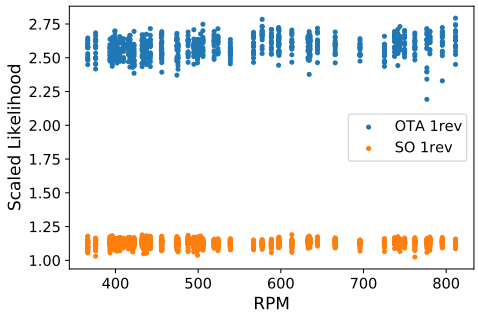
4.3.4 Nonlinear VAE models

To investigate the nonlinear VAE models, we consider the iVAE model, so that the latent space discrepancies can also be investigated. In Fig. 43 the scaled likelihoods on the healthy test samples for iVAE-S and iVAE-L models, with OTA and SO pre-processing for 1-revolution and 1-tooth, are compared, the same as was done for PCA-C-LR in Fig. 39. Looking at the four plots, the following can be noted:

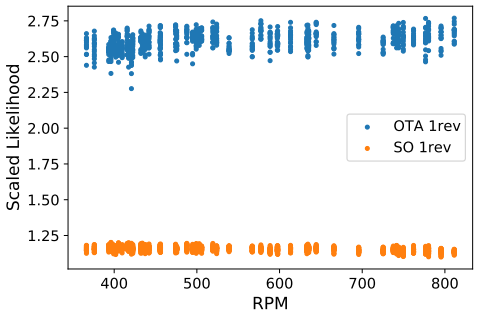
- The scaled likelihood values improve only slightly when increasing the size of the latent space. This shows that the nonlinear VAE models that use variational inference, are less prone to overfitting (learning an identity function transformation) when given larger latent spaces, than the linear PCA models. The regularization applied by the latent space prior successfully prevents overfitting.
- The scaled likelihood values for a specific pre-processing method and latent space size, fall in the same range for samples over the entire operating speed range. This shows that the nonlinear models do not learn to reconstruct certain frequencies and their harmonics, as was seen to be the case for the PCA models.
- The OTA pre-processed data is captured well in all four cases, showing that order tracking and aligning the input signal segments, make the learning task easier for the nonlinear models. This corresponds with the results shown in Ziyin et al. (2020), where it is shown that the activation functions commonly used in neural network architectures prevent the networks from capturing periodic functions accurately, since the models have no inductive periodic bias. The models have no understanding of how frequency shifts influence signal segments. Until this is addressed, one cannot expect nonlinear deep learning models to fit the signals in the time domain just as well as it does in the angle domain.
- While the iVAE models capture the SO-1-tooth pre-processed data fairly well, it breaks down on the longer SO-1-revolution input signals.

Looking at reconstructions from the iVAE-L model for SO-1-revolution and SO-1-tooth pre-processing, shown in Fig. 44, it is clear that posterior collapse occurs when using longer input signals that are not order tracked. Even with the more informed prior in the iVAE, the encoder still struggles to encode the long input signals with varying frequencies and amplitudes to sensible latent representations. There is too much noise in the latent space, causing the decoder to ignore the latent representations, and only capture the mean and variance of the dataset. With the shorter input lengths, interpolation between signal segments consisting of different frequencies is easier, and the model manages to capture the healthy data.

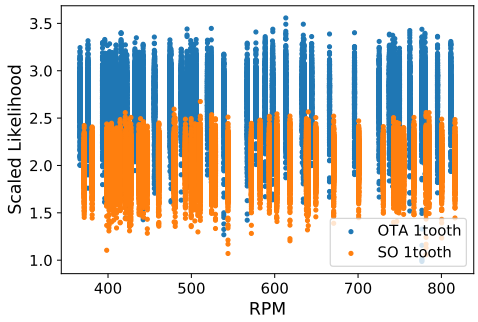
With nonlinear models being less prone to overfitting (reconstructing any input data) due to the appropriate latent space regularization, it is expected that adding the latent space discrepancy signal to the condition monitoring process will only slightly improve the results obtained when using only the reconstruction discrepancy signal. This is confirmed in Table 9, which shows the TPR and the DDL, for both reconstruction space discrepancy feature monitoring and latent space discrepancy feature monitoring separately, as well as when the two discrepancy signals are combined. The results improve slightly when combining the two, but this suggests that nonlinear models that do not incorporate latent space discrepancy monitoring, such as the VAE, might still perform well using only the reconstruction space discrepancy signals, even when a bigger latent space is used.



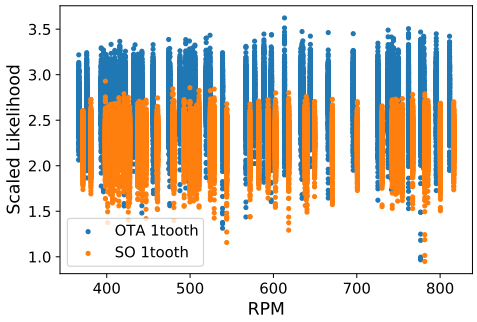
(a) S: 1-rev



(b) L: 1-rev

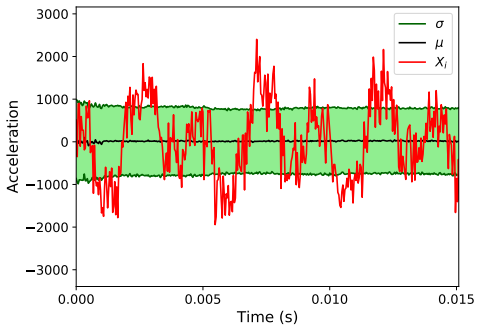


(c) S: 1-tooth

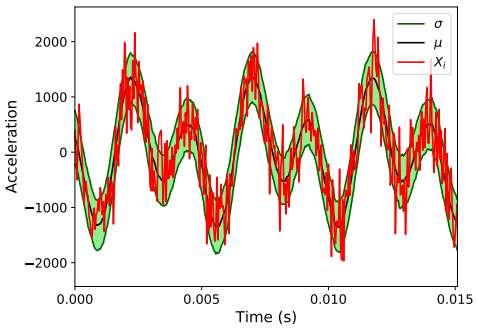


(d) L: 1-tooth

Figure 43: iVAE (S and L): Scaled likelihood values for the healthy test set samples, against the associated operating speeds, SO vs. OTA pre-processing (1-revolution and 1-tooth)



(a) SO-1-rev



(b) SO-1-tooth

Figure 44: iVAE-L: SO-1-revolution vs SO-1-tooth reconstructions (635 RPM)

Table 9: TPR and DDL for iVAE model with small and large latent spaces (Outer race fault, OTA-1-tooth)

Model	**Discrepancy space**	**TPR**				**DDL**			
		mean	var	kurt	OST	mean	var	kurt	OST
iVAE-S	Recon	0.41	0.54	0.63	0.47	21	**16**	**14**	**19**
	Latent	0.43	0.44	0.04	0.40	23	22	31	23
	Recon+Latent	**0.47**	**0.56**	**0.64**	**0.48**	**20**	**16**	**14**	**19**
iVAE-L	Recon	0.50	0.59	0.61	0.47	**20**	**15**	**15**	**19**
	Latent	0.44	0.44	0.12	0.41	23	21	31	21
	Recon+Latent	**0.53**	**0.60**	**0.65**	**0.48**	**20**	**15**	**15**	**19**

4.3.5 Monitoring the kurtosis of discrepancy signals

In the few examples shown in this section, it is often seen that the kurtosis feature produces the best result. This can be attributed to two reasons:

1. The nature of bearing faults: bearing faults cause cyclostationary impulses in the measured signal at very high frequencies. Kurtosis is much more sensitive to impulsive components than the other features.

2. The data is relatively simple, with no large impulsive components present in the healthy data. This allows the kurtosis measure to be highly sensitive to impulsive components appearing in the discrepancy signal.

It is however common for real-world data to have impulsive components present even in the healthy data. These components drastically affect the performance when using kurtosis as a feature, as will be shown in Chapter 5. It is therefore advised to keep this in mind when looking at the results presented in the following subsection. While kurtosis gives the best results in most cases, the focus should rather be on how the mean, variance, and OST features perform on fault detection and tracking tasks, as these features give a better indication of how well the model captures the healthy data.

4.4 Result Analysis

Only results from the outer race fault dataset are presented in this subsection, as similar trends were seen for both inner and outer race fault monitoring. Only the DDL results are shown in this section, since the TPR and AUC results follow similar trends. The full DDL, TPR and AUC result tables for both the outer and inner race faults are given in Appendix E.

4.4.1 Unsupervised approach

Table 10: Phenomenological dataset analysis results: DDLs for unsupervised approach, where no OC information was used (green: <18, red: >24)

Pre-processing	**Model**	**Discrepancy space**	**Feature**			
			mean	var	kurt	OST
SO-1-rev	PPCA-S	Recon	31	31	18	22
	PPCA-L	Recon	31	27	13	20
	PCA-Z-LR-S	Recon	31	31	31	31
	PCA-Z-LR-L	Recon	31	31	31	31
	VAE-S	Recon	31	24	15	18
	VAE-L	Recon	31	22	15	17
SO-1-tooth	PPCA-S	Recon	31	27	15	21
	PPCA-L	Recon	31	27	13	20
	PCA-Z-LR-S	Recon	31	24	15	20
	PCA-Z-LR-L	Recon	31	31	31	29
	VAE-S	Recon	31	21	13	18
	VAE-L	Recon	23	18	13	15

The DDLs for the model and pre-processing combinations where no OC information was used during modelling or pre-processing are presented in Table 10. DDLs below level 18 are highlighted in green, and above level 24 in red, so that one can easily get an idea of the overall performance. Only the reconstruction space is monitored for discrepancies, since OC information is not available to learn the expected latent space representations for a healthy machine at certain OCs. The results are interpreted as follows:

- Ignoring the results for kurtosis, the results for the rest of the monitored features are poor on average, showing little promise that one can robustly monitor faults in a completely unsupervised setting.

- The poor results for PPCA (S and L) are attributed to the features being correlated with the operating speed, because of the assumption of constant variance across the dataset, as was shown in Section 4.3.2.
- The poor results for PCA-Z-LR-S are attributed to the models not reconstructing all frequencies equally well, similar to the PCA-C-LR-S SO-1-revolution results shown in Fig. 39(a).
- The poor results for PCA-Z-LR-L are attributed to the models not being able to accurately infer the variance in the data from the latent representations $\boldsymbol{z}$. The models use enough latent variables to reconstruct most frequencies accurately, but most of the noise present in the data gets mapped to the latent representations, making it difficult for the linear regression model to learn the variance from $\boldsymbol{z}$.
- The VAE models' reconstructions collapsed for the SO-1-revolution data, similar to the iVAE reconstructions shown in Fig. 44(a).
- The only model that performed fairly well was the VAE-L model on the SO-1-tooth data. The input data windows are short enough that the model can learn to map the data to the latent space, even though the model does not learn an 'understanding' of phase and frequency shifts.
- The models struggle to accurately fit the data as it is not order-tracked. Most of the variance in the data is caused by phase and frequency shifts. The models are not formulated to capture periodic signals with phase and frequency shifts well.

4.4.2 Using OC information for pre-processing

Table 11: Phenomenological dataset analysis results: DDLs where available OC information was used only during pre-processing (green: <18, red: >24)

Pre-processing	**Model**	**Discrepancy space**	**Feature**			
			mean	var	kurt	OST
OT-1-rev	PPCA-S	Recon	31	31	15	23
	PPCA-L	Recon	31	27	14	21
	PCA-Z-LR-S	Recon	22	18	14	18
	PCA-Z-LR-L	Recon	31	31	31	31
	VAE-S	Recon	31	31	21	26
	VAE-L	Recon	26	19	15	19
OT-1-tooth	PPCA-S	Recon	31	28	15	22
	PPCA-L	Recon	31	26	15	19
	PCA-Z-LR-S	Recon	31	24	15	21
	PCA-Z-LR-L	Recon	31	21	16	22
	VAE-S	Recon	27	19	15	19
	VAE-L	Recon	21	18	14	18
OTA-1-rev	PPCA-S	Recon	31	31	15	23
	PPCA-L	Recon	31	28	15	21
	PCA-Z-LR-S	Recon	21	17	15	18
	PCA-Z-LR-L	Recon	31	31	31	31
	VAE-S	Recon	31	31	21	26
	VAE-L	Recon	24	18	15	19
OTA-1-tooth	PPCA-S	Recon	31	28	15	21
	PPCA-L	Recon	31	24	14	18
	PCA-Z-LR-S	Recon	23	18	15	18
	PCA-Z-LR-L	Recon	21	21	19	23
	VAE-S	Recon	26	18	15	19
	VAE-L	Recon	21	18	14	19

The DDLs for the model and pre-processing combinations where the OC information is used only during pre-processing are presented in Table 11. Only the reconstruction space is monitored for

discrepancies, since OC information is not available for modelling to learn the expected latent space representations for a healthy machine at certain OCs. The results are interpreted as follows:

- Ignoring the results for kurtosis, the results for the rest of the monitored features are slightly better on average than for the unsupervised approach, showing that the models capture the order-tracked data distributions better.
- The poor results for PPCA (S and L) are again attributed to the features being correlated with the operating speed (because of the assumption of constant variance across the dataset).
- The PCA-Z-LR-S models perform fairly well on all four pre-processing methods. This shows that for the smaller PCA latent space, $\boldsymbol{z}$ is a good proxy for the operating conditions, so the linear regression model is able to infer the variance in the data from the latent representation $\boldsymbol{z}$.
- The results for PCA-Z-LR-L are mixed. For the longer input signals from OTA-1-revolution and OT-1-revolution, the model fits the data poorly. The models are not able to accurately infer the variance in the data from the latent representations $\boldsymbol{z}$, since a lot of the noise is captured in the latent space. In this case, the results are very poor. For the shorter input signals from OTA-1-tooth and OT-1-tooth, the models completely overfit the data, so the variances that the linear regression model must learn are all very small. In this case, the signal components associated with the damage get reconstructed accurately, and therefore the damage cannot be detected early on in the reconstruction space. The damage would have been detected earlier in the latent space, but because the OC info is not used during modelling, the expected latent space representations at a certain operating condition cannot be learned.
- The VAE-S model struggles to fit the data, especially for the longer 1-revolution input signals. The VAE-L model performs better, with the best results achieved on the shorter 1-tooth input signals. It is surmised that the VAE-S latent space is too small, resulting in the latent space representations being pulled to the origin, as described in Section 3.5.3 which causes posterior collapse, while the larger latent space in the VAE-L can learn better latent representations because of the 'soap-bubble'-effect.
- The models perform relatively the same on the OT and OTA pre-processing approaches. No improvement is seen in this case when aligning the input signals.

4.4.3 Using OC information for modelling

Table 12: Phenomenological dataset analysis results: DDLs where available OC information was used only during modelling (green: <18, red: >24)

Pre-processing	Model	Discrepancy space	Feature			
			mean	var	kurt	OST
SO-1-rev	C-decoder	Recon	26	21	15	18
	PCA-C-LR-S	Recon+Latent	28	23	18	21
	PCA-C-LR-L	Recon+Latent	16	15	15	14
	iVAE-S	Recon+Latent	27	21	15	18
	iVAE-L	Recon+Latent	27	21	15	17
SO-1-tooth	C-decoder	Recon	24	19	15	18
	PCA-C-LR-S	Recon+Latent	18	18	15	18
	PCA-C-LR-L	Recon+Latent	14	13	14	14
	iVAE-S	Recon+Latent	21	16	13	15
	iVAE-L	Recon+Latent	21	16	13	15

The DDLs for the model and pre-processing combinations where the OC information is used only during modelling are presented in Table 12. The results are interpreted as follows:

- Ignoring the results for kurtosis, a clear improvement can be seen in the results for the rest of the monitored features over the unsupervised approach, and the approach where OC information is used only for pre-processing.
- While the C-decoder, PCA-C-LR-S, iVAE-S and iVAE-L models all still struggle to capture the distribution of the healthy data for the longer SO-1-revolution input signals, PCA-C-LR-L performs very well. Because of the large PCA latent space, the model can reconstruct most frequencies, and the linear regression model easily learns the relationship between $\boldsymbol{c}$ and the variance in the data. Where the model starts to overfit, the damage can be detected in the latent space. This results in a good performance on SO-1-tooth data as well.
- All the models perform better on the SO-1-tooth data than on the SO-1-revolution data, which is expected. Because the models do not learn an actual 'understanding' of phase and frequency shifts, it is easier for the models to map short signal segments to the latent space and then back to the data space.
- Note that the C-decoder struggles more than the other models for both the SO-1-revolution and SO-1-tooth data. This again shows that it is not easy for the neural network architecture to learn a good understanding of how varying frequency and phase values translate to a time-series signal, even when the frequency and phase values are given to the network as inputs. The non-linear models perform better when they can encode the input data to more abstract representations that capture the phase and frequency, and then decode the data from there.

4.4.4 Using OC information for pre-processing and modelling

Table 13: Phenomenological dataset analysis results: DDLs where available OC information was used during pre-processing and modelling (green: <18, red: >24)

Pre-processing	**Model**	**Discrepancy space**	**Feature**			
			mean	var	kurt	OST
OT-1-rev	C-decoder	Recon	21	18	14	18
	PCA-C-LR-S	Recon+Latent	19	15	15	18
	PCA-C-LR-L	Recon+Latent	18	14	15	16
	iVAE-S	Recon+Latent	23	17	15	18
	iVAE-L	Recon+Latent	23	18	15	19
OT-1-tooth	C-decoder	Recon	23	18	18	20
	PCA-C-LR-S	Recon+Latent	18	15	15	18
	PCA-C-LR-L	Recon+Latent	18	18	15	16
	iVAE-S	Recon+Latent	20	16	15	18
	iVAE-L	Recon+Latent	20	15	15	18
OTA-1-rev	C-decoder	Recon	21	18	14	18
	PCA-C-LR-S	Recon+Latent	20	15	15	16
	PCA-C-LR-L	Recon+Latent	18	14	15	16
	iVAE-S	Recon+Latent	24	18	14	18
	iVAE-L	Recon+Latent	23	17	15	19
OTA-1-tooth	C-decoder	Recon	19	16	15	18
	PCA-C-LR-S	Recon+Latent	18	15	15	18
	PCA-C-LR-L	Recon+Latent	18	18	17	15
	iVAE-S	Recon+Latent	20	16	14	19
	iVAE-L	Recon+Latent	20	15	15	19

The DDLs for the model and pre-processing combinations where the OC information was used during both pre-processing and modelling are presented in Table 12. The results are interpreted as follows:

- Ignoring the results for kurtosis, it can easily be seen that the models' performances are better and more robust than the approaches shown before. Damage is detected in all cases before level 25 (no red highlighted cells in the table).

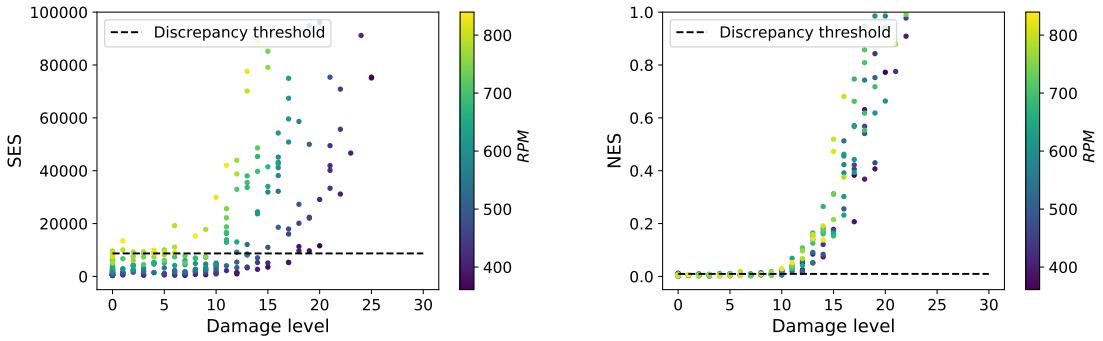
- No clear difference in performance can be seen on the OT-1-revolution, OT-1-tooth, OTA-1-revolution and OTA-1-tooth pre-processed data, with the models capturing the healthy data distributions well in all cases, and where overfitting occurs, the damage can be detected in the latent space.
- Note that the C-decoder performs just as well as the latent variable models. If the shaft speed and phase information are available, it is not necessary to consider latent variable models for this dataset, since the variance in the healthy data can be sufficiently modelled using only the shaft speed and phase information. This also shows that even though the nonlinear models should theoretically be able to capture input data that is not order tracked, containing varying frequencies, the learning task is greatly simplified when the input signals are order tracked.
- The iVAE models also perform notably better on the order tracked data than the VAE models, showing that the prior conditioned on the OC info, allows the model to learn better latent representations than the uninformed zero-mean, isotropic unit variance prior used in the VAE. The iVAE-S captures the data just as well as the iVAE-L, since 10 latent variables are more than enough to capture the phase shifts, and the amplitude modulation caused by the varying shaft speeds.

4.4.5 Signal processing benchmark

The DDL, TPR and AUC benchmarks obtained by monitoring the SES and NES for the unfiltered and bandpass filtered datasets are documented in Table 14. As expected, bandpass filtering the data around the frequency band that contains the impulsive bearing component improves the results. The NES performs considerably better than the SES. Fig. 45 shows how the SES and NES at order 4.12 for the bandpass filtered dataset develop over time as the damage level increases. The SES is highly correlated with the shaft speed, which causes poor results. Normalizing the spectrum, as is done with the NES, effectively removes the correlation between the spectrum magnitudes and the shaft speed, which leads to the damage being detected as early as level 11.

Table 14: Phenomenological model dataset signal processing benchmark

Metric	**No filter**		**Bandpass filtered (6000 - 8000 Hz)**	
	SES @ order 4.12	NES @ order 4.12	SES @ order 4.12	NES @ order 4.12
DDL	24	16	18	11
TPR	0.47	0.61	0.66	0.73
AUC	0.78	0.77	0.86	0.89



(a) SES @ order 4.12

(b) NES @ order 4.12

Figure 45: The SES and NES at order 4.12 for the bandpass filtered dataset as the damage level increases.

Tracking the NES on the bandpass-filtered dataset outperforms all the learning-based approaches, where damage was detected at the earliest at level 13. Since almost all the relevant fault information is captured in the frequency band between 6000 Hz and 8000 Hz, and all the irrelevant information that complicates the damage detection task lies outside this band, it is not a surprise that the signal processing approach performs better than the learning-based approaches. The only benefit of following the learning-based approach on this dataset, is that the results are not dependent on identifying a frequency band of interest where the fault presents itself.

While it might be possible to improve the results of the learning-based approaches by applying further signal processing to the discrepancy signals (e.g. bandpass filtering the discrepancy signals before enveloping, and squaring the envelope), the goal of this investigation was to see how well the discrepancy analysis methods perform with little to no domain knowledge built into the methods. When it is possible to filter out all the components except the component associated with the fault, it can be expected that monitoring the filtered signal will perform better than monitoring a discrepancy signal, since the discrepancy signal will still contain some noise across the frequency spectrum.

It should however also be noted that data from a phenomenological model is expected to be less complex than real-world data. In real-world data, the damage might present itself in multiple frequency bands, and there might be other impulsive components not associated with damage that also appear in these frequency bands. The potential benefits of learning-based approaches over signal processing approaches on such complex datasets are explored in Chapter 5.

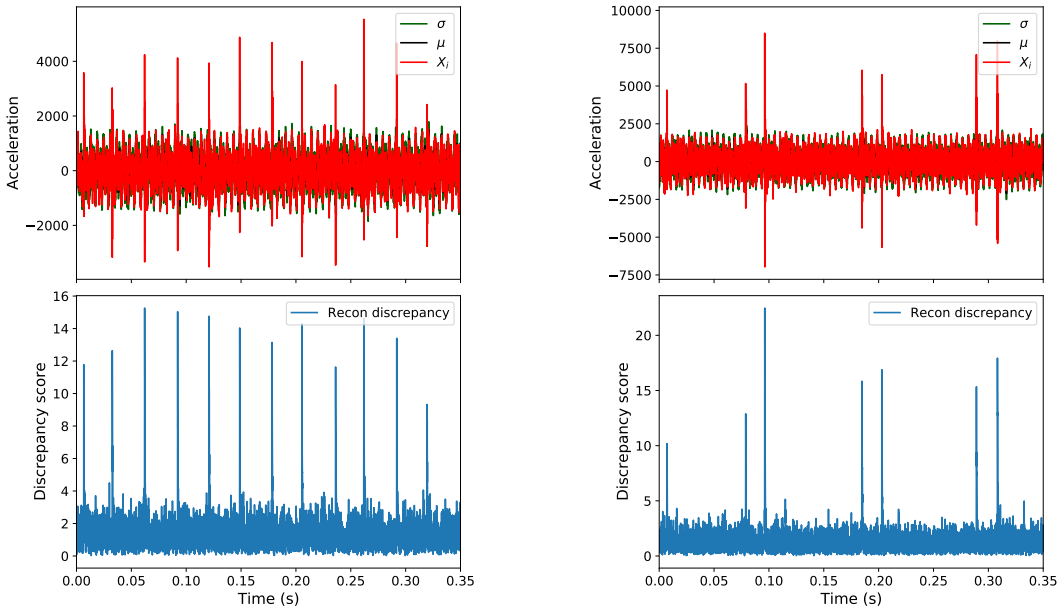
4.5 Fault classification: outer race vs. inner race fault

This section demonstrates the potential of this proposed discrepancy analysis approach for fault classification tasks. The reconstruction space discrepancy signals for the PCA-C-LR-L model, with SO-1-revolution pre-processing, are presented. This combination was chosen, as it performed well on the fault detection and tracking tasks compared to the other learning-based approaches, fitting the distribution of the healthy data well, while not reconstructing the damage components present in the evaluation data.

Fig. 46 shows the reconstruction of samples with outer and inner race faults, at damage level 22 (sample nr. 350), along with the associated reconstruction space discrepancy signals. From the reconstruction, it can be seen that the impulses associated with the bearing faults are not reconstructed, resulting in spikes in the discrepancy signals. From the discrepancy signal plots, the differences between the outer and inner race faults can clearly be seen. The spikes in the discrepancy signal for the outer race fault are all relatively the same amplitude, as the fault stays in the same load zone, while the spikes in the discrepancy signal for the inner race fault vary as the fault moves in and out of the load zone.

Without knowing the fault frequencies, the faults can still be classified as an inner race or outer race fault, based only on the structure of the discrepancy signals in the time domain. This will however become slightly more difficult when the speed varies significantly per sample, with the spikes moving closer or further from each other. By order tracking the discrepancy signals, the discrepancy signals will keep the same form at different operating speeds. Another challenge in time domain classification is noise. By taking the FFT of the order-tracked discrepancy signal, an interpretable representation of the fault present can be found, that is more robust to noise than the time domain discrepancy signal. Fig. 47 shows how the order spectra of the reconstruction space discrepancy signals for outer and inner race faults develop over time as the fault level increases. The outer race fault has a clear peak at the fault order of 4.12, along with peaks at its harmonics. The inner race fault has peaks at the fault

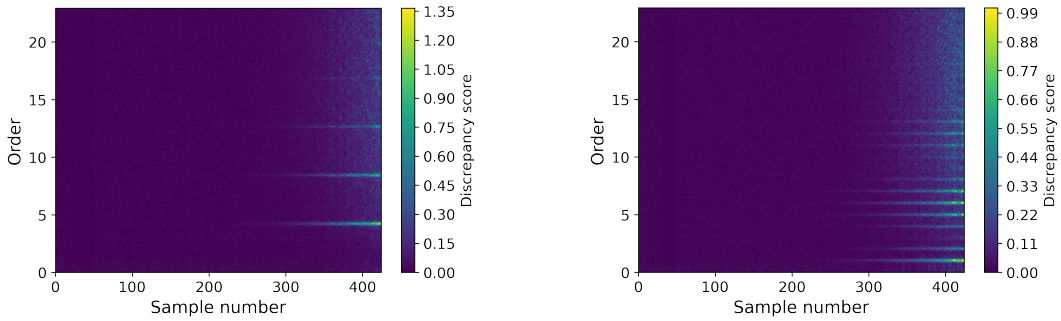
order of 5.88 along with its harmonics, but the peaks are surrounded by side bands, caused by the fault moving in and out of the load zone.



(a) Outer race fault

(b) Inner race fault

Figure 46: Reconstruction of samples with outer and inner race faults, along with the associated reconstruction space discrepancy signals (PCA-C-LR-L, SO-1-revolution, sample nr. 350, damage level 22)

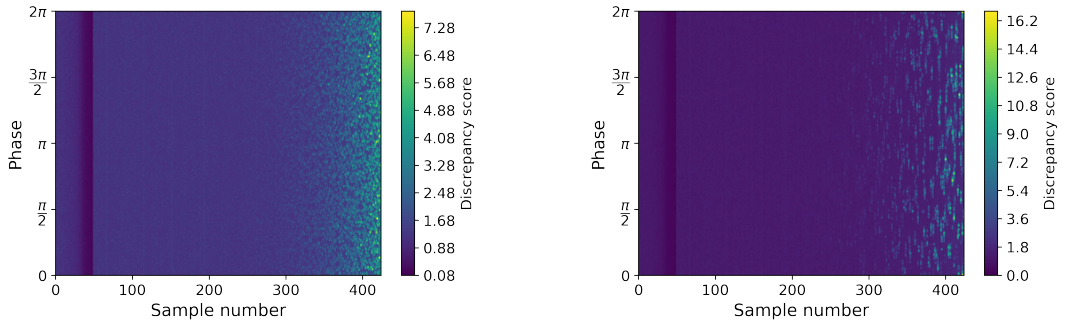


(a) Outer race fault

(b) Inner race fault

Figure 47: Order spectra of reconstruction space discrepancy signals for outer and inner race faults over time, as the fault develops (PCA-C-LR-L, SO-1-revolution)

Fig. 48 shows how the synchronous averages of the reconstruction space discrepancy signals for outer and inner race faults develop over time as the fault level increases. Because bearing faults are non-synchronous with the shaft rotation angle, one would expect to see no clear peaks developing on these plots, with only the noise levels increasing as the fault severity increases. These plots immediately indicate that there are no gear faults present. The very low discrepancy scores at samples 40 to 50 show that the model overfits slightly on the training data with faster operating speeds.



(a) Outer race fault

(b) Inner race fault

Figure 48: Synchronous averages of reconstruction space discrepancy signals for outer and inner race faults over time, as the fault develops (PCA-C-LR-L, SO-1-revolution)

The main advantage of the proposed methodology from a classification perspective is that the plots shown in Figs. 46, 47 and 48 should be representative of all outer and inner race bearing faults, for different datasets from different machines with different applications (as long as the model fits the healthy training data well). While the scales will depend on the fault frequencies, the differences in the structures between an inner and outer race bearing fault will remain the same.

The classification approaches in literature mainly focus on training models to classify the faults present in the data, taking either the raw vibration signal or features extracted from the raw data as input. These approaches are however limited to the specific machine from which the training data was acquired, and won't generalize well if data from another machine was fed into the model. By training classifiers on discrepancy signals, this problem can be overcome, allowing easier incorporation of transfer learning techniques from the larger available datasets which include labelled faults, into datasets where only healthy data is available. The reader is referred to the work of Guo et al. (2019) to see how transfer learning techniques are currently implemented in the vibration-based condition monitoring context. Training models on the discrepancy signals from large datasets, and then transferring the knowledge to smaller datasets, is an exciting prospect that can be explored in future work. For this to be possible, it is however very important that models are developed that can robustly capture the distribution of the healthy data, to generate clean discrepancy signals.

4.6 Conclusion

For this dataset, it is difficult to motivate the use of latent variable models. The C-decoder performed just as well as the latent variable models when the data was order tracked, highlighting the simplicity of the dataset, and showing that the shaft speed and phase information are sufficient to model the variance in the healthy data distribution to perform discrepancy analysis. The latent variable models struggled to capture the raw vibration signals in the unsupervised setting where no order tracking was applied. The latent variable models were not able to perform well without the shaft speed and phase information, showing that even on a relatively simple phenomenological model dataset, it is very challenging to get accurate health indicators for varying speed conditions when the shaft speed is not known.

Traditional signal processing approaches outperformed all learning-based approaches on this dataset. This is attributed to the simplicity of the dataset. The fault presents itself in a frequency band where there is little information related to other components, so if this band can be identified successfully and bandpass filtered, fault detection and tracking using traditional envelope techniques will be an easy task.

While the latent variable model results were not positive compared to the supervised C-decoder and traditional signal processing results, the investigation was very useful to identify why PCA and VAE models struggle in the unsupervised setting and to understand why incorporating OC information during data pre-processing and during modelling improves the models' performances.

For the PPCA model, the discrepancy signal features were correlated with the shaft speed, leading to poor results. The correlation is caused by the model's inability to capture the heteroscedastic noise in the data, showing the importance of using models that can capture the variance per sample. The PCA results were improved by using the PCA-Z-LR and PCA-C-LR models, where the variance in the data was inferred from $\boldsymbol{z}$ and $\boldsymbol{c}$ respectively, using linear regression. The PCA-C-LR model inferred the correct variance more robustly than the PCA-Z-LR model. This is because $\boldsymbol{c}$ is not influenced by noise or damage components present in the vibration data, while $\boldsymbol{z}$ captures some of these components as well. It was shown that the PCA models easily overfit (approach an identity function transformation) when given a large latent space. This makes it crucial that the latent space discrepancies are also monitored, since damage will not be detected so clearly in the reconstruction space. The PCA-C-LR model successfully monitored the latent space for discrepancies, by learning the relationship between $\boldsymbol{c}$ and $\boldsymbol{z}$ under healthy conditions. This made the PCA-C-LR model's performance more robust to the choice of latent space size. The PPCA and PCA-Z-LR models' performances were badly affected by overfitting, since the latent space representations could not be monitored using $\boldsymbol{c}$.

For the VAE models, the results were mixed. The models were able to get decent results when the input data windows were short (1-tooth pre-processing) and large latent space sizes were used. In the other cases, the VAE models struggled, with posterior collapse occurring often. It is clear that the VAE prior is not well suited to capture the vibration data with a small latent space. The VAE results were improved by using the iVAE model, where a more informed prior was learned by conditioning it on $\boldsymbol{c}$. Overall the results for the iVAE model were robust, only failing on the SO-1-revolution pre-processed data. The VAE models were seen to be less prone to overfitting (learning an identity function transformation) than the PCA models, because of the regularization that the latent prior applies.

All the models fit the healthy order tracked data significantly better than the raw data. This is easy to explain for the PCA models, since the frequency and phase shifts are highly nonlinear operations, and it was shown that the PCA models use certain principal components to reconstruct certain frequencies and their harmonics. For the nonlinear VAE models, this is attributed to the models not having any sort of inductive periodic bias, so the models simply try to map the data to the latent space and back, without learning any understanding of the frequency and phase shifts. The current neural network architectures are not well suited to capture periodic data, even when given the frequency and phase information. This was seen in the poor results of the C-decoder on the SO pre-processed data.

The data samples contain no impulsive components that are not associated with damage. This leads to good results when monitoring the kurtosis of the discrepancy signals. This is however not expected to be the case when working with real-world data, as will be investigated in the next chapter.

5 C-AIM Gearbox Dataset Analysis

In this chapter, a real-world dataset is analysed, to see if the findings on the phenomenological model dataset in Chapter 4 are also applicable to real-world data.

The dataset has been extensively analysed from both a signal processing perspective, where the goal is to extract and enhance the diagnostic information captured in the signals (Schmidt et al. 2020, Schmidt & Gryllias 2021), and a discrepancy analysis perspective, to detect and track how the fault develops (Schmidt et al. 2018*a*, Balshaw et al. 2022). This chapter builds on the work by Schmidt et al. (2018*a*) and Balshaw et al. (2022), as the dataset is also analysed from a discrepancy analysis perspective.

5.1 Dataset Overview

The C-AIM Gearbox dataset contains vibration data for an experimental gearbox setup operated at fluctuating speeds and loads. In the experiment, a gear tooth fault was induced on a healthy gear. The dataset consists of a healthy set recorded before the fault was seeded, as well as an unhealthy set after the fault was seeded.

The experimental setup (shown in Fig. 49) consists of an electrical motor, that is used to drive an alternator. The motor and alternator are connected through three helical gearboxes, with the gearbox in the middle being monitored. The axial acceleration of the monitored gearbox is captured with a 100 mV/g tri-axial accelerometer mounted on the input shaft bearing housing. The input shaft speed of the monitored gearbox is recorded with an optical probe and zebra tape shaft encoder. The data was captured with an Oros OR35 data acquisition system. The accelerometer signal was sampled at 25.6 kHz, and the optical probe at 51.2 kHz. The monitored gearbox is a step-up gearbox with a gear ratio of 1.85. The gear connected to the input shaft has 37 teeth, and the pinion connected to the output shaft has 20 teeth.

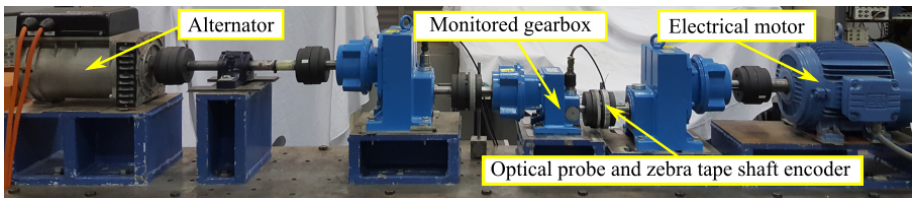


Figure 49: C-AIM Gearbox Experimental Setup

The healthy set consists of 100 samples of 20 seconds each, recorded in a relatively short period, to ensure that the gearbox condition remained relatively constant over the 100 samples. After the healthy data was recorded, the gearbox was disassembled, and one of the gear teeth was damaged. A slot was seeded into the root of a gear tooth, along the entire width of the tooth, 50% of the tooth thickness deep, with a height of 0.3mm. The gearbox was reassembled and run until complete failure occurred. After 20 days, the gear tooth failed. 200 samples of 20 seconds each were recorded before the tooth failed. This gives 300 samples in total for the dataset. The damaged tooth meshes with the pinion approximately $\frac{4\pi}{5}$ radians of shaft rotation after the butt joint on the zebra tape passes the optical probe. The butt joint is used as the reference point for all signals in this investigation, so the fault is expected to appear at $\frac{4\pi}{5}$ radians on synchronous average plots.

Each of the samples was recorded for the same speed profile over 20 seconds, with the speed profile of the input shaft of the monitored gearbox shown in Fig. 50. So the operating speed varies within the sample period, but the speed profile remains the same for all 300 samples.

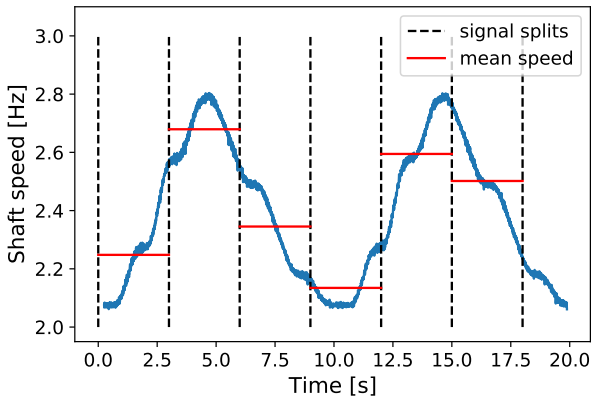


Figure 50: Speed profile of the input shaft of the monitored gearbox over 20 seconds. The sample splits are indicated by the dashed black lines, and the red lines indicate the mean shaft speeds over the shorter 3-second samples.

For this investigation, it was decided to split each 20-second sample into six shorter 3-second samples, with the last 2 seconds of each sample discarded. These splits are indicated in Fig. 50 by the dashed black lines, with the red lines indicating the mean shaft speeds over the shorter 3-second samples. The samples were split for two reasons:

- This allows one to easily detect if the discrepancy signal features are correlated with the shaft speed or not. If the whole 20 seconds were used, the discrepancy features would all be for signals from the same speed profile, so even if the discrepancy signal features varied greatly within the sample, the average would still be the same.
- By using shorter samples, the analysis cannot rely so heavily on synchronous averaging techniques to cancel out non-synchronous components, since each 3-second sample captures 8 shaft rotations at most, while the original samples captured up to 50 rotations. Since the monitored fault is a gear tooth fault, it makes sense to use synchronous averaging techniques, as was done by both Schmidt et al. (2018*a*) and Balshaw et al. (2022). By using shorter windows, the synchronous averaging results become more sensitive to any non-synchronous components in the healthy data that the model fails to capture. So using shorter samples will increase the difficulty, but also give a better indication of the models' abilities to capture the healthy data distribution.

By splitting the samples, the dataset used in this investigation has 600 healthy samples and 1200 unhealthy samples. The damage levels in this dataset are assigned based on the original signals before it was split, ranging from 0 to 200, where 0 corresponds to the healthy data, and each level from 1 to 200 corresponds to the original unhealthy signals. So the dataset with the split samples has 600 samples at damage level 0, 6 samples at damage level 1, 6 samples at damage level 2, and so on up to damage level 200.

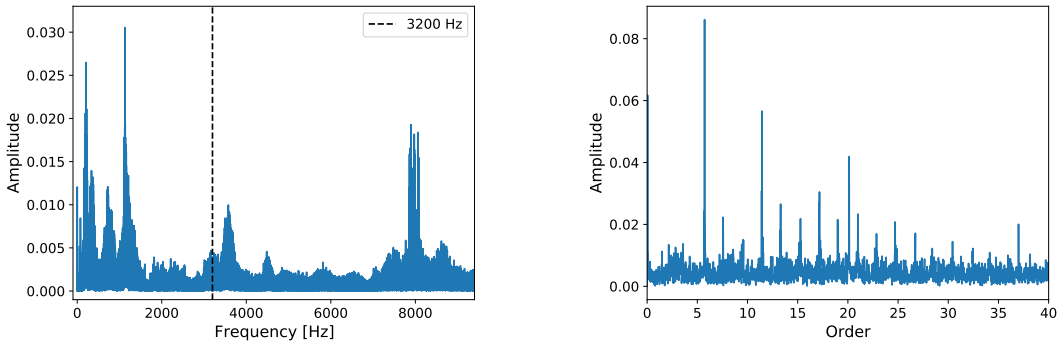
This dataset presents a challenge that was not encountered in the phenomenological model dataset analysis: the healthy and unhealthy vibration data both include impulsive components. In the phenomenological model dataset, the healthy data did not include impulsive components, so any impulsive components showing up in the data could immediately be identified as bearing damage.

In this dataset, the models must learn to capture these impulsive components in the healthy data, so that these components do not reflect in the discrepancy signals. The data contains a cyclo-stationary signal component at 5.72 shaft orders. Schmidt & Gryllias (2021) attribute this component to the movement of the floating shaft in the monitored gearbox. The input shaft experiences strong axial excitation due to the axial force of the helical gears. This results in undesired contact between the bearing and the casing of the gearbox, in both the healthy and unhealthy data.

Schmidt et al. (2020) showed that it is difficult to detect the gear damage with conventional signal analysis techniques, stating that this is due to the high noise levels in the dataset, and that the helical gears of the gearbox have higher contact ratios than spur gears, resulting in the gear mesh stiffness to only slightly decrease when a tooth starts to fail. It was shown that the signal components at 5.72 shaft orders negatively affect the results.

Balshaw (2020) analysed the raw dataset, as well as a version of the dataset that was low-pass filtered at 3200 Hz. This allows one to scale the difficulty of the fault detection task since low-pass filtering of the data at 3200 Hz removes most of the impulsive components at 5.72 shaft orders. The same is done in this analysis, to get an idea of how much a model's performance is influenced by the presence of impulsive components in the healthy data. A third-order Butterworth filter is used. It is also investigated how the results are affected by low-pass filtering of the discrepancy signals from the unfiltered dataset at 3200 Hz as a post-processing step, before enveloping is applied to the discrepancy signals.

Fig. 51(a) shows the frequency content of one of the healthy signals (of 20 seconds, before it was split). Two clear peaks can be seen in the healthy signal frequency content at 3700 Hz and 8000 Hz. To get an idea of what causes these peaks, the sample was high-pass filtered at 3200 Hz, Hilbert transformed, and order tracked, to extract the cyclostationary components from the data. The order spectrum for the Hilbert transformed frequency content above 3200 Hz is shown in Fig. 51(b). The component at 5.72 shaft orders can be clearly seen, along with its harmonics at 11.44 and 17.16 shaft orders. There is also a cyclostationary component at shaft order 20 included in these frequency bands. This is shown to give the reader an idea of what is filtered away when the filtered dataset is analysed. It will be shown later how these components at 5.72, 11.44, 17.16 and 20 shaft orders also affect the results in this investigation.



(a) (b)

Figure 51: Visualization of the frequency content of a healthy signal filtered away by the low-pass filter at 3200 Hz. (a) shows the FFT of the raw data. (b) shows the order spectrum for the Hilbert transformed frequency content above 3200 Hz.

5.2 Experimental setup

All 11 models, as shown in Table 3 were implemented. The dataset is quite large, and therefore the L_{sft} parameter was chosen to be equal to L_w. By doing this, the OTA and OT pre-processing approaches reduce to the exact same methods. So only 4 pre-processing approaches were implemented: OTA-1-revolution, OTA-1-tooth, SO-1-revolution and SO-1-tooth. The window lengths and latent space sizes are documented in Appendix C.

The Bayesian Geometry Compensation algorithm, proposed by Diamond et al. (2016) was implemented throughout this investigation to compensate for the geometrical imperfection caused by the butt joint of the zebra tape shaft encoder.

The features of the discrepancy signals that are monitored in this investigation are the mean, variance, kurtosis, OST at order 1 Hz/f_{shaft}, and SAT at $\frac{4\pi}{5}$ radians. For the unsupervised approach, the OC information will not be available that allows one to perform order tracking and calculate the OST or SAT features from the discrepancy signals. It is however still calculated and documented for the unsupervised approach, as it gives a good indication of a model's ability to capture the distribution of the healthy data. The only metric used to evaluate how the models perform on fault detection and tracking tasks is the True Positive Rate (TPR), where the discrepancy threshold is set as the 95^{th} percentile of the discrepancy signal features from the healthy test set. Since the damage was seeded after sample number 600, this dataset does not allow one to clearly track how the damage develops. A good model is expected to detect all samples after sample number 600 as unhealthy, resulting in a TPR of 1.

The same signal processing methods used to set benchmarks for the phenomenological model dataset investigation in Section 4.4.5 are used in this section. The SES and NES are monitored at order 1, and since it is a synchronous fault, the synchronous average of the squared envelope (SASE) is also monitored, at $\frac{4\pi}{5}$ radians, where the fault is known to be located. For the bandpass filtered dataset, the dataset is filtered around the 450-550 Hz frequency band. This frequency band was chosen based on the work done by Schmidt et al. (2020). In their work, it was shown that because of the cyclostationary impulsive components at 5.72 Hz, it is not a simple task to automatically identify the frequency band of interest that contains the gear damage, with both the fast kurtogram and ICS2gram wrongly identifying bands related to the 5.72 Hz cyclostationary component. The IFBI$_\alpha$gram was proposed, that successfully identified bands around 500 Hz as the frequency band of interest. For this benchmark, the robustness of techniques commonly used to identify the frequency band of interest, such as the fast kurtogram, is not considered. This benchmark assumes that the signal processing technique can successfully identify the 450-550 Hz frequency band for each signal.

5.3 Why do some latent variable models work better than others?

The goal of this section is to see if the discussion presented in Section 4.3, on the reasons that some models work better than others, also holds when used to analyse experimental data.

The performances are again compared based on whether homoscedastic models or heteroscedastic models are used, and thereafter the linear PCA and nonlinear VAE models are compared. In Section 4.3, when investigating the phenomenological model dataset, it was possible to scale the healthy test set likelihood values based on the operating speeds, since the amplitude modulation function was known. In this chapter, since it is experimental data, the amplitude modulation function is not known, so the likelihood values can't be scaled. This means that the likelihood values will be correlated with the operating speeds. For this reason, the quality of the fits is only investigated qualitatively

by looking at the discrepancy signals for the healthy and damaged data, and not quantitatively by comparing the likelihood values.

All the results presented from Section 5.3.1 to Section 5.3.3 are for the low-pass filtered dataset. Because the models fit the filtered dataset better, there is less noise in the discrepancy signals caused by healthy components, making it easier to understand how the different models fit the data. In Appendix F, all the results shown in Section 5.3.1 to Section 5.3.3 are presented again, but for the unfiltered dataset.

The subsection ends with a comparison between the order spectra of the discrepancy signals obtained by the models on the filtered and raw datasets, to show why filtering improves the results.

5.3.1 Homoscedastic vs. heteroscedastic models

Section 4.3.2 showed that, for the phenomenological dataset, using homoscedastic models (models that assume a constant variance across the whole dataset), results in the discrepancy signal features being correlated with the operating speed. To confirm that this is also the case for experimental data, the results for the PPCA and PCA-C-LR models are compared.

Fig. 52 shows the reconstruction space discrepancy features as the damage develops, for the PPCA-S and PCA-C-LR-S models when using OTA-1-tooth pre-processing. Note how the mean, variance and SAT features for the PPCA model are highly correlated with the shaft speed, with the features increasing as the RPM increases. For the heteroscedastic PCA-C-LR model, there is no clear correlation between the features and the shaft speed, resulting in more samples being classified as anomalous. The kurtosis and OST features perform poorly in both cases, showing little correlation with the damage level. Table 15 shows the TPRs achieved by monitoring the five reconstruction space discrepancy features. The TPRs for the mean, variance and SAT are improved by learning the variance for each sample, confirming the importance of using heteroscedastic models.

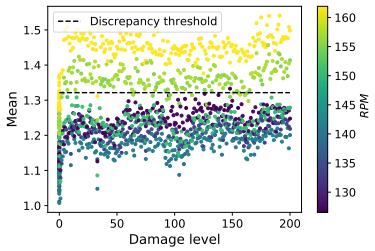
Table 15: PPCA-S vs. PCA-C-LR-S: TPR for reconstruction space discrepancy signal features (OTA-1-tooth, low-pass filtered dataset)

Model	**Discrepancy space**	**Feature**				
		mean	var	kurt	OST	SAT
PPCA-S	Recon	0.32	0.32	**0.01**	**0.40**	0.86
PCA-C-LR-S	Recon	**0.83**	**0.57**	**0.01**	**0.40**	**0.98**

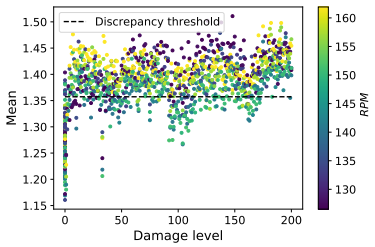
5.3.2 Linear PCA models

Section 4.3.3 showed that the linear PCA models learn to reconstruct the frequencies and their harmonics that are responsible for the most variance in the data. When adding more principal components, the models quickly start to overfit and can reconstruct almost any input data accurately. In these cases, the latent space was much more responsive to damage than the reconstruction space. To confirm that this is also the case for experimental data, we look at how the PCA-C-LR models performed.

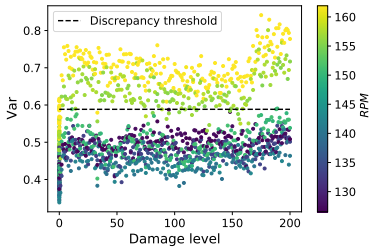
The results shown in this subsection are for OTA-1-tooth processed data. It is expected that the models will fit the order tracked data easier, and the shorter input lengths are chosen for this discussion so that the latent space discrepancy signals have a high resolution.



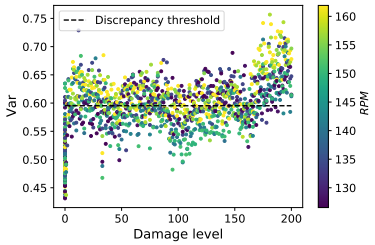
(a) PPCA: mean



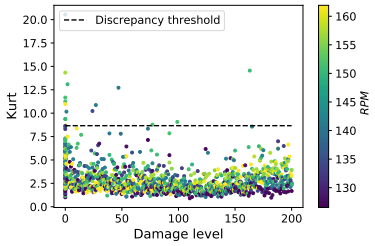
(b) PCA-C-LR: mean



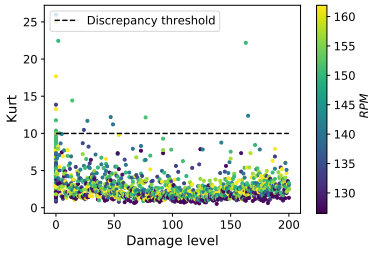
(c) PPCA: variance



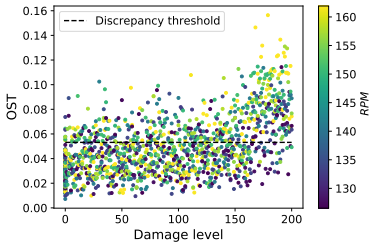
(d) PCA-C-LR: variance



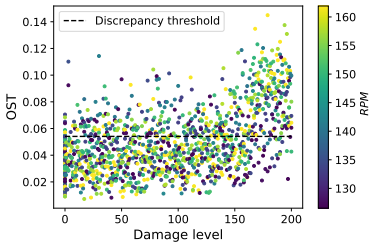
(e) PPCA: kurtosis



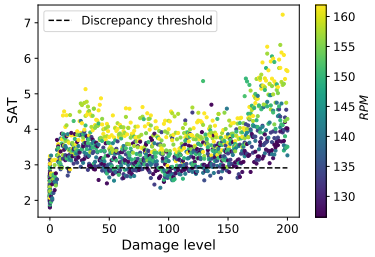
(f) PCA-C-LR: kurtosis



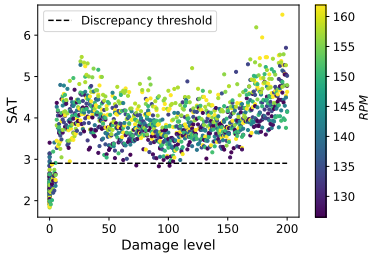
(g) PPCA: OST



(h) PCA-C-LR: OST



(i) PPCA: SAT

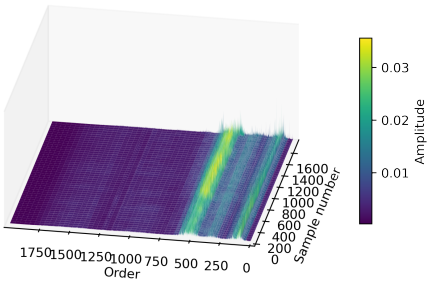


(j) PCA-C-LR: SAT

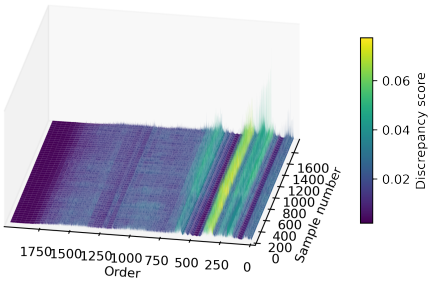
Figure 52: PPCA-S vs PCA-C-LR-S: reconstruction space discrepancy signal features for fault detection and tracking (OTA-1-tooth, low-pass filtered dataset)

To get an idea of how the models fit the data, we first look at the order spectra content of the vibration data, and then compare it to the reconstruction space discrepancy signals order spectra for both the PCA-C-LR-S and PCA-C-LR-L models, before the Hilbert transform is applied. These three plots are shown in Fig. 53. Note that for the vibration data, most of the order spectral content lies between

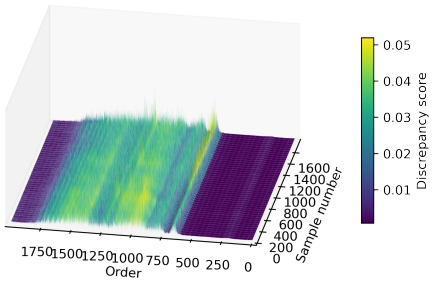
shaft orders 40 and 600, and then there are some very small components between orders 600 and orders 1800. The PCA-C-LR-S model captures some of the order spectra bands between shaft orders 40 and 600, while the PCA-C-LR-L model captures all the order spectra content below order 600. The PCA-C-LR-L model clearly overfits, as it not only captures the order spectral content below order 600 for the healthy data but also for the unhealthy data, showing that it will most likely reconstruct the signal components caused by damage just as well. The reconstruction space discrepancy signals for the PCA-C-LR-L model are made up of the very small components between orders 600 and orders 1800 that are barely seen in the vibration data order spectra. This confirms that the PCA models learn to capture certain frequencies, and can then reconstruct any signal content at that frequency very well. This makes it necessary to monitor the latent space for damage as well.



(a) Vibration data



(b) S: Recon space discrepancy



(c) L: Recon space discrepancy

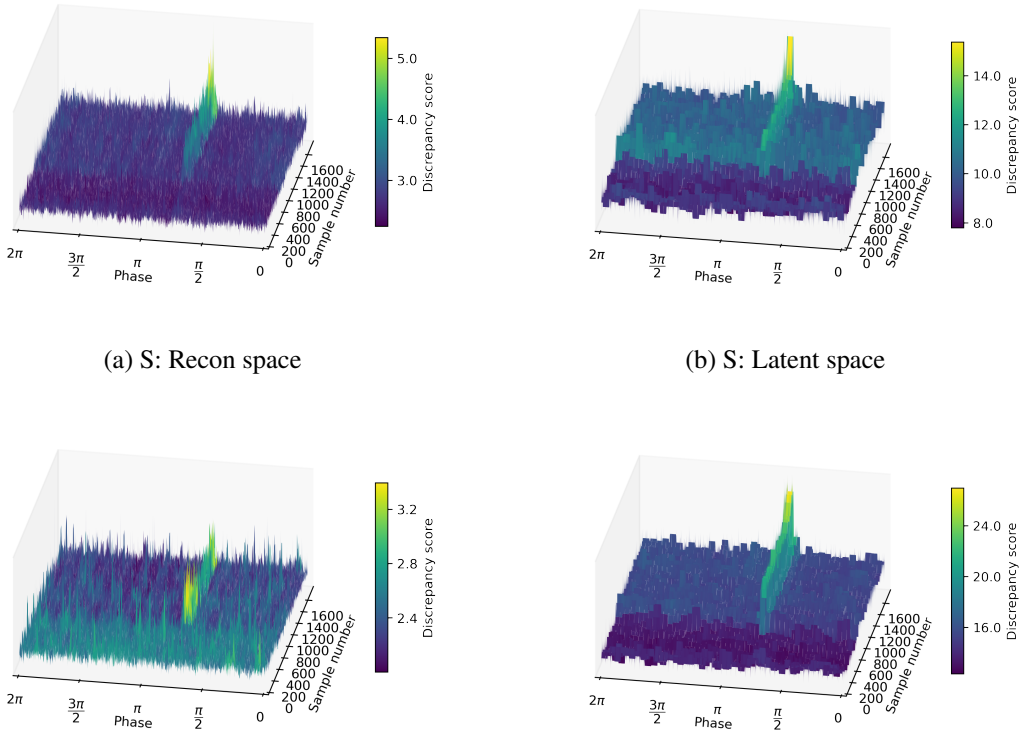
Figure 53: PCA-C-LR-S vs PCA-C-LR-L: Order spectra of reconstruction space discrepancy signals before Hilbert transform is applied, compared to the order spectra of the vibration data (OTA-1-tooth, low pass filtered dataset)

Fig. 54 shows the synchronous average of the reconstruction space and latent space discrepancy signals, as the fault develops, for both the PCA-C-LR-S and PCA-C-LR-L models. For the smaller latent space, the gear tooth fault can easily be identified in the reconstruction space, while the latent space synchronous average has a bit more noise. For the larger latent space, this is reversed, with the reconstruction space synchronous average being noisy, but the latent space being highly responsive to damage.

Looking at the discrepancy signal features monitored, these trends continue, with the small latent space model detecting damage in the reconstruction space, and the large latent space model detecting damage in the latent space. The discrepancy signal features for both the PCA-C-LR-S and PCA-C-LR-

L models, for both the latent space and reconstruction space, are plotted in Fig. 55. The TPR results are summarized in Table 16. The poor results from the reconstruction space discrepancy signal for the PCA-C-LR-L show that the model overfits and reconstructs the components associated with damage as well. It is interesting to note that the small latent space is also highly responsive to damage in this case, and for both the PCA-C-LR-S and PCA-C-LR-L models, the results improve considerably when the reconstruction space discrepancy monitoring results are combined with the latent space discrepancy monitoring results. The combined results for the PCA-C-LR-S and PCA-C-LR-L models are very similar. This shows that when both the latent space and reconstruction space are monitored, the results become less dependent on the latent space size decision.

The kurtosis and OST results are poor in both cases since these features are easily influenced by healthy impulsive components that are not captured by the model.



(a) S: Recon space

(b) S: Latent space

(c) L: Recon space

(d) L: Latent space

Figure 54: PCA-C-LR-S vs PCA-C-LR-L: Synchronous average of reconstruction- and latent space discrepancy signals, as the fault develops (OTA-1-tooth, low pass filtered dataset)

Table 16: TPR for PCA-C-LR model with small and large latent spaces (OTA-1-tooth)

Model	**Discrepancy space**	**Feature**				
		mean	var	kurt	OST	SAT
PCA-C-LR-S	Recon	0.83	0.57	0.01	0.40	0.98
	Latent	0.99	0.91	0.16	0.24	0.96
	Recon+Latent	**1.00**	**0.95**	**0.17**	**0.49**	**0.99**
PCA-C-LR-L	Recon	0.00	0.00	**0.06**	0.28	0.47
	Latent	**1.00**	0.15	0.00	0.41	**0.99**
	Recon+Latent	**1.00**	**0.16**	**0.06**	**0.58**	**0.99**

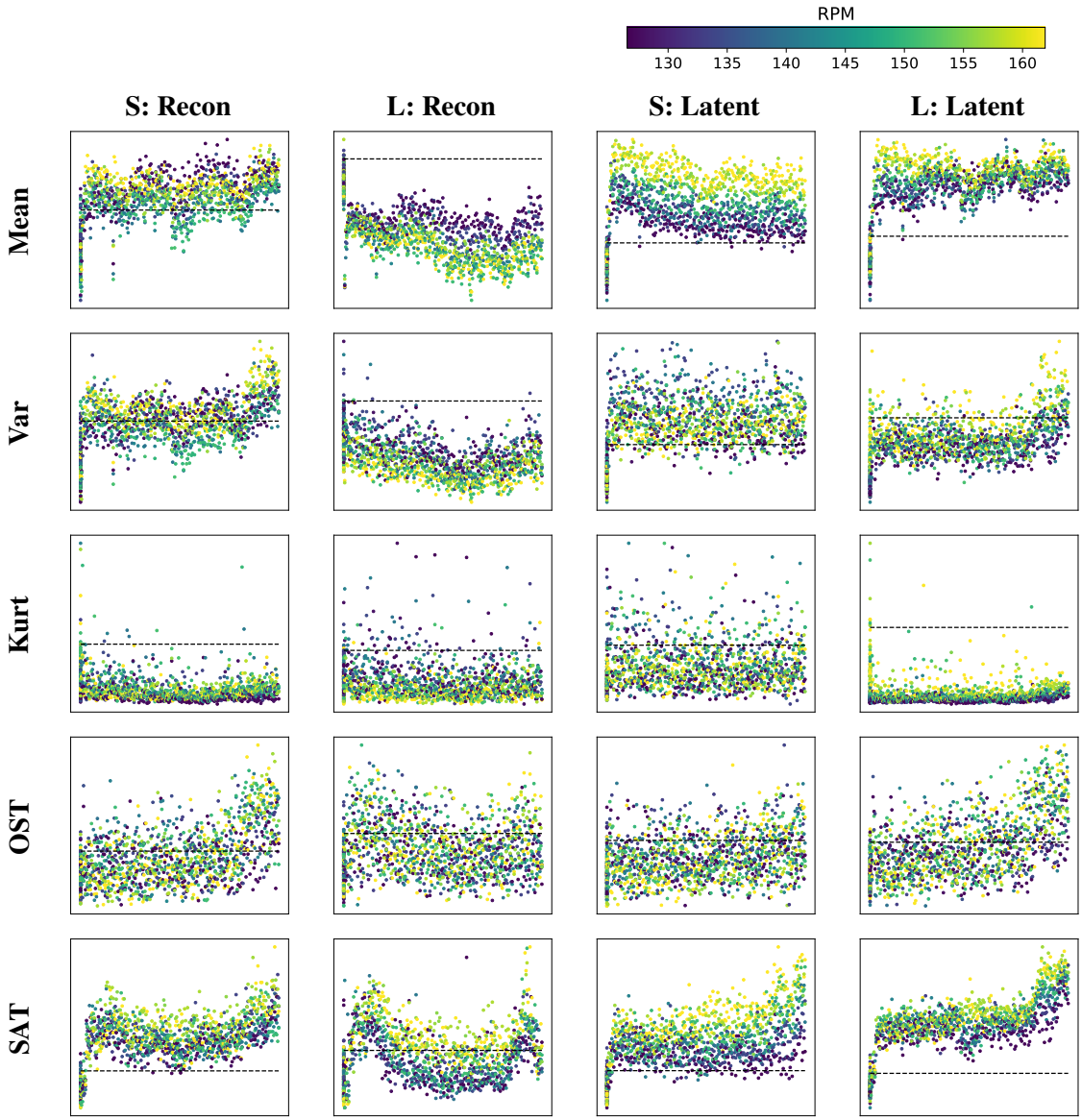


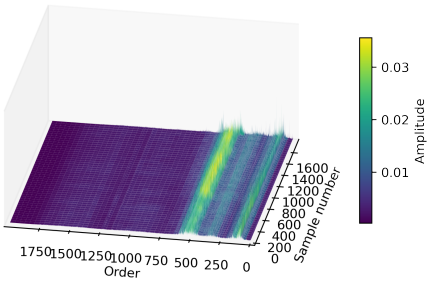
Figure 55: PCA-C-LR-S vs PCA-C-LR-L: reconstruction space and latent space discrepancy signal features for fault detection and tracking (OTA-1-tooth, low-pass filtered dataset, x-axis: Damage level 0-200, y-axis: Feature value, dashed line: Discrepancy threshold)

5.3.3 Nonlinear VAE models

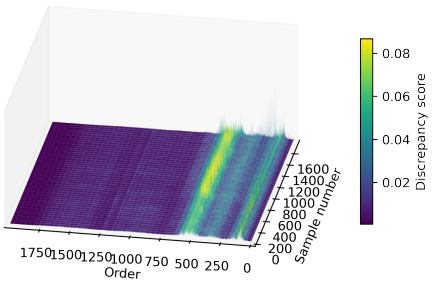
Section 4.3.4 showed that the nonlinear VAE models are less prone to overfitting than the PCA models when given a larger latent space. This made the latent space and reconstruction space evenly responsive to damage. To confirm that this is also the case for real-world data, we now look at how the iVAE models performed. This subsection contains the same results as the previous section on the PCA-C-LR models, only using iVAEs this time. The results are again for OTA-1-tooth processed data.

To get an idea of how the models fit the data, we again look at the order spectral content of the vibration data, and then compare it to the reconstruction space discrepancy signals order spectra for both the iVAE-S and iVAE-L models, before the Hilbert transform is applied. These three plots are shown in Fig. 53. From these plots, it can be seen that the nonlinear models fit the healthy data differently. By

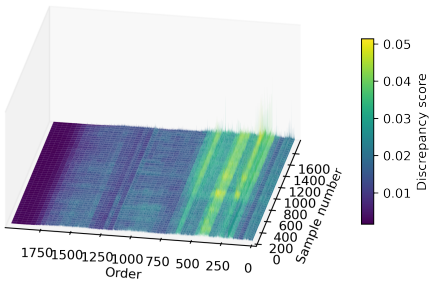
increasing the latent space size, the model can capture more of the order spectral content, but instead of reconstructing both the healthy and unhealthy signal components between shaft orders 40 and 600 perfectly, a change after sample 600 can now be detected in these bands. This shows that the iVAE model does not learn to simply reconstruct certain frequency content perfectly, making it less prone to overfit to the point where it can reconstruct any input data.



(a) Vibration data



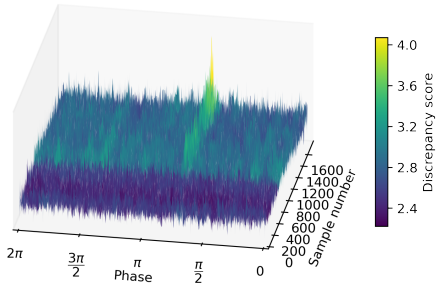
(b) S: Recon space



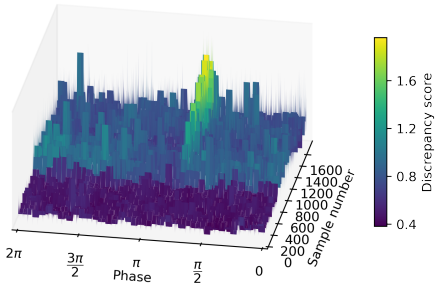
(c) L: Recon space

Figure 56: iVAE-S vs iVAE-L: Order spectra of reconstruction space discrepancy signals before Hilbert transform is applied, compared to order spectra of the vibration data (OTA-1-tooth, low-pass filtered dataset)

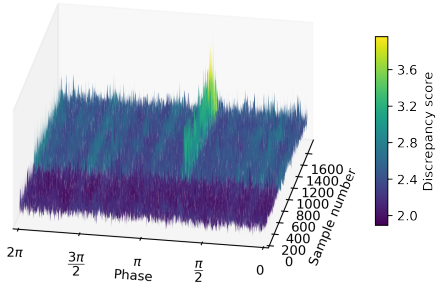
Fig. 57 shows the synchronous average of the reconstruction space and latent space discrepancy signals, as the fault develops, for both the iVAE-S and iVAE-L models. This time, the damage can be detected more clearly when the model is given a larger latent space, with the noise in both the reconstruction space and latent space discrepancy signal decreasing. This shows that, unlike the PCA models, the VAE models do not overfit (learn an identity function transformation) when given a larger latent space. This is because of the regularization that the latent space prior applies to the latent space. The VAE models simply improve on fitting the healthy data when given a larger latent space. The VAE models are therefore less prone to reconstruct the damage components as well.



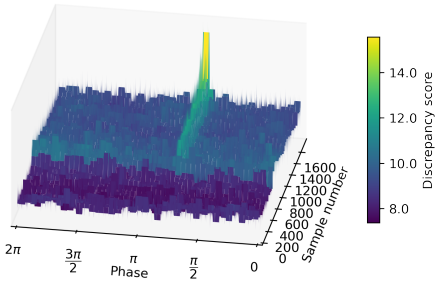
(a) S: Recon space



(b) S: Latent space



(c) L: Recon space



(d) L: Latent space

Figure 57: iVAE-S vs iVAE-L: Synchronous average of reconstruction- and latent space discrepancy signals, as the fault develops (OTA-1-tooth, low-pass filtered dataset)

While both models perform well when monitoring either the reconstruction space or latent space for damage, the larger latent space model can move the unhealthy samples slightly further above the discrepancy threshold, as can be seen in Fig. 58, which will make damage detection in the iVAE-L model slightly more robust. This is however a minor improvement from the iVAE-S model's performance, with both achieving near-perfect TPRs from the monitored discrepancy signals' mean, variance and SAT values, as shown in Table 17.

The kurtosis and OST results are once again poor in both cases, showing that these features are highly sensitive to the healthy signal components not captured by the models.

Table 17: TPR for iVAE model with small and large latent spaces (OTA-1-tooth)

Model	**Discrepancy space**	**Feature**				
		mean	var	kurt	OST	SAT
iVAE-S	Recon	0.99	0.99	**0.10**	0.29	0.98
	Latent	0.99	0.99	0.01	0.55	0.98
	Recon+Latent	**1.00**	**1.00**	**0.10**	**0.68**	**0.99**
iVAE-L	Recon	0.99	0.98	0.01	0.45	0.97
	Latent	0.97	0.96	**0.20**	0.38	0.98
	Recon+Latent	**1.00**	**1.00**	**0.20**	**0.60**	**0.99**

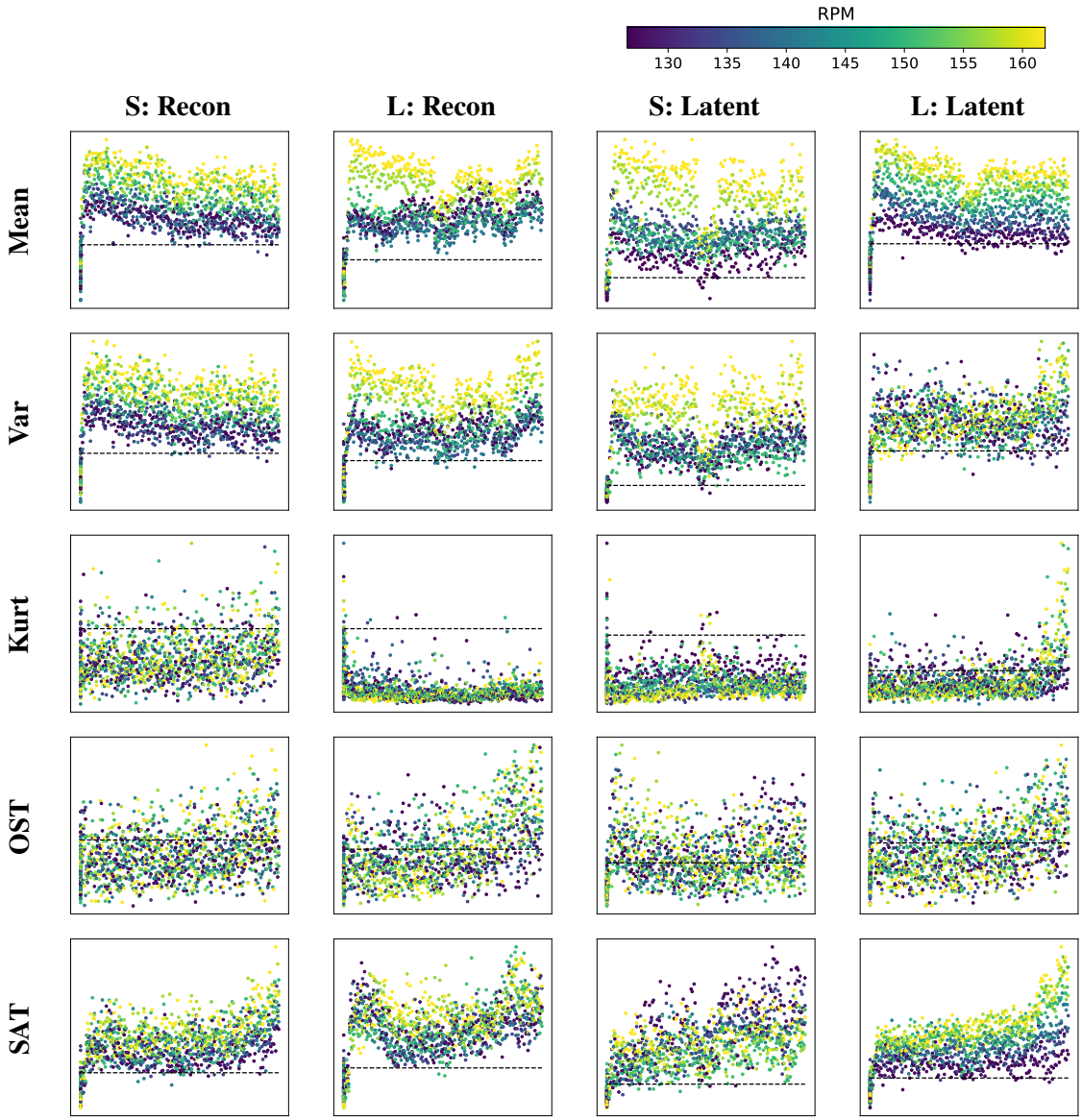
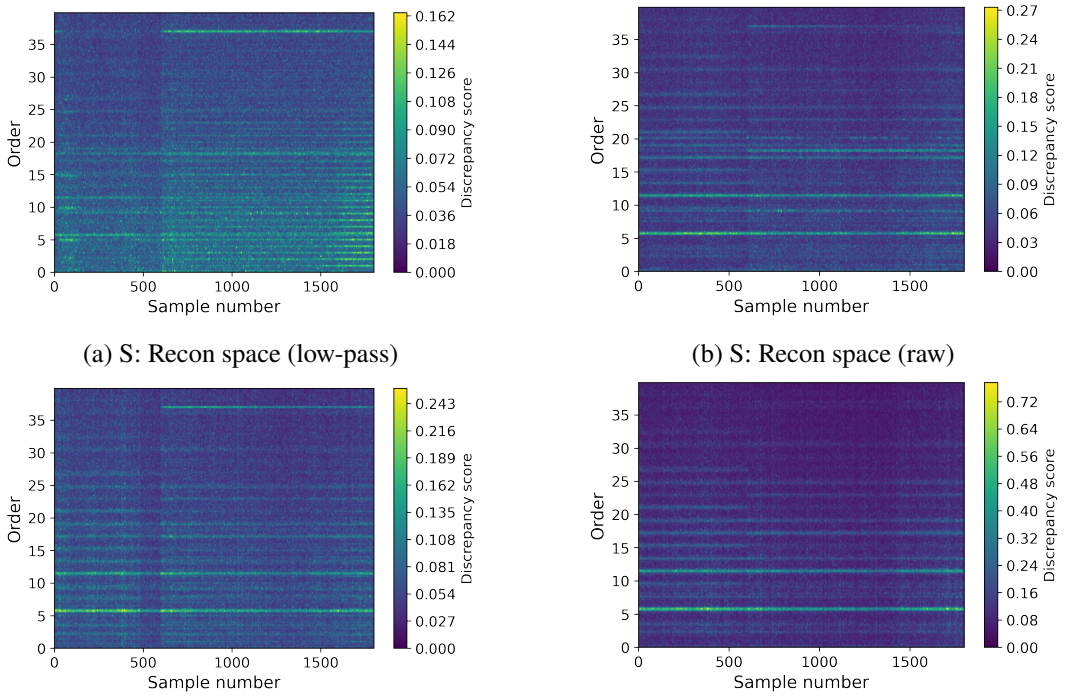


Figure 58: iVAE-S vs iVAE-L: reconstruction space and latent space discrepancy signal features for fault detection and tracking (OTA-1-tooth, low-pass filtered dataset, x-axis: Damage level 0-200, y-axis: Feature value, dashed line: Discrepancy threshold)

5.3.4 Performance on raw data vs. low-pass filtered data

The results on the raw data and the low-pass filtered data differed quite a lot, indicating that the models struggled to capture some of the impulsive components at shaft orders 5.72, 11.44, 17.16 and 20. In this subsection, the order spectra of the discrepancy signals obtained by the PCA-C-LR and iVAE models on the OTA-1-tooth pre-processed data, are investigated, for both the filtered and raw datasets. The goal is to show that some models do not capture the impulsive components and that this directly influences the TPR results that will be shown in Section 5.4. The ideal case would be that one detects little to no order spectral content before sample 600, and that one then clearly sees the lines at shaft order 1 and its harmonics, after sample 600. Any lines on the order spectra present from sample 1 to 1800, will indicate a healthy component not captured by the model.

The order spectra of the reconstruction space discrepancy signals obtained by the PCA-C-LR-S and PCA-C-LR-L models are shown in Fig. 59. On the filtered dataset, good results were obtained with the reconstruction space discrepancy signal obtained by the PCA-C-LR-S model (Fig. 59(a)). This is evident from the plot, as the fault orders at shaft order 1 and its harmonics are clearly visible, and not too many uncaptured healthy components are present. For the raw dataset (Fig. 59(b)), the order spectra are dominated by uncaptured healthy components at shaft order 5.72 and 11.44, with the fault orders at shaft order 1 and its harmonics barely visible. These uncaptured healthy components make it difficult to detect the damage from the monitored features, leading to poor results. For the PCA-C-LR-L model, the model overfits and reconstructs the damaged components well, for both the filtered data (Fig. 59(c)) and raw data (Fig. 59(d)). The frequency components that the model doesn't capture, are mostly healthy cyclostationary components at order 5.72 and 11.44. This makes it necessary to detect the damage in the latent space discrepancy signals.



(a) S: Recon space (low-pass)

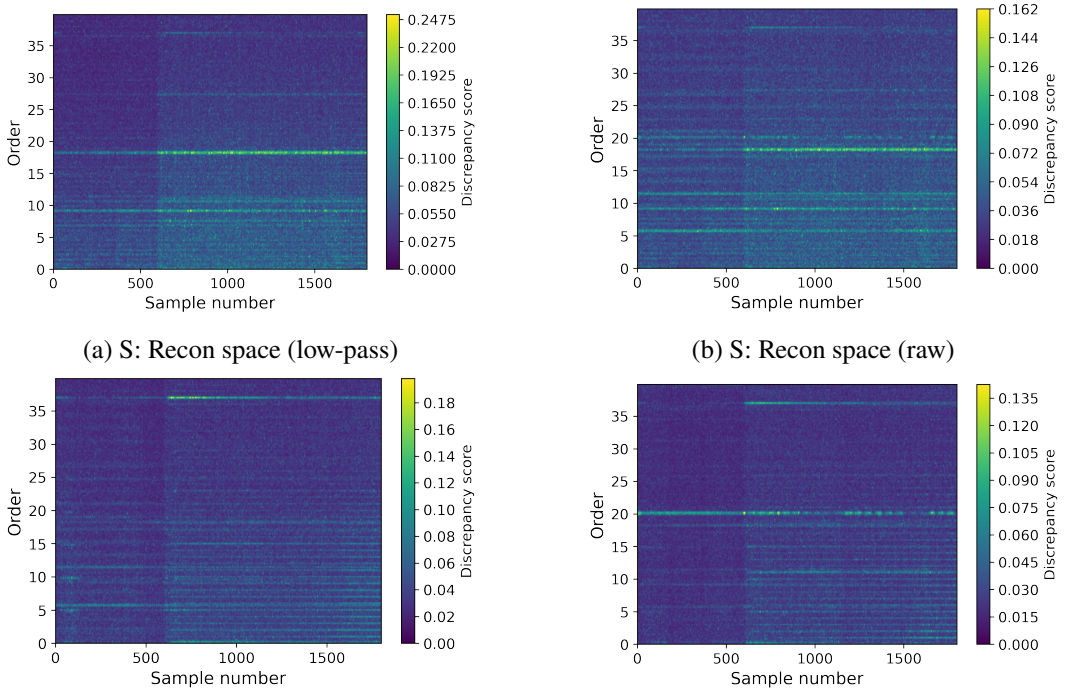
(b) S: Recon space (raw)

(c) L: Recon space (low-pass)

(d) L: Recon space (raw)

Figure 59: PCA-C-LR-S vs PCA-C-LR-L: Order spectra of reconstruction space discrepancy signals, as the fault develops (OTA-1-tooth, raw and low-pass filtered dataset)

The order spectra of the reconstruction space discrepancy signals obtained by the iVAE-S and iVAE-L models are shown in Fig. 60. Comparing Fig. 60(a) and Fig. 60(c), it can be seen why the damage was detected more clearly in the reconstruction space discrepancy signal for the iVAE-L model than the iVAE-S model, as was documented in Section 5.3.3. For the iVAE-S model, there are uncaptured healthy components at orders 9 and 18, while the iVAE-L model managed to successfully capture these components. Now looking at the results on the raw dataset, it can be seen that the iVAE-S model failed to capture components at shaft orders 5.72, 9, 11.44, 18 and 20 (Fig. 60(b)), making the fault orders at shaft order 1 and its harmonics barely visible. The iVAE-L model performs better, with the only uncaptured component being at order 20 (Fig. 60(d)). The model successfully captured the large impulsive components at orders 5.72 and 11.44, and the fault orders at shaft order 1 and its harmonics can be clearly seen.



(a) S: Recon space (low-pass)

(b) S: Recon space (raw)

(c) L: Recon space (low-pass)

(d) L: Recon space (raw)

Figure 60: iVAE-S vs iVAE-L: Order spectra of reconstruction space discrepancy signals, as the fault develops (OTA-1-tooth, raw and low pass filtered dataset)

This discussion shows that while the PCA-C-LR models performed competitively with the iVAE models on the phenomenological model dataset, the iVAE models outperform the PCA-C-LR models when the dataset contains many impulsive components.

5.4 Result Analysis

5.4.1 Unsupervised approach

Table 18: C-AIM gearbox dataset analysis results: TPR for unsupervised approach, where no OC information was used (green: >0.8, red: <0.5)

SO-1-revolution																
Model	**Discrepancy space**	**No filtering**					**Filtered discrepancy signals (after training)**					**Filtered vibration signals (before training)**				
		mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT
PPCA-S	Recon	0.15	0.04	0.01	0.32	0.08	0.39	0.48	0.06	0.27	0.65	0.42	0.50	0.12	0.27	0.67
PPCA-L	Recon	0.05	0.02	0.01	0.14	0.05	0.31	0.32	0.00	0.31	0.83	0.37	0.35	0.01	0.38	0.80
PCA-Z-LR-S	Recon	0.31	0.07	0.00	0.23	0.08	0.50	0.51	0.01	0.22	0.72	0.44	0.29	0.09	0.20	0.58
PCA-Z-LR-L	Recon	0.18	0.18	0.03	0.09	0.22	0.23	0.19	0.03	0.11	0.20	0.38	0.34	0.04	0.15	0.34
VAE-S	Recon	0.11	0.04	0.01	0.24	0.07	0.36	0.47	0.13	0.13	0.62	0.36	0.47	0.17	0.14	0.66
VAE-L	Recon	0.11	0.03	0.01	0.22	0.07	0.36	0.47	0.12	0.13	0.62	0.36	0.48	0.16	0.14	0.66
SO-1-tooth																
Model	**Discrepancy space**	**No filtering**					**Filtered discrepancy signals (after training)**					**Filtered vibration signals (before training)**				
		mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT
PPCA-S	Recon	0.01	0.01	0.01	0.15	0.07	0.12	0.16	0.04	0.14	0.64	0.36	0.32	0.01	0.23	0.74
PPCA-L	Recon	0.02	0.02	0.01	0.04	0.03	0.06	0.03	0.01	0.03	0.22	0.01	0.02	0.06	0.17	0.43
PCA-Z-LR-S	Recon	0.00	0.01	0.01	0.09	0.07	0.05	0.10	0.04	0.10	0.68	0.44	0.32	0.01	0.15	0.71
PCA-Z-LR-L	Recon	0.05	0.06	0.06	0.04	0.06	0.15	0.23	0.03	0.08	0.11	0.00	0.01	0.03	0.06	0.34
VAE-S	Recon	0.11	0.03	0.01	0.24	0.17	0.38	0.47	0.08	0.12	0.65	0.36	0.48	0.17	0.14	0.65
VAE-L	Recon	0.09	0.06	0.08	0.16	0.64	0.54	0.51	0.15	0.04	0.71	0.36	0.48	0.16	0.14	0.65

The TPRs for the model and pre-processing combinations where no OC information was used during modelling or pre-processing are presented in Table 18. The results are interpreted as follows:

- None of the models were able to capture the unfiltered dataset.
- The results were slightly improved by filtering the input data, and by filtering the discrepancy signals. This shows that the models struggle to capture the high-frequency impulsive components.
- The only reasonably good results were from monitoring the SAT, since the non-synchronous components that the models did not capture, could then be averaged out slightly. To compute the SAT, one however needs to order track the discrepancy signals, making the process not completely unsupervised anymore.
- The healthy data distribution is simply too complex when the data is not order tracked, and the limitations of the PCA and VAE frameworks further worsen the results. This shows that PCA and VAE models cannot be used for discrepancy analysis in an unsupervised framework on real-world data.

5.4.2 Using OC information for pre-processing

Table 19: C-AIM gearbox dataset analysis results: TPR where available OC information was used only during pre-processing (green: >0.8, red: <0.5)

OTA-1-revolution																
Model	**Discrepancy space**	**No filtering**					**Filtered discrepancy signals (after training)**					**Filtered vibration signals (before training)**				
		mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT
PPCA-S	Recon	0.36	0.06	0.01	0.26	0.11	0.57	0.66	0.19	0.16	0.88	0.60	0.69	0.10	0.19	0.91
PPCA-L	Recon	0.16	0.03	0.01	0.24	0.07	0.63	0.57	0.01	0.53	0.98	0.92	0.64	0.00	0.51	0.98
PCA-Z-LR-S	Recon	0.65	0.53	0.01	0.36	0.29	0.77	0.78	0.11	0.28	0.89	0.74	0.67	0.20	0.32	0.66
PCA-Z-LR-L	Recon	0.31	0.29	0.04	0.29	0.30	0.31	0.29	0.04	0.28	0.30	0.91	0.91	0.00	0.90	0.94
VAE-S	Recon	0.35	0.25	0.04	0.35	0.28	0.48	0.53	0.48	0.23	0.64	0.46	0.53	0.38	0.21	0.68
VAE-L	Recon	0.17	0.08	0.02	0.27	0.20	0.42	0.49	0.33	0.14	0.62	0.39	0.49	0.27	0.16	0.66
OTA-1-tooth																
Model	**Discrepancy space**	**No filtering**					**Filtered discrepancy signals (after training)**					**Filtered vibration signals (before training)**				
		mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT
PPCA-S	Recon	0.05	0.02	0.02	0.32	0.07	0.88	0.89	0.19	0.28	0.93	0.32	0.32	0.01	0.40	0.86
PPCA-L	Recon	0.01	0.01	0.03	0.05	0.05	0.01	0.01	0.03	0.06	0.05	0.00	0.00	0.07	0.22	0.50
PCA-Z-LR-S	Recon	0.19	0.04	0.02	0.29	0.08	0.97	0.98	0.08	0.29	0.94	0.56	0.43	0.01	0.34	0.96
PCA-Z-LR-L	Recon	0.18	0.03	0.04	0.07	0.22	0.24	0.22	0.04	0.09	0.18	0.00	0.00	0.04	0.13	0.54
VAE-S	Recon	0.30	0.14	0.01	0.31	0.15	0.49	0.55	0.15	0.17	0.67	0.48	0.54	0.24	0.20	0.73
VAE-L	Recon	0.57	0.64	0.04	0.44	0.88	0.99	0.99	0.12	0.48	0.98	0.99	0.98	0.00	0.49	0.98

The TPRs for the model and pre-processing combinations where the OC information was used only during pre-processing are presented in Table 19. The results are interpreted as follows:

- The only model that manages to capture the unfiltered data to a reasonable degree is the VAE-L model when OTA-1-tooth pre-processing is applied. The latent space is large enough to capture the short input windows. With the monitored SAT, the non-synchronous components get averaged out, leading to good results
- The PCA-Z-LR-L model performs poorly on the unfiltered OTA-1-revolution input signals, even after the discrepancy signals are filtered. The PCA-Z-LR-S, PPCA-S and PPCA-L models perform well when the discrepancy signals are filtered. This shows that while the models were not able to capture the high-frequency impulsive components, the PCA-Z-LR-S, PPCA-S

and PPCA-L models were still able to get reasonable estimates of the variance in the data. PCA-Z-LR-L however breaks down, with the linear regression model failing to capture the variance, since the high-dimensional latent representation captured too much noise. When the dataset is filtered before training, the PCA-Z-LR-L model performs well, showing that the linear regression model can learn the variance from the latent space representations if they are not too noisy.

- For the shorter OTA-1-tooth input signal, the large latent space PCA models overfit and reconstruct the damaged signal components, and since the latent space can't be monitored, the results are poor. The small latent space PCA models perform better, since the models do not overfit.
- The VAE models struggle to capture the longer OTA-1-revolution input signals, even when filtering is applied. The latent representations are too noisy, leading to posterior collapse. The models perform better on the shorter OTA-1-tooth input data, with the VAE-L model performing very well when the input data or discrepancy signals are filtered. This shows that the VAE-L model only struggled to capture some of the high-frequency impulsive components, but captured the rest of the signal components very well.
- Overall, a clear improvement can be seen from the completely unsupervised approach.

5.4.3 Using OC information for modelling

Table 20: C-AIM gearbox dataset analysis results: TPR where available OC information was used only during modelling (green: >0.8, red: <0.5)

SO-1-revolution																
Model	**Discrepancy space**	**No filtering**					**Filtered discrepancy signals (after training)**					**Filtered vibration signals (before training)**				
		mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT
C-decoder	Recon	0.18	0.03	0.01	0.29	0.04	0.65	0.78	0.10	0.16	0.86	0.70	0.84	0.28	0.30	0.84
PCA-C-LR-S	Recon+LS	0.32	0.06	0.03	0.32	0.05	0.76	0.91	0.14	0.31	0.85	0.85	0.93	0.28	0.46	0.91
PCA-C-LR-L	Recon+LS	0.30	0.07	0.03	0.19	0.31	0.62	0.50	0.03	0.34	0.93	0.95	0.74	0.06	0.43	0.98
iVAE-S	Recon+LS	0.15	0.10	0.04	0.25	0.12	0.39	0.49	0.16	0.16	0.63	0.59	0.56	0.20	0.36	0.74
iVAE-L	Recon+LS	0.18	0.11	0.05	0.26	0.18	0.38	0.49	0.16	0.17	0.62	0.37	0.50	0.19	0.24	0.66
SO-1-tooth																
Model	**Discrepancy space**	**No filtering**					**Filtered discrepancy signals (after training)**					**Filtered vibration signals (before training)**				
		mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT
C-decoder	Recon	0.17	0.03	0.00	0.29	0.05	0.64	0.84	0.12	0.16	0.80	0.72	0.85	0.27	0.31	0.83
PCA-C-LR-S	Recon+LS	0.78	0.99	0.13	0.42	0.62	0.78	0.99	0.14	0.40	0.96	0.68	0.95	0.21	0.53	0.96
PCA-C-LR-L	Recon+LS	0.02	0.01	0.04	0.09	0.06	0.03	0.01	0.05	0.08	0.09	0.79	0.40	0.06	0.58	0.98
iVAE-S	Recon+LS	0.26	0.07	0.03	0.15	0.23	0.52	0.58	0.06	0.12	0.74	0.81	0.94	0.42	0.30	0.88
iVAE-L	Recon+LS	0.88	0.72	0.21	0.42	0.89	0.88	0.72	0.48	0.22	0.96	0.84	0.94	0.03	0.44	0.98

The TPRs for the model and pre-processing combinations where the OC information was used only during modelling are presented in Table 20. The results are interpreted as follows:

- The models that perform reasonably well on the unfiltered data are the PCA-C-LR-S and VAE-L models when SO-1-tooth pre-processing is applied. It is easier for the models to capture the short input signals that are not order-tracked, than the long input signals that are not order-tracked.
- It is interesting to note that when the discrepancy signals or input data are filtered for SO-1-revolution pre-processing, the linear PCA models perform better than the nonlinear models. This might be due to the PCA models' abilities to robustly fit certain frequencies, while the

iVAE models are still prone to posterior collapse when the latent space representations are too noisy.

- For the shorter SO-1-tooth processed input data, the models all perform generally well when the discrepancy signals or input data samples are filtered. The only odd case is for the PCA-C-LR-L model when the discrepancy signals are filtered. We believe that the reason for this is that the model learns to reconstruct data from certain operating speeds very well, while not capturing others. This makes it difficult for the linear regression model to accurately capture the variance of the reconstruction errors.
- The C-decoder performs fairly well when the discrepancy signals or input data are filtered, but is not able to reach TPRs higher than 0.9 as some of the PCA and VAE frameworks achieve. This again shows that it is not easy for the neural network architecture to learn a good understanding of how varying frequency and phase values translate to a time-series signal, even when the frequency and phase values are given to the network as inputs. The latent variable models perform better when they can encode the input data to more abstract representations that capture the phase and frequency, and then decode the data from there.
- Overall the results are better than for the unsupervised approach, as well as for the approach that uses the OC information only for pre-processing. This indicates how much value the available operating condition information adds to the PCA and VAE frameworks, overcoming the limitations of these frameworks, as discussed in Chapter 3.

5.4.4 Using OC information for pre-processing and modelling

Table 21: C-AIM gearbox dataset analysis results: TPR where available OC information was used during pre-processing and modelling (green: >0.8, red: <0.5)

OTA-1-revolution																
Model	**Discrepancy space**	**No filtering**					**Filtered discrepancy signals (after training)**					**Filtered vibration signals (before training)**				
		mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT
C-decoder	Recon	0.83	0.12	0.00	0.31	0.13	0.99	1.00	0.14	0.27	0.99	1.00	1.00	0.09	0.36	0.99
PCA-C-LR-S	Recon+LS	0.81	0.70	0.04	0.86	0.59	1.00	1.00	0.23	0.81	0.99	1.00	1.00	0.14	0.82	0.99
PCA-C-LR-L	Recon+LS	0.79	0.14	0.07	0.37	0.80	1.00	0.99	0.07	0.63	0.99	1.00	1.00	0.03	0.60	1.00
iVAE-S	Recon+LS	0.74	0.65	0.02	0.71	0.74	0.74	0.66	0.35	0.68	0.77	0.84	0.82	0.46	0.83	0.86
iVAE-L	Recon+LS	0.75	0.75	0.03	0.80	0.75	0.75	0.75	0.42	0.78	0.78	0.81	0.80	0.34	0.83	0.84
OTA-1-tooth																
Model	**Discrepancy space**	**No filtering**					**Filtered discrepancy signals (after training)**					**Filtered vibration signals (before training)**				
		mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT	mean	var	kurt	OST	SAT
C-decoder	Recon	0.81	0.11	0.00	0.20	0.12	1.00	1.00	0.16	0.23	0.97	1.00	1.00	0.09	0.35	0.98
PCA-C-LR-S	Recon+LS	0.87	0.74	0.17	0.44	0.71	1.00	1.00	0.25	0.40	0.99	1.00	0.95	0.17	0.49	0.99
PCA-C-LR-L	Recon+LS	0.07	0.03	0.10	0.11	0.10	0.24	0.05	0.10	0.14	0.12	1.00	0.16	0.06	0.58	0.99
iVAE-S	Recon+LS	0.83	0.44	0.02	0.28	0.61	0.98	0.99	0.05	0.28	0.97	1.00	1.00	0.10	0.68	0.99
iVAE-L	Recon+LS	1.00	0.90	0.10	0.74	0.99	1.00	1.00	0.30	0.67	1.00	1.00	1.00	0.20	0.60	0.99

The TPRs for the model and pre-processing combinations where the OC information was used for both pre-processing and modelling are presented in Table 21. The results are interpreted as follows:

- There is a clear improvement in performance on the unfiltered dataset, with the iVAE-L on the OTA-1-tooth processed data achieving near-perfect results. Overall the results are far better than the other approaches.
- From the SAT results for the C-decoder model on the unfiltered data, it can be seen that the C-decoder struggles more than the iVAE and PCA models to capture the data. This highlights

the advantage of using latent variable models. The phase information of the non-synchronous components at 5.72 shaft orders is not known, and therefore not fed into the C-decoder model. The model can thus never learn to reconstruct the components at 5.72 shaft orders accurately. With the latent variable models, it is possible to capture the components at 5.72 shaft orders, since the models can learn to capture a representation of the phase of these components in the latent space.

- When the discrepancy signals or input data samples are filtered, most of the models perform very well, especially on the OTA-1-tooth processed input data.
- Monitoring the kurtosis of the discrepancy signals did not give any positive results. Monitoring the OST gave some positive results, but overall the mean, variance and SAT gave the best results.

5.4.5 Signal processing benchmark

The TPRs obtained by monitoring the SES, NES and SASE for the unfiltered and bandpass-filtered datasets are documented in Table 22. When the dataset is not filtered, the SES, NES and SASE all return poor results. Bandpass filtering the dataset improves the results, with the SASE performing the best with a TPR of 0.98. The SES and NES still perform relatively poorly with TPRs below 0.8. Because the dataset is split into the shorter 3-second windows, the envelope spectra are not as refined as would be the case for the original 20-second windows, making it very difficult to detect damage by tracking the envelope spectra.

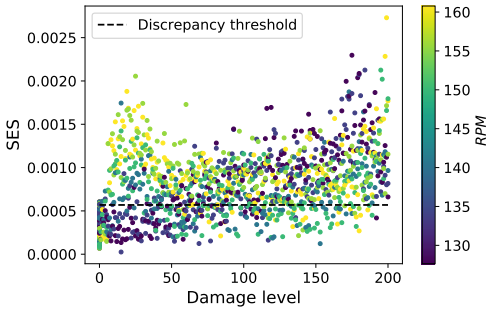
Table 22: C-AIM gearbox dataset signal processing benchmark (TPR metric)

	SES @ order 1	NES @ order 1	SASE @ $\frac{4\pi}{5}$ rad
No filter	0.10	0.06	0.02
Bandpass filtered (450-550 Hz)	0.74	0.55	0.98

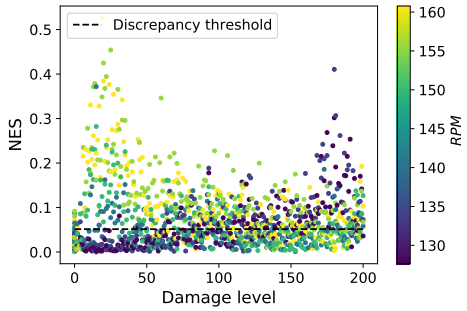
The SES and NES at order 1, and the SASE at $\frac{4\pi}{5}$ radians for the bandpass filtered dataset are plotted in Fig. 61, as the damage level increases.

The semi-supervised learning-based approaches show clear potential to perform just as well as the signal processing approaches, without the requirement of having to successfully identify the frequency band where the fault presents itself. By not having to identify a frequency band of interest, and then filtering the content outside the band away, the learning-based methods can potentially also consider diagnostic information outside the main frequency band where the damage presents itself. For the unfiltered dataset, the iVAE-L model with OTA-1-tooth pre-processing performed similarly to the signal processing approach, returning a TPR of 0.74 for OST compared to the SES also returning a TPR of 0.74. The SAT returned a TPR of 0.99, compared to the SASE of 0.98. These results are very promising, considering that traditional signal processing approaches struggle to automatically identify the frequency bands where the fault presents itself in this dataset (Schmidt et al. 2020).

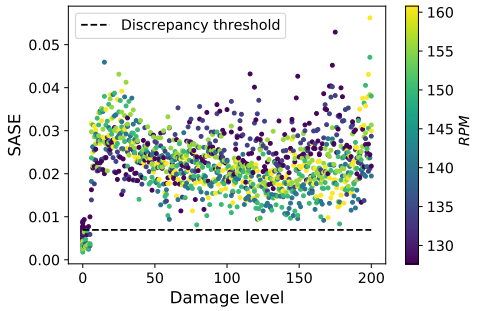
Most of the learning-based approaches struggled to successfully capture the cyclostationary components at 5.72 Hz. When most of this cyclostationary component is filtered away by low-pass filtering the data at 3200 Hz, all of the semi-supervised approaches that use OC information for pre-processing and modelling return results that perform just as well or better than the signal processing approaches.



(a) SES @ order 1



(b) NES @ order 1



(c) SASE @ $\frac{4\pi}{5}$ rad

Figure 61: The SES and NES at order 1 and the SASE at $\frac{4\pi}{5}$ radians for the bandpass filtered dataset over time as the damage level increases.

5.5 Fault classification: local gear tooth fault

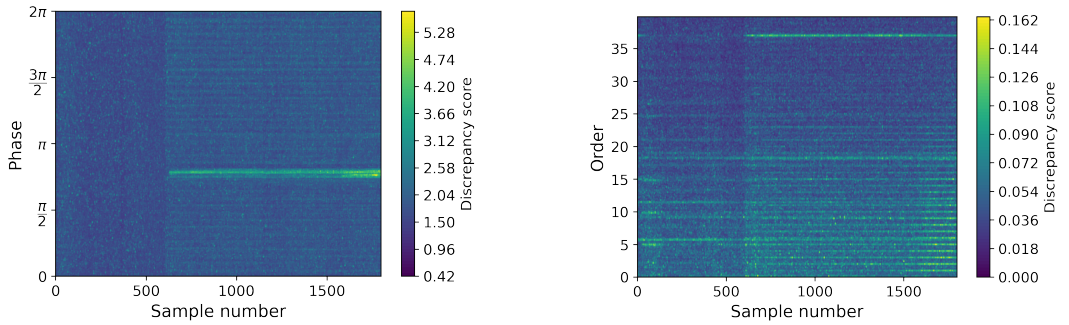
In Section 4.5, the differences in the discrepancy signal order spectra and synchronous average plots for bearing inner and outer race faults were shown. Bearing faults have certain characteristics that can be identified in these plots, irrespective of the machine which the data was collected from. In this subsection, we look at the discrepancy signal order spectra and synchronous average plots for the local gear tooth fault, to see how these plots differ from the ones associated with bearing faults.

The synchronous average and order spectra plots of the reconstruction space discrepancy signal, as it develops over the 1800 samples, are shown in Fig. 62. The plots are from the low-pass filtered dataset, using the PCA-C-LR-S model, with OTA-1-tooth pre-processing. Results from the filtered dataset are shown, as the models were able to capture the healthy distribution from this dataset better than for the unfiltered dataset, resulting in cleaner plots since there is less noise in the discrepancy signals caused by healthy components.

Looking at the synchronous average plot in Fig. 62(a), one can clearly see the local gear tooth fault seeded after the first 600 healthy samples, with a clear spike in the discrepancy synchronous average at around $\frac{4\pi}{5}$ radians. This shows that the fault is synchronous with the shaft rotation, indicating that it is not a bearing fault. Other than the clear line at $\frac{4\pi}{5}$ radians, there are also 36 other lighter lines from sample number 600 onward, each corresponding to one of the other 36 gear teeth meshing. This indicates that after the fault was seeded, and the gearbox was reassembled, the meshing of the healthy gear teeth also caused slightly higher vibrations. The 15^{th} line of the 37 is the one associated with the fault. One can also see that as the fault developed, the 14^{th} line also starts to light up, showing that as the tooth starts to fail, the fault can be seen earlier during the meshing cycle, with anomalous vibrations seen as soon as the tooth makes contact with the pinion. Looking at the order spectra plot in Fig. 62(b), one can see that after sample 600, discrepancies at shaft order 1 and its harmonics become visible, in-

dicating a fault synchronous with the shaft rotation. Also note the clear spike at order 37, showing that there is additional vibration associated with the gear teeth meshing after the gearbox was reassembled.

When deep learning models are trained to classify faults from raw vibration data based on class labels assigned to each sample, all of the diagnostic information discussed above are lost, with the model only returning a label such as 'gear tooth fault'. The models will also not be able to classify gear or bearing faults from other assets that are not included in the labelled training data. A discrepancy analysis framework can add great value to transfer learning approaches, since the discrepancy signals for a local gear tooth fault will look similar to the plots shown in Fig. 62 for most assets.



(a) Synchronous average

(b) Order spectra

Figure 62: Gear tooth fault classification: Synchronous average and order spectra plots of the reconstruction space discrepancy signal, as it develops over the 1800 samples (PCA-C-LR-S, OTA-1-tooth, low-pass filtered dataset)

5.6 Conclusion

On the low-pass filtered dataset, the models performed very similarly to what was seen in the phenomenological model dataset analysis. This shows that the discussions in Chapter 4 generalize to real-world data as well when the real data samples do not have too many impulsive components associated with healthy conditions. For the sake of brevity, these trends are not repeated here. It was seen that the kurtosis of the discrepancy signals is very sensitive to any uncaptured impulsive healthy components, showing that the good results obtained by monitoring the kurtosis in Chapter 4, do not necessarily generalize to real-world data.

The raw unfiltered dataset is very difficult to analyse in an unsupervised manner. The investigation showed that unsupervised PCA and VAE models break down completely when the data is not order-tracked and contains other impulsive cyclostationary components not associated with damage. When one incorporates the available operating conditions into the VAE and PCA frameworks in a semi-supervised manner, good results can be achieved. For the phenomenological model dataset, the supervised C-decoder model and semi-supervised latent variable models yielded similar results, showing little benefit in using latent variable models. This was however not the case for the C-AIM gearbox dataset. The phase information of the non-synchronous components at 5.72 shaft orders is not known, so the supervised C-decoder model cannot capture these components. Some of the latent variable models were however able to successfully capture these components in the latent space, showing the benefit of using latent variable models even when the shaft speed and phase information are known.

It was also shown that the semi-supervised approaches have the potential to perform just as well as traditional signal processing approaches, without the requirement of having to successfully identify the frequency band where the fault presents itself.

6 Conclusion and Future Research

6.1 Conclusion

To conclude, we revisit the three questions listed in the Scope of Research in Section 1.5.

When the operating conditions (shaft speed and phase information) are known or can be estimated, is it necessary to use latent variable models to model the healthy data distribution, or can the operating conditions simply be mapped to the healthy data in a supervised setting?

In the phenomenological model dataset analysis, the C-decoder performed just as well as the best-performing latent variable models when the signals were order-tracked. This makes it difficult to motivate the use of latent variable models over a simple supervised learning approach for this dataset, when the shaft speed and phase information are known. The variance in the data is sufficiently explained for discrepancy analysis purposes by only considering the shaft speed and phase information, which shows that, for this dataset, the latent space representations of the latent variable models are only abstract representations of the shaft speed and phase information, capturing little other useful information from the signals.

In the C-AIM dataset analysis, there are clear advantages to using latent variable models rather than the supervised C-decoder approach. While the C-decoder performed just as well as the latent variable models on the lowpass filtered dataset when the impulsive components in the healthy data were mostly removed, this was not the case for the unfiltered dataset. The iVAE-L model performed significantly better than the C-decoder model, showing that the iVAE-L model is capable of capturing some of the impulsive cyclostationary components that are associated with healthy conditions, while the C-decoder is not. The impulsive cyclostationary components are not described by the shaft speed and phase information, so the C-decoder cannot be used to model these components. As a consequence, these components associated with healthy conditions show up in the discrepancy signals, making fault detection and tracking more difficult. The latent variable models can however capture the generative factors behind these impulsive components in abstract latent space representations, and as a result, return less noisy discrepancy signals.

The performance of the C-decoder gave a clear indication of the differences in complexity between the generated phenomenological model data and the real-world C-AIM dataset, with the C-AIM dataset containing a lot of information that cannot be modelled by only considering the shaft speed and phase information.

The good performance of the C-decoder on the phenomenological model dataset and the filtered C-AIM dataset shows that, in order to perform well on these datasets, the latent variable models must primarily learn to capture the generative factors behind the shaft speed and phase information in the latent representations. When the latent variable models perform poorly, it is an indication that the models fail to capture these generative factors. With this insight, we can conclude that the poor performances of the unsupervised models are caused by the models not accurately capturing the varying phases and frequencies in the data caused by the varying operating speed conditions. One can therefore expect that feeding the shaft speed and phase information directly into the model in a semi-supervised setting would drastically improve the results.

To what extent can the availability of the instantaneous shaft speed and phase information improve the performance of latent variable models on fault detection and tracking tasks, when it is used in a semi-supervised setting?

In the unsupervised setting, the models struggled to capture the healthy data distribution accurately. For the phenomenological dataset, the VAE-L model was the only model that was able to capture the SO data fairly well, and for the C-AIM gearbox dataset, all the unsupervised approaches failed to capture the healthy data distribution. This shows that the normal PCA and VAE frameworks cannot be used to reliably detect and track faults under varying operating conditions in an unsupervised setting. The unsupervised approaches struggle to capture the varying frequencies associated with data from varying operating speeds that are not order-tracked.

The PCA models require many principal components to accurately reconstruct all the frequencies associated with the healthy data. Adding more principal components to reconstruct more of the frequencies associated with the healthy data leads to overfitting (fitting towards an identity transformation). Overfitting is a big problem in the unsupervised setting, since the models accurately reconstruct some of the damage components, making it necessary to model the latent space for discrepancies. Since no operating condition information is available in the unsupervised setting, there is no way to learn what the expected latent representation for a sample at certain operating conditions and a healthy machine state would be, making it difficult to monitor the latent space for discrepancies. It is also challenging to accurately capture the heteroscedastic noise in the data with the linear PCA models in an unsupervised setting, since PCA is formulated under the assumption of constant noise across the whole dataset. When the variance is not captured per sample, the discrepancy signals are correlated with the shaft rotational speed, making it difficult to distinguish between the effect of damage and the effect of varying speeds. The PCA-Z-LR model was proposed to learn the heteroscedastic variance in an unsupervised setting, by mapping the latent representations (which are expected to include variables that are a proxy for the shaft speed) to the reconstruction difference mean and variance, using a linear regression model. While the model managed to accurately capture the variance when the input data distribution was relatively simple (order-tracked data containing few impulsive components), the linear regression model broke down when the input data distribution was more complex. This is attributed to the fact that the latent space representations contain some of the noise present in the input data, making it difficult to capture the correlation between the latent space representations and the input data variance.

For the VAE framework, it was shown that the zero-centred isotropic variance latent prior that is usually chosen when implementing VAEs, is not well suited to capture the periodic vibration signals. Because of its periodic nature, circular manifolds are learned. In this case, the isotropic variance latent prior commonly used to enhance disentanglement in VAEs, leads to posterior collapse for low-dimensional latent spaces. When posterior collapse occurs, the healthy data samples are captured very poorly. The neural network activation functions commonly used in encoders and decoders are also not well suited to capture periodic signals. This was seen most clearly in the C-decoder struggling to reconstruct any of the phenomenological model SO data, when the shaft speed and phase information are fed directly into the model. The neural network architecture has no inductive periodic bias, therefore not learning to understand the concepts of phase and frequency shifts in the data.

The fault detection results were significantly improved by incorporating the shaft speed and phase information during pre-processing (order tracking) and during modelling. The PCA models are able to fit the order-tracked data accurately with only a few principal components, since the order-tracked data do not include frequency components over the full range of the operating speeds. The heteroscedastic noise in the data was also captured more accurately in the semi-supervised setting, with

the PCA-C-LR model proving to be much more robust than the PCA-Z-LR model. When the shaft speed and phase information are acquired from a tacho signal, the operating condition information will not be influenced by the noise in the input vibration data. When it is obtained with tacholess order tracking methods, the noise in the vibration data will affect the operating conditions slightly. It is however still a much less noisy representation of the operating conditions than the latent space representations, and for this reason, the PCA-C-LR model can model the correlation with the input data variance using linear regression more accurately than the PCA-Z-LR model.

The VAE framework was changed to a semi-supervised framework by incorporating the shaft speed and phase information into the iVAE model. By conditioning the latent prior on the shaft speed and phase information, the iVAE was able to learn a more informed prior. The iVAE-L model managed to capture the impulsive cyclostationary components in the C-AIM gearbox dataset, which proved to be a big challenge for the other latent variable models.

The latent space representations were also successfully monitored for discrepancies by learning the distribution of the healthy latent representations conditioned on the associated shaft speed and phase information. It was shown that it is especially important to monitor the latent space for discrepancies when the model starts to overfit to an identity transformation function, since the damage is then not detected clearly in the reconstruction space. The PCA models with large latent space sizes benefitted the most from monitoring the latent space for discrepancies, since these models easily overfit towards an identity transformation function. The VAE models were found to be less prone to overfitting because of the regularization applied by latent space prior. As a result, the VAE models performed relatively the same when only the reconstruction space was monitored for discrepancies, with the latent space and reconstruction space both being evenly responsive to damage.

Until the PCA and VAE frameworks can be improved to robustly capture the varying frequencies associated with varying speed conditions, it should be considered to incorporate the shaft speed and phase information in a semi-supervised setting. This information can be extracted from tacho signals, or by implementing tacholess order tracking methods when tacho signals are not available.

How do the discrepancy analysis methods perform compared to traditional envelope analysis methods?

In the phenomenological model dataset analysis, monitoring the NES of the bandpass-filtered signal outperformed all the discrepancy analysis methods. This result can be expected, since the bearing fault presents itself in a frequency band where there is little information related to other components. When this frequency band of interest is successfully identified and bandpass filtered, most of the components that complicate the condition inference task is removed, and it is then a simple task to detect and track the fault with traditional envelope techniques.

This was however not the case for the C-AIM gearbox dataset. In this case, the frequency band of interest still contained components not related to the fault. It was shown that the semi-supervised approaches have the potential to perform just as well as traditional signal processing approaches, without the requirement of having to successfully identify the frequency band where the fault presents itself. This result is very promising, since it is not an easy task to successfully identify the frequency bands where the fault presents itself in this dataset, with traditional methods such as the fast kurtogram and ICS2gram identifying the wrong frequency bands.

This highlights the potential of the learning-based approaches to automatically extract fault indicators from the vibration data, without incorporating domain knowledge of how a fault presents itself into the

methods. This work does not attempt to prove that either signal processing approaches or learning-based approaches perform better than the other. The underlying question relates more to the extent to which domain knowledge is necessary and available. These learning-based methods provide more than viable options when domain expertise is not readily available.

6.2 Future Research

The following research areas can be explored in future work:

- The discrepancy analysis framework can be extended to fault classification tasks, as was explained in Sections 4.5 and 5.5. Discrepancy signals can be extracted from available datasets with labelled failure data, to build a dataset of discrepancy signals for certain fault classes. Transfer learning between models trained on discrepancy signals should be easier than models trained on raw vibration signals, since the discrepancy signals mostly contain only the relevant fault information, with little information about the machine or application captured in the data. It is however very important that clean discrepancy signals are extracted from the data, where the healthy components in the data are all captured well and do not appear in the discrepancy signals.

- A benchmark real-world dataset is required for fault detection and tracking tasks under highly varying operating conditions. It is very difficult to compare different implementations found in literature, since most are applied to datasets with stationary operating conditions. The C-AIM gearbox dataset provides a viable benchmark option for varying operating conditions, but it will be useful to have a dataset where the speed profile also varies greatly between samples. If the C-AIM gearbox dataset is used, the samples can be split as was done in this work, to ensure that the speed profile varies between samples. Standard performance metrics should also be proposed and documented so that studies that implement these performance metrics can be easily compared. A phenomenological model benchmark dataset will also be useful, since all components in the data will then be fully interpretable, making it easier to identify exactly what component models struggle to capture.

- The challenge of capturing first-order and second-order cyclostationary signals using deep learning models has not been addressed at a more fundamental level, where simple toy datasets are used to investigate how well different generative model architectures perform at these learning tasks. A study that shows which models successfully capture the first-order cyclostationary components in the reconstruction mean, and the second-order cyclostationary components in the reconstruction variance, will be very beneficial to the condition monitoring field. Current condition monitoring literature is mainly focused on showing that the models can be used for end-level tasks such as fault detection, tracking and classification, while it should be possible to predict how the models will perform much earlier, by only focusing on how well the models fit the data.

- A study investigating how deep learning models interpolate and extrapolate for data from frequencies not included in the training data will be beneficial. The work of Ziyin et al. (2020) explains that current neural network architectures are not well suited for capturing periodic signals. In the context of discrepancy analysis on vibration data, this directly affects the results, since the models do not extrapolate well outside the training data operating conditions, and the models have no inductive periodic bias that can give the models an idea that the information of interest appears periodically in the data. It will also be beneficial for fault detection and tracking purposes if the model weights or latent representations capture interpretable frequency content, similar to what Hou et al. (2022) proposed.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems'. Software available from tensorflow.org.
URL: *https://www.tensorflow.org/*

Abboud, D., Antoni, J., Sieg-Zieba, S. & Eltabach, M. (2017), 'Envelope analysis of rotating machine vibrations in variable speed conditions: A comprehensive treatment', *Mechanical Systems and Signal Processing* **84**, 200–226.
URL: *https://doi.org/10.1016/j.ymssp.2016.06.033*

Abboud, D., Elbadaoui, M., Smith, W. & Randall, R. (2019), 'Advanced bearing diagnostics: A comparative study of two powerful approaches', *Mechanical Systems and Signal Processing* **114**, 604–627.
URL: *https://doi.org/10.1016/j.ymssp.2018.05.011*

Antoni, J. (2007), 'Fast computation of the kurtogram for the detection of transient faults', *Mechanical Systems and Signal Processing* **21**(1), 108–124.
URL: *https://doi.org/10.1016/j.ymssp.2005.12.002*

Antoni, J. (2009), 'Cyclostationarity by examples', *Mechanical Systems and Signal Processing* **23**(4), 987–1036.
URL: *https://doi.org/10.1016/j.ymssp.2008.10.010*

Baggerohr, S. (2019), 'A deep learning approach towards diagnostics of bearings operating under non-stationary conditions'. Master's dissertation, University of Pretoria.
URL: *https://repository.up.ac.za/handle/2263/73452*

Balshaw, R. (2020), 'Latent analysis of unsupervised latent variable models in fault diagnostics of rotating machinery under stationary and time-varying operating conditions'. Master's dissertation, University of Pretoria.
URL: *https://repository.up.ac.za/handle/2263/78513*

Balshaw, R., Heyns, P. S., Wilke, D. N. & Schmidt, S. (2022), 'Importance of temporal preserving latent analysis for latent variable models in fault diagnostics of rotating machinery', *Mechanical Systems and Signal Processing* **168**, 108663.
URL: *https://doi.org/10.1016/j.ymssp.2021.108663*

Bartelmus, W. & Zimroz, R. (2009), 'A new feature for monitoring the condition of gearboxes in non-stationary operating conditions', *Mechanical Systems and Signal Processing* **23**(5), 1528–1534.
URL: *https://doi.org/10.1016/j.ymssp.2009.01.014*

Bengio, Y., Courville, A. & Vincent, P. (2012), 'Representation learning: A review and new perspectives'.
URL: *https://doi.org/10.48550/ARXIV.1206.5538*

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg.

Booyse, W., Wilke, D. N. & Heyns, S. (2020), 'Deep digital twins for detection, diagnostics and prognostics', *Mechanical Systems and Signal Processing* **140**, 106612.
URL: *https://doi.org/10.1016/j.ymssp.2019.106612*

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G. & Lerchner, A. (2018), 'Understanding disentangling in β-VAE'.
URL: *https://doi.org/10.48550/arXiv.1804.03599*

Cerrada, M., Sánchez, R.-V., Li, C., Pacheco, F., Cabrera, D., Valente de Oliveira, J. & Vásquez, R. E. (2018), 'A review on data-driven fault severity assessment in rolling bearings', *Mechanical Systems and Signal Processing* **99**, 169–196.
URL: *https://doi.org/10.1016/j.ymssp.2017.06.012*

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T. & Tomczak, J. M. (2018), 'Hyperspherical variational auto-encoders', *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)* .
URL: *https://doi.org/10.48550/arXiv.1804.00891*

Denouden, T., Salay, R., Czarnecki, K., Abdelzad, V., Phan, B. & Vernekar, S. (2018), 'Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance'.
URL: *https://doi.org/10.48550/ARXIV.1812.02765*

Diamond, D. H., Heyns, P. S. & Oberholster, A. J. (2016), 'Online shaft encoder geometry compensation for arbitrary shaft speed profiles using bayesian regression', *Mechanical Systems and Signal Processing* **81**, 402–418.
URL: *https://doi.org/10.1016/j.ymssp.2016.02.060*

Fefferman, C., Mitter, S. & Narayanan, H. (2016), 'Testing the manifold hypothesis', *Journal of the American Mathematical Society* **29**(4), 983–1049.
URL: *https://doi.org/10.1090/jams/852*

Fyfe, K. R. & Munck, E. D. S. (1997), 'Analysis of computed order tracking', *Mechanical Systems and Signal Processing* **11**(2), 187–205.
URL: *https://doi.org/10.1006/mssp.1996.0056*

Guo, L., Lei, Y., Xing, S., Yan, T. & Li, N. (2019), 'Deep convolutional transfer learning network: A new method for intelligent fault diagnosis of machines with unlabeled data', *IEEE Transactions on Industrial Electronics* **66**(9), 7316–7325.
URL: *https://doi.org/10.1109/TIE.2018.2877090*

Heyns, T., Heyns, P. S. & de Villiers, J. P. (2012), 'Combining synchronous averaging with a gaussian mixture model novelty detection scheme for vibration-based condition monitoring of a gearbox', *Mechanical Systems and Signal Processing* **32**, 200–215. Uncertainties in Structural Dynamics.
URL: *https://doi.org/10.1016/j.ymssp.2012.05.008*

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S. & Lerchner, A. (2017), β-VAE: Learning basic visual concepts with a constrained variational framework, *in* '5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings'.
URL: *https://openreview.net/forum?id=Sy2fzU9gl*

Hong, D., Balzano, L. & Fessler, J. A. (2019), Probabilistic pca for heteroscedastic data, *in* '2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)', pp. 26–30.
URL: *https://doi.org/10.1109/CAMSAP45676.2019.9022436*

Hou, B., Wang, D., Chen, Y., Wang, H., Peng, Z. & Tsui, K.-L. (2022), 'Interpretable online updated weights: Optimized square envelope spectrum for machine condition monitoring and fault diagnosis', *Mechanical Systems and Signal Processing* **169**, 108779.
URL: *https://doi.org/10.1016/j.ymssp.2021.108779*

Hyvarinen, A., Sasaki, H. & Turner, R. E. (2018), 'Nonlinear ICA using auxiliary variables and generalized contrastive learning'.
URL: *https://doi.org/10.48550/arXiv.1805.08651*

Janssens, O., Slavkovikj, V., Vervisch, B., Stockman, K., Loccufier, M., Verstockt, S., Van de Walle, R. & Van Hoecke, S. (2016), 'Convolutional neural network based fault detection for rotating machinery', *Journal of Sound and Vibration* **377**, 331–345.
URL: *https://doi.org/10.1016/j.jsv.2016.05.027*

Jardine, A. K., Lin, D. & Banjevic, D. (2006), 'A review on machinery diagnostics and prognostics implementing condition-based maintenance', *Mechanical Systems and Signal Processing* **20**(7), 1483–1510.
URL: *https://doi.org/10.1016/j.ymssp.2005.09.012*

Jia, F., Lei, Y., Lin, J., Zhou, X. & Lu, N. (2016), 'Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data', *Mechanical Systems and Signal Processing* **72-73**, 303–315.
URL: *https://doi.org/10.1016/j.ymssp.2015.10.025*

Khan, S. & Yairi, T. (2018), 'A review on the application of deep learning in system health management', *Mechanical Systems and Signal Processing* **107**, 241–265.
URL: *https://doi.org/10.1016/j.ymssp.2017.11.024*

Khemakhem, I., Kingma, D. P., Monti, R. P. & Hyvärinen, A. (2019), 'Variational autoencoders and nonlinear ICA: A unifying framework'.
URL: *https://doi.org/10.48550/ARXIV.1907.04809*

Kingma, D. P. & Ba, J. (2015), Adam: A method for stochastic optimization, *in* Y. Bengio & Y. LeCun, eds, '3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings'.
URL: *https://doi.org/10.48550/arXiv.1412.6980*

Kingma, D. P. & Welling, M. (2014), Auto-Encoding Variational Bayes, *in* '2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings'.
URL: *https://doi.org/10.48550/arXiv.1312.6114*

Kingma, D. P. & Welling, M. (2019), 'An introduction to variational autoencoders', *Foundations and Trends® in Machine Learning* **12**(4), 307–392.
URL: *https://doi.org/10.1561/2200000056*

Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L. & Siegel, D. (2014), 'Prognostics and health management design for rotary machinery systems—reviews, methodology and applications', *Mechanical Systems and Signal Processing* **42**(1), 314–334.
URL: *https://doi.org/10.1016/j.ymssp.2013.06.004*

Lee, K., Lee, K., Lee, H. & Shin, J. (2018), 'A simple unified framework for detecting out-of-distribution samples and adversarial attacks'.
URL: *https://doi.org/10.48550/ARXIV.1807.03888*

Lee, P. M. (2012), *Bayesian Statistics: An Introduction*, Wiley.

Lei, Y., Jia, F., Lin, J., Xing, S. & Ding, S. X. (2016), 'An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data', *IEEE Transactions on Industrial Electronics* **63**(5), 3137–3147.
URL: *https://doi.org/10.1109/TIE.2016.2519325*

Lei, Y., Li, N., Guo, L., Li, N., Yan, T. & Lin, J. (2018), 'Machinery health prognostics: A systematic review from data acquisition to RUL prediction', *Mechanical Systems and Signal Processing* **104**, 799–834.
URL: *https://doi.org/10.1016/j.ymssp.2017.11.016*

Li, X., Ding, Q. & Sun, J.-Q. (2018), 'Remaining useful life estimation in prognostics using deep convolution neural networks', *Reliability Engineering & System Safety* **172**, 1–11.
URL: *https://doi.org/10.1016/j.ress.2017.11.021*

Liao, L. & Köttig, F. (2014), 'Review of hybrid prognostics approaches for remaining useful life prediction of engineered systems, and an application to battery life prediction', *IEEE Transactions on Reliability* **63**(1), 191–207.
URL: *https://doi.org/10.1109/TR.2014.2299152*

Lin, H. W., Tegmark, M. & Rolnick, D. (2016), 'Why does deep and cheap learning work so well?', *CoRR* **abs/1608.08225**.
URL: *https://doi.org/10.48550/arXiv.1608.08225*

Lin, J. & Zhao, M. (2015), 'A review and strategy for the diagnosis of speed-varying machinery', *2014 International Conference on Prognostics and Health Management, PHM 2014* .
URL: *https://doi.org/10.1109/ICPHM.2014.7036368*

Locatello, F., Bauer, S., Lucic, M., Gelly, S., Schölkopf, B. & Bachem, O. (2018), 'Challenging common assumptions in the unsupervised learning of disentangled representations', *CoRR* **abs/1811.12359**.
URL: *https://doi.org/10.48550/arXiv.1811.12359*

Lu, S., Yan, R., Liu, Y. & Wang, Q. (2019), 'Tacholess speed estimation in order tracking: A review with application to rotating machine fault diagnosis', *IEEE Transactions on Instrumentation and Measurement* **68**(7), 2315–2332.
URL: *https://doi.org/10.1109/TIM.2019.2902806*

Lucas, J., Tucker, G., Grosse, R. B. & Norouzi, M. (2019), Understanding posterior collapse in generative latent variable models, *in* 'Deep Generative Models for Highly Structured Data, DGS@ICLR 2019 Workshop, Conference Track Proceedings'.
URL: *https://openreview.net/forum?id=r1xaVLUYuE*

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D. & Lakshminarayanan, B. (2018), 'Do deep generative models know what they don't know?'.
URL: *https://doi.org/10.48550/ARXIV.1810.09136*

Perez Rey, L. A., Menkovski, V. & Portegies, J. (2020), Diffusion variational autoencoders, *in* C. Bessiere, ed., 'Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20', International Joint Conferences on Artificial Intelligence Organization, pp. 2704–2710.
URL: *https://doi.org/10.24963/ijcai.2020/375*

Randall, R. B. & Antoni, J. (2011), 'Rolling element bearing diagnostics—a tutorial', *Mechanical Systems and Signal Processing* **25**(2), 485–520.
URL: *https://doi.org/10.1016/j.ymssp.2010.07.017*

Schmidt, S. & Gryllias, K. C. (2021), 'The anomalous and smoothed anomalous envelope spectra for rotating machine fault diagnosis', *Mechanical Systems and Signal Processing* **158**, 107770.
URL: *https://doi.org/10.1016/j.ymssp.2021.107770*

Schmidt, S. & Heyns, P. S. (2019), 'An open set recognition methodology utilising discrepancy analysis for gear diagnostics under varying operating conditions', *Mechanical Systems and Signal Processing* **119**, 1–22.
URL: *https://doi.org/10.1016/j.ymssp.2018.09.016*

Schmidt, S. & Heyns, P. S. (2020), 'Normalisation of the amplitude modulation caused by time-varying operating conditions for condition monitoring', *Measurement* **149**, 106964.
URL: *https://doi.org/10.1016/j.measurement.2019.106964*

Schmidt, S., Heyns, P. S. & de Villiers, J. P. (2018*a*), 'A novelty detection diagnostic methodology for gearboxes operating under fluctuating operating conditions using probabilistic techniques', *Mechanical Systems and Signal Processing* **100**, 152–166.
URL: *https://doi.org/10.1016/j.ymssp.2017.07.032*

Schmidt, S., Heyns, P. S. & de Villiers, J. P. (2018*b*), 'A tacholess order tracking methodology based on a probabilistic approach to incorporate angular acceleration information into the maxima tracking process', *Mechanical Systems and Signal Processing* **100**, 630–646.
URL: *https://doi.org/10.1016/j.ymssp.2017.07.053*

Schmidt, S., Heyns, P. S. & Gryllias, K. C. (2019*a*), 'A discrepancy analysis methodology for rolling element bearing diagnostics under variable speed conditions', *Mechanical Systems and Signal Processing* **116**, 40–61.
URL: *https://doi.org/10.1016/j.ymssp.2018.06.026*

Schmidt, S., Heyns, P. S. & Gryllias, K. C. (2019*b*), 'A pre-processing methodology to enhance novel information for rotating machine diagnostics', *Mechanical Systems and Signal Processing* **124**, 541–561.
URL: *https://doi.org/10.1016/j.ymssp.2019.02.005*

Schmidt, S., Mauricio, A., Heyns, P. S. & Gryllias, K. C. (2020), 'A methodology for identifying information rich frequency bands for diagnostics of mechanical components-of-interest under time-varying operating conditions', *Mechanical Systems and Signal Processing* **142**, 106739.
URL: *https://doi.org/10.1016/j.ymssp.2020.106739*

Smith, W. A. & Randall, R. B. (2015), 'Rolling element bearing diagnostics using the case western reserve university data: A benchmark study', *Mechanical Systems and Signal Processing* **64-65**, 100–131.
URL: *https://doi.org/10.1016/j.ymssp.2015.04.021*

Steck, H. (2020), Autoencoders that don't overfit towards the identity, *in* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan & H. Lin, eds, 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 19598–19608.
URL: *https://proceedings.neurips.cc/paper/2020/file/e33d974aae13e4d877477d51d8bafdc4-Paper.pdf*

Tipping, M. E. & Bishop, C. M. (1999), 'Probabilistic principal component analysis', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **61**(3), 611–622.
URL: *https://doi.org/10.1111/1467-9868.00196*

Wang, D., Peng, Z. & Xi, L. (2020), 'The sum of weighted normalized square envelope: A unified framework for kurtosis, negative entropy, gini index and smoothness index for machine health monitoring', *Mechanical Systems and Signal Processing* **140**, 106725.
URL: *https://doi.org/10.1016/j.ymssp.2020.106725*

Wang, D., Tsui, K.-L. & Miao, Q. (2018), 'Prognostics and health management: A review of vibration based bearing and gear health indicators', *IEEE Access* **6**, 665–676.
URL: *https://doi.org/10.1109/ACCESS.2017.2774261*

Wang, Y., Blei, D. & Cunningham, J. P. (2021), Posterior collapse and latent variable non-identifiability, *in* M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang & J. W. Vaughan, eds, 'Advances in Neural Information Processing Systems', Vol. 34, Curran Associates, Inc., pp. 5443–5455.
URL: *https://proceedings.neurips.cc/paper/2021/file/2b6921f2c64dee16ba21ebf17f3c2c92-Paper.pdf*

Wen, L., Gao, L. & Li, X. (2019), 'A new deep transfer learning based on sparse auto-encoder for fault diagnosis', *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **49**(1), 136–144.
URL: *https://doi.org/10.1109/TSMC.2017.2754287*

Wilke, D. N., Heyns, P. S. & Schmidt, S. (2022), The role of untangled latent spaces in unsupervised learning applied to condition-based maintenance, *in* A. Hammami, P. S. Heyns, S. Schmidt, F. Chaari, M. S. Abbes & M. Haddar, eds, 'Modelling and Simulation of Complex Systems for Sustainable Energy Efficiency', Springer International Publishing, Cham, pp. 38–49.
URL: *https://doi.org/10.1007/978-3-030-85584-0_5*

Xiao, Z., Yan, Q. & Amit, Y. (2020), 'Likelihood regret: An out-of-distribution detection score for variational auto-encoder'.
URL: *https://doi.org/10.48550/ARXIV.2003.02977*

Zhang, S., Ye, F., Wang, B. & Habetler, T. G. (2021), 'Semi-supervised bearing fault diagnosis and classification using variational autoencoder-based deep generative models', *IEEE Sensors Journal* **21**(5), 6476–6486.
URL: *https://doi.org/10.1109/JSEN.2020.3040696*

Zhang, S., Zhang, S., Wang, B. & Habetler, T. G. (2020), 'Deep learning algorithms for bearing fault diagnostics—a comprehensive review', *IEEE Access* **8**, 29857–29881.
URL: *https://doi.org/10.1109/ACCESS.2020.2972859*

Zimroz, R., Bartelmus, W., Barszcz, T. & Urbanek, J. (2014), 'Diagnostics of bearings in presence of strong operating conditions non-stationarity—a procedure of load-dependent features processing with application to wind turbine bearings', *Mechanical Systems and Signal Processing* **46**(1), 16–27.
URL: *https://doi.org/10.1016/j.ymssp.2013.09.010*

Ziyin, L., Hartwig, T. & Ueda, M. (2020), 'Neural networks fail to learn periodic functions and how to fix it'.
URL: *https://doi.org/10.48550/ARXIV.2006.08195*

Appendix A Visualization of Periodic Signals in the Latent Space

Condition monitoring signals are approximately periodic relative to the shaft speed of the monitored rotating machine. In this investigation, a β-VAE is trained to reconstruct simple sinusoidal waves with no noise, to see what the learned latent manifold for periodic signals looks like. The amplitude, offset, frequency and phase of the sinusoidal waves are varied. The goal of this investigation is to gain an understanding of how VAEs transform periodic signals into a lower-dimensional latent space, and how the generative factors (the specified amplitude, offset, frequency and phase) are represented in the latent space. The focus is on visualizing the latent space.

This investigation aims to answer the following three questions:

- How is each of the generative factors represented in the latent space?
- How many latent variables are necessary to capture each of the generative factors?
- Is the latent space entangled, and if so, can regularization of the latent space untangle each of the generative factors?

These three questions are revisited and discussed at the end of the appendix.

For the base case, a periodic signal $\boldsymbol{x}$ consisting of two sinusoidal waves with frequencies 3Hz and 5Hz respectively is investigated:

$$x(t) = \sin(2\pi \cdot 3 \cdot t) + \sin(2\pi \cdot 5 \cdot t) \tag{A-1}$$

The base case signal was varied using the following function to generate signals:

$$x(a, p, f, k, t) = a \cdot \sin(2\pi \cdot 3f \cdot (t - p)) + a \cdot \sin(2\pi \cdot 5f \cdot (t - p)) + k \tag{A-2}$$

where a is used to scale the amplitudes of both periodic components of the signal, p is used to shift the phase of the signal, f varies the frequency of the signal, and k is used to vary the offset of the signal.

Note that no noise is added to the signals. If the VAE can capture the signals successfully, it should therefore learn very small variance terms for the reconstruction, as all components are deterministic. Refer to Appendix C for the network architecture of the VAE used. For all cases (except where otherwise specified) the regularization parameter β was set to zero so that no prior distribution is placed on the latent space. The VAE can therefore freely find any latent representation that would lead to accurate reconstruction of the input signals.

Table A.1 contains the ranges between which each parameter was varied, as well as the values it was kept constant at if the effect of the specific parameter is removed from the investigation. Signals were generated for a time window of 2 seconds, with 1024 samples. f was never varied below 1 so that each sample contains 2 or more periodic cycles. p is varied from 0 to 1, as this will shift the signal through the full periodic cycle (the signal is the same for $p = 0$ and $p = 1$).

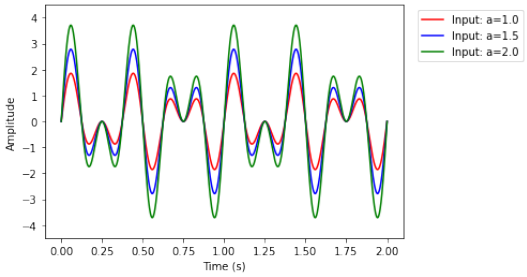
Table A.1: Parameter values when kept constant and when varied

Parameter	Constant	Varying	
		min	max
a	1	1	2
p	0	0	1
f	1	1	3
k	0	-1	1

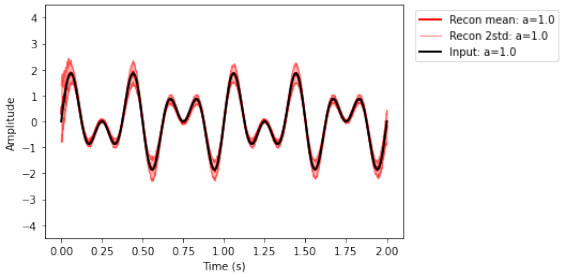
In the following subsections, the effect of each parameter on the latent space is first isolated, by varying one parameter at a time while keeping the other parameters constant. Thereafter combinations of the parameters are varied.

A.1 Varying amplitude

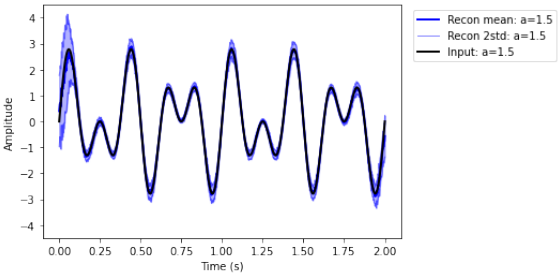
The VAE was trained on signals where only a was varied. Fig. A.1 shows the reconstructions of three of the samples, for a VAE with a 1D latent space (so only 1 latent variable). It can be seen that the VAE can accurately capture the input signals with this 1 latent variable, giving accurate reconstructions of the input signals. Small variances are learned, so the VAE can reconstruct the given signals with high certainty.



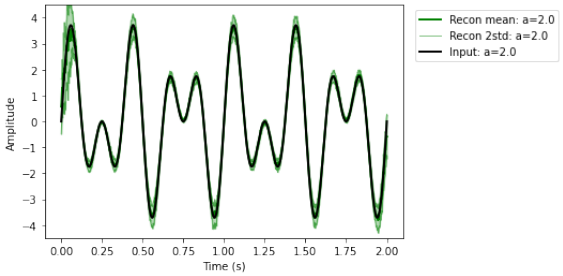
(a) Samples



(b) $a = 1$



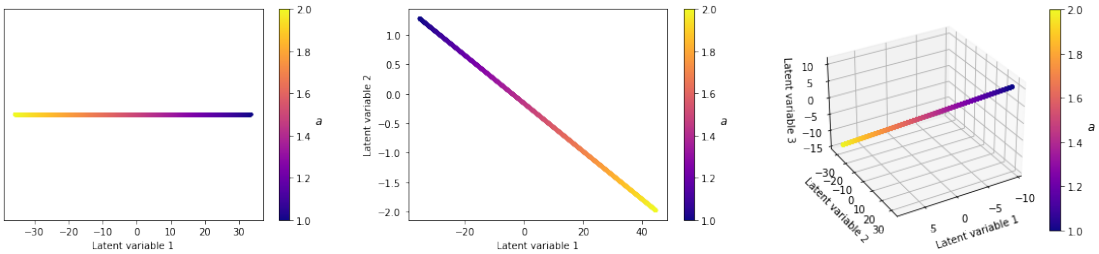
(c) $a = 1.5$



(d) $a = 2$

Figure A.1: Varying amplitude: Reconstructions of three of the samples for 1D latent space

Fig. A.2 shows the learned latent representations when using a 1D, 2D and 3D latent space. In all three cases, the VAE learns to project the signals to a 1D manifold in the latent space, with the variations in a mapped to distinct regions.

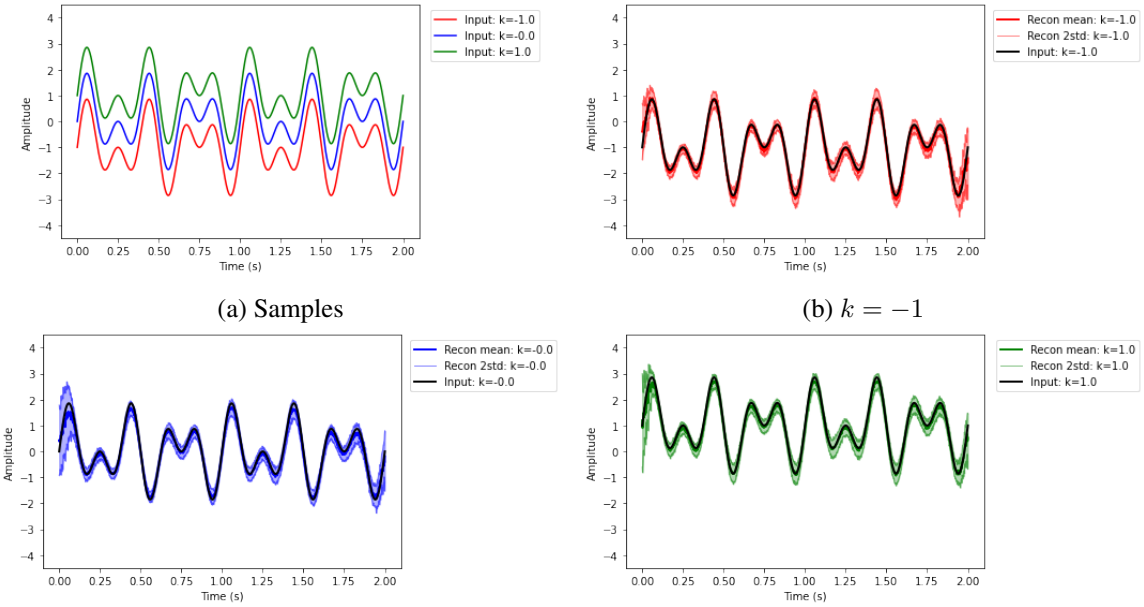


(a) 1D (b) 2D (c) 3D

Figure A.2: Varying amplitude: Latent representations for latent spaces with different dimensionalities

A.2 Varying offset

The VAE was trained on signals where only k was varied. Fig. A.3 shows the reconstructions of three of the samples, for a VAE with a 2D latent space. It can be seen that the VAE can accurately capture the input signals with these 2 latent variables, giving accurate reconstructions of the input signals. Small variances are learned, so the VAE can reconstruct the given signals with high certainty.

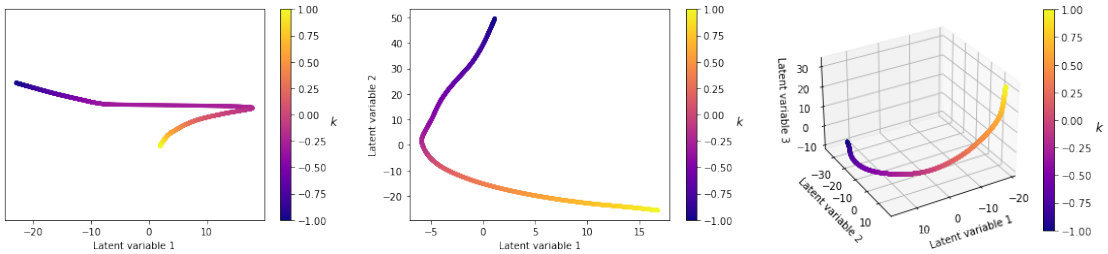


(a) Samples (b) $k = -1$

(c) $k = 0$ (d) $k = 1$

Figure A.3: Varying offset: Reconstructions of three of the samples for 2D latent space

Fig. A.4 shows the learned latent representations when using a 1D, 2D and 3D latent space. The representations for the 1D latent space are shifted slightly in the y-axis to allow for better visualization. It is clear that for the 1D manifold, there are overlapping representations for some ranges of k. Even though the learned representation varies continuously as k is varied, this specific learned representation will give poor reconstructions when k is above -0.5, for instance, $k = 1$ and $k = -0.25$ are represented with the same latent space representation. The VAE will therefore have trouble deciding if a signal with $k = 1$ should be reconstructed as a signal with $k = 1$ or $k = -0.25$. In all three cases, the VAE learns to project the signals to a 1D manifold in the latent space. In the 2D and 3D latent spaces, the variations in k are mapped to distinct regions, and this shows why accurate reconstructions can be achieved for a VAE with 2 latent variables.

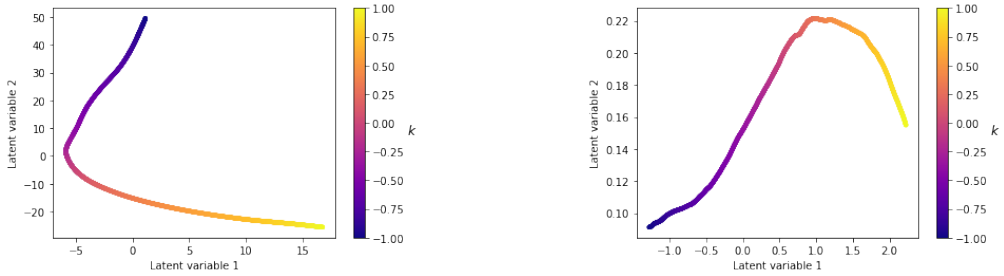


(a) 1D (b) 2D (c) 3D

Figure A.4: Varying offset: Latent representations for latent spaces with different dimensionalities

What is seen in the case of the 1D latent space representation, can also be viewed as some form of latent space entanglement. Even though the samples fall on a 1D manifold, the VAE cannot project this 1D manifold to distinct regions in the latent space, without the learned manifold intersecting itself. This is largely influenced by the initial values of the weights of the VAE network, so for some cases, the VAE might learn an untangled representation with only 1 latent variable, but to ensure that the VAE captures the 1D manifold accurately each time, 2 latent variables are necessary.

By applying latent space regularization (making β larger than 0), one of the latent variables can then be squeezed to a small range, so that the other latent variable captures the variance in k. This is visualized in Fig. A.5. Note how, when $\beta = 0$, both the latent variables in this 2D latent space are spread over large ranges (in this case latent variable 1 lies between -6 and 18, and latent variable 2 between -25 and 50). When $\beta = 1$, latent variable 2 gets compressed to a small range (between 0.1 and 0.22), with latent variable 1 capturing the variance in k.

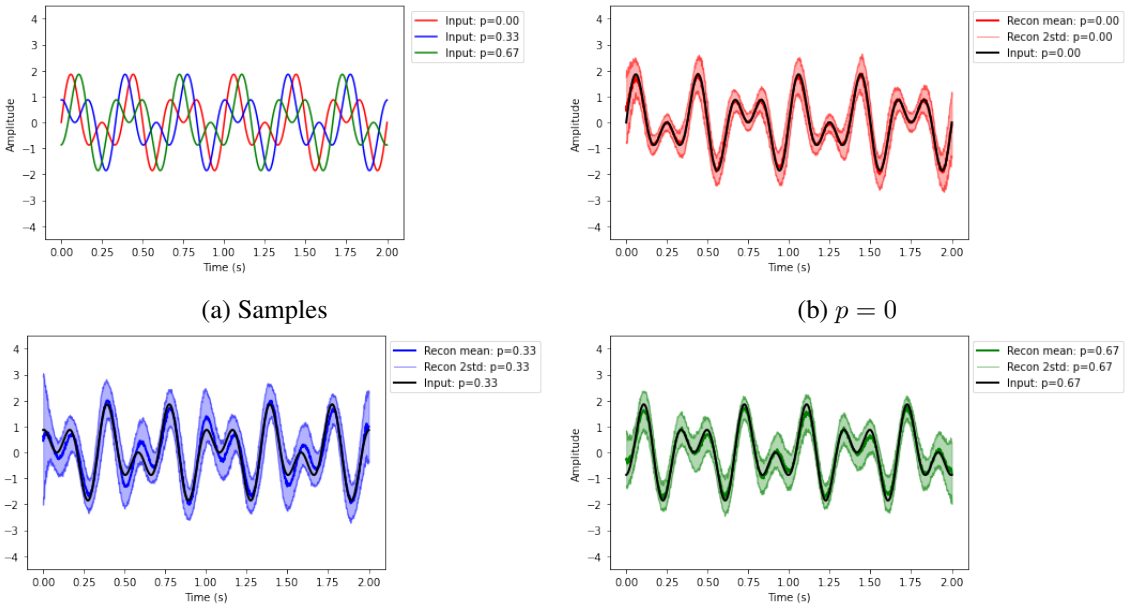


(a) $\beta = 0$ (b) $\beta = 1$

Figure A.5: Varying offset: Effect of increasing β on 2D latent space

A.3 Varying phase

The VAE was trained on signals where only p was varied. Fig. A.6 shows the reconstructions of three of the samples, for a VAE with a 2D latent space. It can be seen that the VAE can accurately capture the input signals with these 2 latent variables, giving accurate reconstructions of the input signals. The variances learned are however larger, so the VAE cannot reconstruct the given signals with as much certainty as was the case for the variations in a and k. The reason for this larger variance will be discussed shortly.



(a) Samples

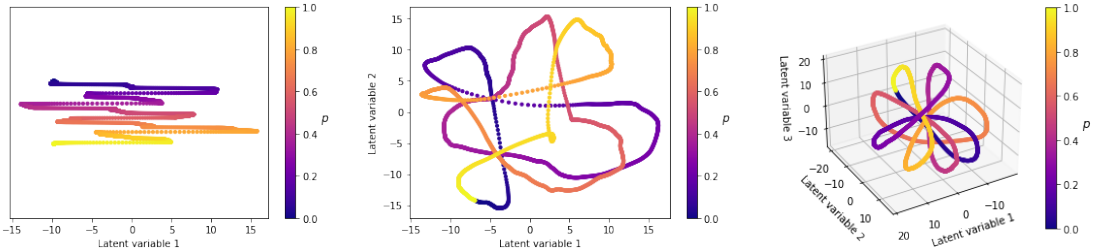
(b) $p = 0$

(c) $p = 0.33$

(d) $p = 0.67$

Figure A.6: Varying phase: Reconstructions of three of the samples for 2D latent space

Fig. A.7 shows the learned latent representations when using a 1D, 2D and 3D latent space. Once again, the representations for the 1D latent space are shifted slightly along the y-axis to allow for better visualization. For the 1D manifold, there are many overlapping representations for some ranges of p. Even though the learned representation varies continuously as p is varied, this specific learned representation will give poor reconstructions in the overlapping regions. In all three cases, the VAE learns to project the signals to a circular 1D manifold in the latent space (note how the colours representing $p = 0$ and $p = 1$ meet in all three cases). In the 2D and 3D latent spaces, the variations in p are mapped to more distinct regions, but there are still some places where the manifold intersects itself, which will lead to poor reconstruction.



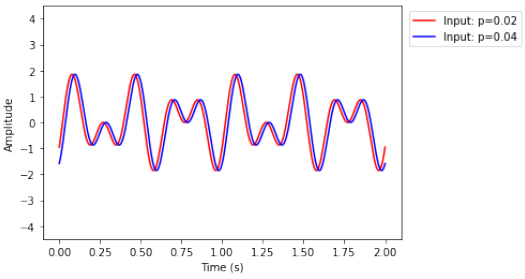
(a) 1D

(b) 2D

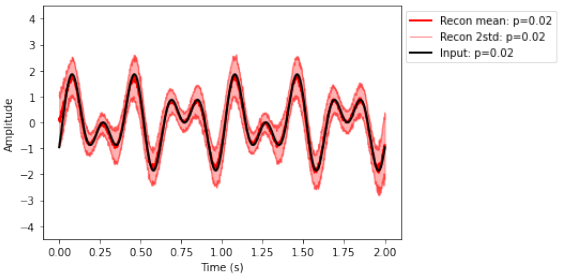
(c) 3D

Figure A.7: Varying phase: Latent representations for latent spaces with different dimensionalities

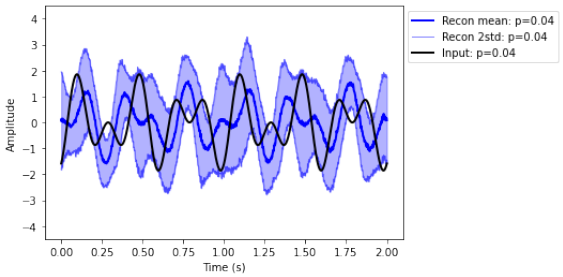
This poor reconstruction for signals where the latent manifold intersects itself is shown in Fig. A.8. Two signals very close to each other, with $p = 0.02$ and $p = 0.04$, are considered. Looking at the 2D latent space in Fig. A.7, it can be seen that close to $p = 0.04$, the manifold intersects itself twice. This leads to the signal with $p = 0.04$ being reconstructed poorly, with great uncertainty, while the signal with $p = 0.02$ is reconstructed very well, as it is far enough from the intersection in the latent space.



(a) Samples



(b) No overlapping regions in latent space



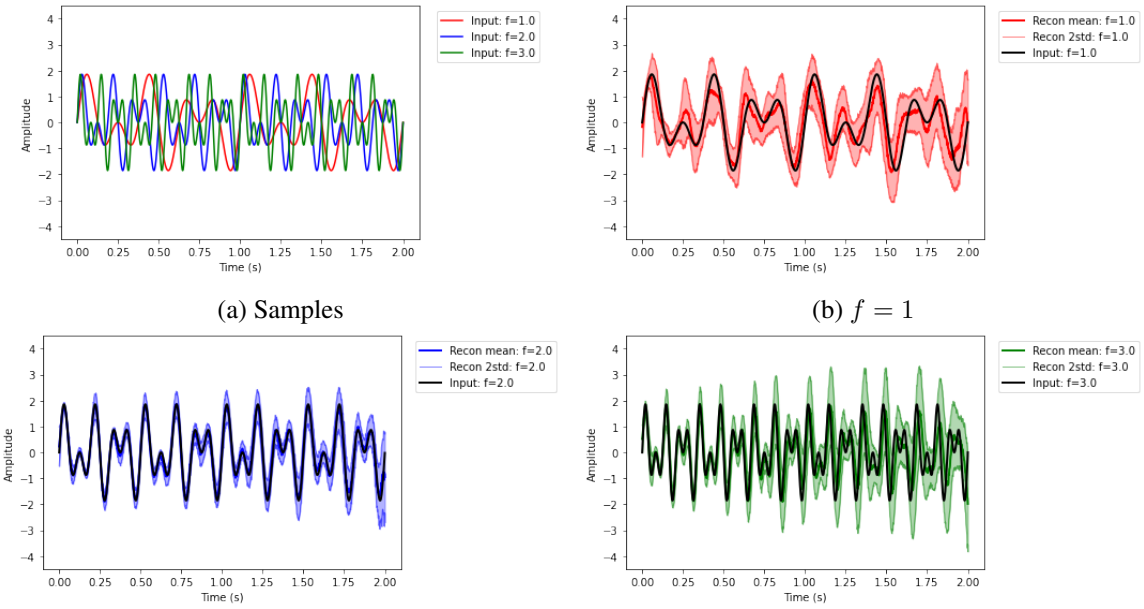
(c) Overlapping regions in latent space

Figure A.8: Varying phase: Poor reconstruction of the samples close to overlapping regions in 2D latent space

The VAE maps the phase shift to a circular manifold. The circular manifold is by definition entangled, as two correlated latent variables are necessary to define the circle. The direct partitioning scheme and continuous time approach proposed by Balshaw (2020) introduce this phase shift to condition monitoring signals, and therefore one can expect to see circular manifolds when learning a lower dimensional manifold of the condition monitoring signals.

A.4 Varying frequency

The VAE was trained on signals where only f was varied. The reconstructions of three of the samples, for a VAE with a 3D latent space, are shown in Fig. A.9. It can be seen that the VAE can accurately capture most of the input signals with these 3 latent variables, giving accurate reconstructions of the input signals. The variances are large in some cases, with poor reconstruction.



(a) Samples

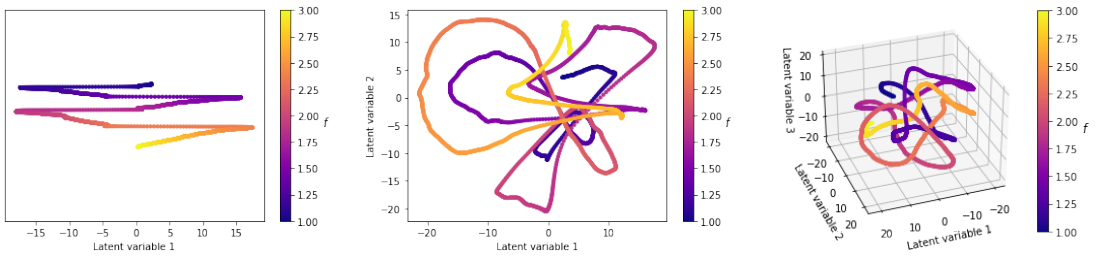
(b) $f = 1$

(c) $f = 2$

(d) $f = 3$

Figure A.9: Varying frequency: Reconstructions of three of the samples in 3D latent space

Looking at the learned latent representations when using a 1D, 2D and 3D latent space, shown in Fig. A.10, it can be seen that intersections in the manifold once again cause poor reconstruction for some samples. Once again, the representations for the 1D latent space are shifted slightly along the y-axis to allow for better visualization. For the 1D manifold, there are many overlapping representations for some ranges of f. In all three cases, the VAE learns to project the signals to a 1D manifold in the latent space. Note that the manifold is not circular as was the case with varying the phase of the signal. The latent spaces are very entangled, and it is clear that learning interpretable latent spaces with a VAE for signals with varying frequencies, is not an easy task.



(a) 1D

(b) 2D

(c) 3D

Figure A.10: Varying frequency: Latent representations for latent spaces with different dimensionalities

A.5 Varying multiple generative factors

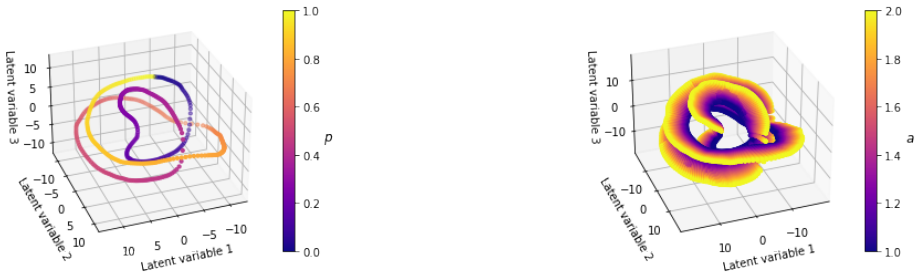
Multiple combinations of varying different factors simultaneously were investigated, but only the ones that can be easily interpreted visually in a 3D latent space are presented in this section. Varying both the frequency and the phase simultaneously leads to very complicated manifolds, with many intersections for a 3D latent space. One will have to use more than 3 latent variables to accurately capture the signal, making it difficult to visualize all latent variables simultaneously.

Three combinations are presented:

- Varying phase and amplitude simultaneously
- Varying frequency and amplitude simultaneously
- Varying two different amplitude factors simultaneously, by modifying Eq. (A-2) so that each frequency component is scaled by a different amplitude:

$$x(a_1, a_2, t) = a_1 \cdot \sin(2\pi \cdot 3 \cdot t) + a_2 \cdot \sin(2\pi \cdot 5 \cdot t) \tag{A-3}$$

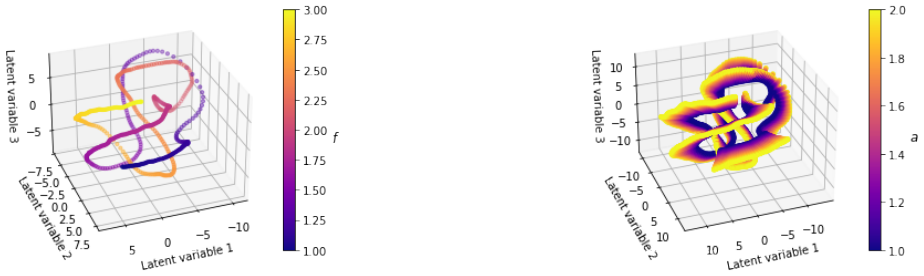
Fig. A.11 shows the latent representations for a 3D latent space when varying both the amplitude (a) and the phase (p). Fig. A.11(a) shows how the circular 1D manifold is still captured when a is constant, and Fig. A.11(b) shows how, when the amplitudes are varied, the larger amplitudes are projected outwardly from this 1D circular manifold. The resulting manifold is thus a 2D manifold, a strip connected at the ends to form a circular loop. Some intersections of the manifold are clearly visible, so to get accurate reconstructions for all signals, the latent space dimensionality will have to be chosen larger than 3 dimensions, or the latent variables must be disentangled, where two latent variables form the circle caused by the phase changes, and the third captures the changes in amplitude.



(a) p representation for $a = 1$ (b) a representation for p ranging from 0 to 1

Figure A.11: Varying phase and amplitude: Latent representations for 3D latent space

Fig. A.12 shows the latent representations for a 3D latent space when varying both the amplitude (a) and the frequency (f). Fig. A.12(a) shows how the 1D manifold is still captured when a is constant, and Fig. A.12(b) shows how, when the amplitudes are varied, the larger amplitudes are projected outwardly from this 1D manifold. The resulting manifold is thus a 2D manifold, a strip, but with the ends not connected as was the case for varying amplitude and phase.

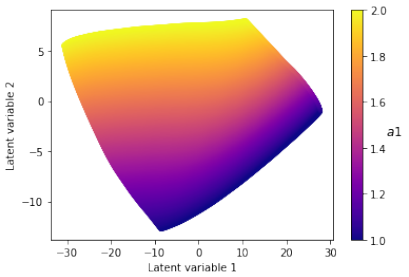


(a) f representation for $a = 1$ (b) a representation for f ranging from 1 to 3

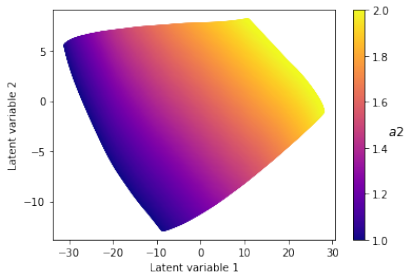
Figure A.12: Varying frequency and amplitude: Latent representations for 3D latent space

Fig. A.13 shows the latent representations for a 2D and 3D latent space when varying both the amplitudes in Eq.(A-3) (a_1 and a_2). A clear 2D manifold is learned, with no intersections. Note

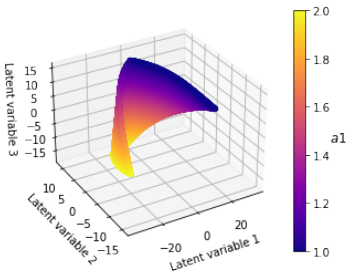
that this VAE will give very accurate reconstructions for all the signals presented to the VAE, as the manifold has no intersections. This latent space is however still considered entangled, as a variation in one of the generative factors a_1 or a_2 will lead to changes in all the latent variables. This entanglement will however not affect the reconstruction ability of the VAE, and will therefore not have a negative effect on anomaly detection. This highlights an important point for using latent variable models for fault detection and tracking: fault detection is not affected by latent space entanglement as much as it is by intersections in the learned manifold. The goal should therefore be to first learn manifolds with no intersections, before trying to disentangle the factors of variation.



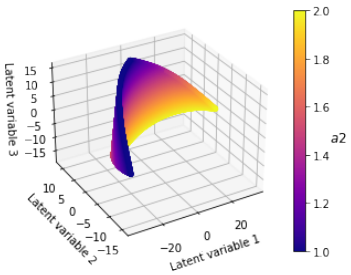
(a) 2D representation of a_1



(b) 2D representation of a_2



(c) 3D representation of a_1



(d) 3D representation of a_2

Figure A.13: Varying amplitudes of different generative factors: Latent representations for 2D and 3D latent space

A.6 Discussion

This investigation was conducted to aid in the development of a good intuition of how the inputs are projected to the lower dimensional latent space, before it is attempted to use the learned latent representations for anomaly detection. The questions at the start of the appendix can now be answered.

How is each of the generative factors represented in the latent space? Variations in amplitude, phase, frequency and offset are each projected to 1-dimensional manifolds. The generative factor associated with the phase leads to a circular 1D manifold. If each of these factors is only defined by a single parameter, as is the case in Eq. (A-2), and all four generative factors are varied simultaneously, then all the signals generated will fall on a 4D manifold in the latent space. All possible signal segments generated with Eq. (A-2) can be projected to a 4D manifold. This does not mean that only 4 or 5 latent variables are required to capture all these signal segments, as the latent space will be highly entangled, resulting in intersections in the learned manifold.

How many latent variables are necessary to capture each of the generative factors? Theoretically, it should be possible to capture a normal 1D manifold with one latent variable, and a 1D circular manifold with 2 latent variables. It was however seen that restricting the latent space to the exact minimum size necessary, will often lead to intersections in the learned manifold, resulting in poor

reconstructions. These intersections are influenced by the initial weight parameters of the network, so there are some cases where the network weights are initialized in such a way that an untangled latent space without intersections in the manifold can be learned, but this will seldomly happen. It is therefore more robust to use a larger-than-necessary latent space dimensionality, and then regularize the latent space to force variance to be captured in the minimum number of latent variables necessary.

Is the latent space entangled, and if so, can regularization of the latent space untangle each of the generative factors? It was shown in this investigation that entangled and untangled latent spaces can have different meanings. The definition of an untangled latent space as presented by Wilke et al. (2022), is that the variation of each latent variable is independent of the variation in the other latent variables. This was seen to be very rarely the case for the learned representations, and different regularization techniques can be applied to try and achieve this. It is important to note that circular manifolds, by this definition will always have entangled latent spaces, as at least two dependent variables are required to capture the circle. The other definition of an untangled latent space will be one where the learned manifold does not intersect itself. From a reconstruction point, the second is far more crucial to be achieved than the first. Accurate reconstructions can be achieved when the latent variables are correlated, but intersections in the learned manifold will always lead to poor reconstructions. If the latent space dimensionality is sufficiently large, it becomes easier for the VAE to project variations in the signal to different regions in the latent space, without overlapping with other representations.

Appendix B PCA variance fitting derivation

For the PCA-C-LR and PCA-Z-LR models, the log-variances of the reconstruction differences, $\log(\boldsymbol{\sigma}_{\boldsymbol{e}}^2)$, are learned by mapping the inputs ($\boldsymbol{c}$ or $\boldsymbol{z}$) to $\log\left((\boldsymbol{E}-\boldsymbol{\mu}_{\boldsymbol{e}})^2\right)+1.27$. In this appendix, Eq. (3-22) is derived:

$$\log(\boldsymbol{\sigma}_{\boldsymbol{e}}^2) \approx \mathbb{E}\left(\log\left((\boldsymbol{e}-\boldsymbol{\mu}_{\boldsymbol{e}})^2\right)\right)+1.27$$

Consider a variable X, sampled from a univariate zero-mean Gaussian with unit variance, $X \sim \mathcal{N}(0,1^2)$. Squaring the variable, and then taking the logarithm, makes the variable a log chi-squared variable.

The expected value of a log chi-squared variable is given by Lee (2012) as:

$$\begin{aligned}\mathbb{E}(\log(X^2)) &= \log 2+\psi(1/2)\\ &= \log 2-2\log 2-\gamma\\ &= -\log 2-\gamma\\ &\approx -1.27\end{aligned} \tag{B-1}$$

where $\psi(.)$ is the digamma function, and γ is the Euler–Mascheroni constant (≈ 0.5772).

We are interested in finding the expected value $\mathbb{E}(\log(A^2))$, where $A \sim \mathcal{N}(0,\sigma_A^2)$.

$$\begin{aligned}\mathbb{E}(\log(A^2)) &= \mathbb{E}(\log((\sigma_A X)^2))\\ &= \mathbb{E}(\log(\sigma_A^2)+\log(X^2))\\ &= \mathbb{E}(\log(\sigma_A^2))+\mathbb{E}(\log(X^2))\\ &= \log(\sigma_A^2)-\log 2-\gamma\end{aligned} \tag{B-2}$$

Rearranging the terms gives the equation used to fit the log-variance of the reconstruction differences in the PCA-C-LR and PCA-Z-LR models:

$$\begin{aligned}\log(\sigma_A^2) &= \mathbb{E}(\log(A^2))+\log 2+\gamma\\ &= \frac{1}{N}\sum_i^N \log(A_i^2)+\log 2+\gamma \qquad \text{(B-3)}\\ &\approx \frac{1}{N}\sum_i^N \log(A_i^2)+1.27 \qquad \text{(B-4)}\end{aligned}$$

Appendix C Pre-processing parameters, model architectures and optimization

The pre-processing parameters L_w, L_{sft} and f_s for both the phenomenological model dataset and the C-AIM gearbox dataset, are listed in Table C.1. For the SO pre-processing, L_w and L_{sft} were chosen to be the same as for the OTA and OT pre-processing methods. The signals were then re-sampled at rates that would cause the SO-1-revolution L_w to correspond to approximately 1 revolution at the average shaft speed, and the SO-1-tooth L_w to approximately 1 gear tooth meshing period at the average shaft speed. This makes the SO pre-processing more comparable with the OTA and OT techniques.

Table C.1: Pre-processing parameters: L_w, L_{sft}, f_s

Pre-processing	**Phenomenological dataset**			**C-AIM dataset**		
	L_w	L_{sft}	f_s	L_w	L_{sft}	f_s
OTA-1-rev	2560	640	2560 spr	9472	9472	9472 spr
OTA-1-tooth	128	32	2560 spr	256	256	9472 spr
OT-1-rev	2560	640	2560 spr	-	-	-
OT-1-tooth	128	32	2560 spr	-	-	-
SO-1-rev	2560	640	25000 Hz	9472	9472	18944 Hz
SO-1-tooth	128	32	25000 Hz	256	256	18944 Hz

The PCA latent space sizes are listed in Table C.2. The latent space sizes were chosen based on the methods discussed in Section 3.6. The VAE and iVAE models all had a latent space of 10 for the 'S'-models, and 100 for the 'L'-models.

Table C.2: PCA Latent space sizes

Pre-processing	**Phenomenological dataset**		**C-AIM dataset (unfiltered)**		**C-AIM dataset (lowpass)**	
	PCA-S	**PCA-L**	**PCA-S**	**PCA-L**	**PCA-S**	**PCA-L**
OTA-1-rev	1	224	23	1426	35	785
OTA-1-tooth	2	84	7	190	19	43
OT-1-rev	1	398	-	-	-	-
OT-1-tooth	4	55	-	-	-	-
SO-1-rev	22	565	19	1280	34	842
SO-1-tooth	5	98	8	166	17	50

All nonlinear models were built and trained using TensorFlow (Abadi et al. 2015). The same model architectures were used for all cases, with the VAE, iVAE and C-decoder architectures being functions only of the window length L_w, the latent dimensionality $D_{\boldsymbol{z}}$, and the operating condition variable dimensionality $D_{\boldsymbol{c}}$.

All convolutional layers have a fixed stride length of 4, kernel size of 32 and padding of 14 (to ensure that the feature map dimensionality can be divided by 4 as it is passed to the next layer). These are the same convolutional layers that Balshaw (2020) used.

The architecture of the VAE is summarized in Table C.3. The C-decoder has the same architecture as the VAE decoder network, only with input dimensionality $D_{\boldsymbol{c}}$. The iVAE encoder and decoder architectures are the same as for the VAE, with the only difference being that $\boldsymbol{c}$ is concatenated to layer nr. 8 in the encoder, before it is connected to layer nr. 9. The iVAE prior network architecture is given in Table C.4.

Table C.3: VAE architecture

Encoder		
Layer nr	**Description**	**Output dimensionality**
Input	Preprocessing: z-score normalisation over all neurons	$N \times 1 \times L_w$
1	1D convolution, ReLU activation	$N \times 32 \times \frac{L_w}{4}$
2	Batch Normalisation	$N \times 32 \times \frac{L_w}{4}$
3	1D convolution, ReLU activation	$N \times 32 \times \frac{L_w}{16}$
4	Batch Normalisation	$N \times 32 \times \frac{L_w}{16}$
5	1D convolution, ReLU activation	$N \times 32 \times \frac{L_w}{64}$
6	Batch Normalisation	$N \times 32 \times \frac{L_w}{64}$
7	Flatten	$N \times \frac{L_w}{2}$
8	Fully Connected, ReLU activation	$N \times 100$
9	Fully Connected, $\mu_{\boldsymbol{z}\vert\boldsymbol{x}}$ and $\log \sigma^2_{\boldsymbol{z}\vert\boldsymbol{x}}$ linear activation	$N \times 2 \cdot D_{\boldsymbol{z}}$
Decoder		
Layer nr	**Description**	**Output dimensionality**
Input	Sampled using reparameterization trick	$N \times D_{\boldsymbol{z}}$
1	Fully Connected, ReLU activation	$N \times 100$
2	Batch Normalisation	$N \times 100$
3	Fully Connected, ReLU activation	$N \times \frac{L_w}{2}$
4	Batch Normalisation	$N \times \frac{L_w}{2}$
5	Reshape	$N \times 32 \times \frac{L_w}{64}$
6	1D Deconvolution, ReLU activation	$N \times 32 \times \frac{L_w}{16}$
7	Batch Normalisation	$N \times 32 \times \frac{L_w}{16}$
8	1D Deconvolution, ReLU activation	$N \times 32 \times \frac{L_w}{4}$
9	Batch Normalisation	$N \times 32 \times \frac{L_w}{4}$
10	1D Deconvolution, $\mu_{\boldsymbol{x}\vert\boldsymbol{z}}$ and $\log \sigma^2_{\boldsymbol{x}\vert\boldsymbol{z}}$ linear activation	$N \times 2 \times L_w$

Table C.4: iVAE Prior architecture

Layer nr	**Description**	**Output dimensionality**
Input	Preprocessing: z-score normalisation over each seperate neuron	$N \times D_{\boldsymbol{c}}$
1	Fully Connected, ReLU activation	$N \times 32$
2	Fully Connected, ReLU activation	$N \times 32$
3	Fully Connected, $\mu_{\boldsymbol{z}\vert\boldsymbol{c}}$ and $\log \sigma^2_{\boldsymbol{z}\vert\boldsymbol{c}}$ linear activation	$N \times 2 \cdot D_{\boldsymbol{z}}$

All networks were trained using the Adam training algorithm (Kingma & Ba 2015), with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Appendix D Phenomenological Model Parameters

Similar model parameters to the parameters from the work by Balshaw (2020), were used to generate the phenomenological model dataset.

Quadratic amplitude modulation with respect to the shaft speed is used for all components that are modulated:

$$M_i(t) = \omega^2_{shaft}(t) \tag{D-1}$$

The gearbox characteristics are summarized in Table D.1. The variances of the white noise and random gear components are listed in Table D.2. The natural frequencies and damping ratios of the impulse responses in Eq. (4-5) are given in Table D.3. The amplitude and phase of the deterministic and random gear components are shown in Table D.4. The SNR used to determine the scaling parameters for each of the components relative to the noise term at a shaft speed of 10 Hz, are shown in Table D.5.

Table D.1: Phenomenological model gearbox characteristics

Characteristic	**Value**
Bearing Pitch Diameter	6.35 mm
Bearing Roller element diameter	36 mm
Bearing Contact angle	0°
Bearing Number of rolling elements	10
Gear teeth	20
Gear ratio	1
Sampling frequency	25 kHz
Sampling duration	0.5 s
Gear Mesh frequency	20 Hz/f_{shaft}
Ball Pass Frequency Outer race (BPFO)	4.12 Hz/f_{shaft}
Ball Pass Frequency Inner race (BPFI)	5.88 Hz/f_{shaft}
Ball Spin Frequency (BSF)	2.75 Hz/f_{shaft}
Fundamental Train Frequency (FTF)	0.41 Hz/f_{shaft}

Table D.2: Variance of the white noise and random gear components

Variance Component	**Value**
σ^2_n	0.01
σ^2_{gr}	1

Table D.3: Impulse response coefficients as shown in Eq.(4-5)

Component	**Natural Frequency $f_{n.i}$ (Hz)**	**Damping ratio ζ_i**
$h_{gd}(t)$	2000	0.05
$h_{gr}(t)$	1300	0.05
$h_{bi}(t)$	7000	0.05
$h_{bo}(t)$	7000	0.05

Table D.4: Amplitude and phase of deterministic and random gear component, as shown in Eqs. (4-6) and (4-7)

k	1	2
Amplitude $A_i^{(k)}$	2	3
Phase $\phi_i^{(k)}$	0	0

Table D.5: SNR used to determine scaling parameters in Eq.(4-14)

Component Signal-to-Noise Ratio	**Value (dB)**
Deterministic gear	5
Random gear	-10
Bearing fault (inner and outer)	-30 to 10, linearly spaced

Appendix E Phenomenological model dataset results tables

Table E.1: Phenomenological model dataset analysis results: OTA-1-revolution pre-processing (green: top 10 %, red: below average)

Outer race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.22	0.35	0.59	0.52	31	31	15	23	0.65	0.72	0.80	0.79
PPCA-L	Recon	0.23	0.36	0.62	0.53	31	28	15	21	0.66	0.73	0.81	0.78
PCA-Z-LR-S	Recon	0.43	0.53	0.59	0.54	21	17	15	18	0.71	0.77	0.79	0.71
PCA-Z-LR-L	Recon	0.15	0.00	0.03	0.00	31	31	31	31	0.64	0.59	0.37	0.43
VAE-S	Recon	0.19	0.32	0.54	0.41	31	31	21	26	0.61	0.73	0.76	0.72
VAE-L	Recon	0.40	0.57	0.61	0.45	24	18	15	19	0.72	0.78	0.78	0.79
C-decoder	Recon	0.55	0.60	0.62	0.56	21	18	14	18	0.79	0.81	0.79	0.73
	Recon	0.53	0.58	0.58	0.55	20	15	15	16	0.80	0.81	0.78	0.71
PCA-C-LR-S	Latent	0.02	0.10	0.29	0.02	31	31	31	31	0.57	0.59	0.66	0.41
	Recon+Latent	0.54	0.62	0.70	0.56	20	15	15	16				
	Recon	0.55	0.61	0.60	0.56	19	14	15	16	0.79	0.80	0.81	0.72
PCA-C-LR-L	Latent	0.26	0.18	0.05	0.08	26	31	31	31	0.58	0.69	0.65	0.49
	Recon+Latent	0.58	0.63	0.62	0.56	18	14	15	16				
	Recon	0.44	0.54	0.61	0.56	24	18	14	18	0.76	0.81	0.78	0.76
iVAE-S	Latent	0.31	0.21	0.09	0.13	26	27	31	29	0.69	0.59	0.62	0.43
	Recon+Latent	0.46	0.54	0.65	0.56	24	18	14	18				
	Recon	0.41	0.54	0.60	0.45	23	17	15	19	0.70	0.75	0.81	0.78
iVAE-L	Latent	0.23	0.28	0.08	0.19	31	31	31	31	0.65	0.70	0.60	0.57
	Recon+Latent	0.50	0.59	0.65	0.48	23	17	15	19				
Inner race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.20	0.37	0.66	0.41	31	29	13	25	0.62	0.71	0.84	0.72
PPCA-L	Recon	0.21	0.38	0.70	0.43	31	29	12	24	0.63	0.72	0.86	0.73
PCA-Z-LR-S	Recon	0.44	0.57	0.66	0.48	22	17	13	19	0.72	0.79	0.82	0.67
PCA-Z-LR-L	Recon	0.14	0.01	0.04	0.00	31	31	31	31	0.62	0.58	0.37	0.39
VAE-S	Recon	0.18	0.32	0.52	0.42	31	31	20	25	0.58	0.70	0.73	0.73
VAE-L	Recon	0.36	0.52	0.64	0.45	28	18	14	20	0.68	0.76	0.82	0.73
C-decoder	Recon	0.46	0.56	0.69	0.47	22	17	13	19	0.74	0.80	0.83	0.71
	Recon	0.51	0.62	0.64	0.47	19	15	13	19	0.78	0.82	0.81	0.68
PCA-C-LR-S	Latent	0.01	0.08	0.31	0.00	31	31	31	31	0.57	0.59	0.67	0.30
	Recon+Latent	0.51	0.65	0.75	0.47	19	15	13	19				
	Recon	0.53	0.64	0.63	0.50	19	15	13	18	0.77	0.81	0.85	0.70
PCA-C-LR-L	Latent	0.24	0.28	0.15	0.17	27	28	31	28	0.60	0.69	0.69	0.51
	Recon+Latent	0.56	0.64	0.64	0.50	19	15	13	18				
	Recon	0.39	0.59	0.67	0.48	25	18	13	19	0.73	0.80	0.83	0.72
iVAE-S	Latent	0.30	0.18	0.09	0.03	28	31	31	31	0.68	0.62	0.61	0.39
	Recon+Latent	0.44	0.59	0.69	0.48	25	18	13	19				
	Recon	0.33	0.52	0.68	0.46	25	18	14	20	0.68	0.76	0.84	0.72
iVAE-L	Latent	0.22	0.30	0.06	0.07	31	31	31	31	0.62	0.72	0.57	0.44
	Recon+Latent	0.39	0.57	0.70	0.46	24	18	14	20				

Table E.2: Phenomenological model dataset analysis results: OTA-1-tooth pre-processing (green: top 10 %, red: below average)

Outer race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.22	0.36	0.61	0.49	31	28	15	21	0.65	0.73	0.80	0.77
PPCA-L	Recon	0.29	0.45	0.59	0.52	31	24	14	18	0.69	0.77	0.83	0.79
PCA-Z-LR-S	Recon	0.39	0.57	0.59	0.50	23	18	15	18	0.68	0.76	0.82	0.70
PCA-Z-LR-L	Recon	0.49	0.49	0.52	0.46	21	21	19	23	0.76	0.79	0.82	0.69
VAE-S	Recon	0.38	0.54	0.62	0.48	26	18	15	19	0.76	0.82	0.81	0.73
VAE-L	Recon	0.54	0.59	0.63	0.47	21	18	14	19	0.79	0.84	0.82	0.70
C-decoder	Recon	0.47	0.58	0.61	0.54	19	16	15	18	0.74	0.77	0.81	0.72
	Recon	0.48	0.57	0.60	0.50	18	15	15	18	0.74	0.79	0.80	0.73
PCA-C-LR-S	Latent	0.11	0.07	0.11	0.05	31	31	31	31	0.60	0.63	0.58	0.39
	Recon+Latent	0.50	0.59	0.63	0.50	18	15	15	18				
	Recon	0.45	0.47	0.52	0.49	21	18	17	20	0.73	0.80	0.83	0.68
PCA-C-LR-L	Latent	0.53	0.50	0.02	0.54	21	21	31	20	0.77	0.77	0.57	0.74
	Recon+Latent	0.60	0.54	0.53	0.59	18	18	17	15				
	Recon	0.41	0.54	0.63	0.47	21	16	14	19	0.70	0.76	0.82	0.70
iVAE-S	Latent	0.43	0.44	0.04	0.40	23	22	31	23	0.69	0.71	0.61	0.55
	Recon+Latent	0.47	0.56	0.64	0.48	20	16	14	19				
	Recon	0.50	0.59	0.61	0.47	20	15	15	19	0.76	0.80	0.81	0.71
iVAE-L	Latent	0.44	0.44	0.12	0.41	23	21	31	21	0.77	0.79	0.63	0.65
	Recon+Latent	0.53	0.60	0.65	0.48	20	15	15	19				
Inner race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.20	0.37	0.68	0.43	31	29	12	24	0.62	0.71	0.84	0.72
PPCA-L	Recon	0.26	0.45	0.67	0.50	31	25	13	20	0.66	0.75	0.86	0.75
PCA-Z-LR-S	Recon	0.39	0.61	0.66	0.49	25	17	13	19	0.69	0.77	0.84	0.67
PCA-Z-LR-L	Recon	0.53	0.52	0.61	0.50	25	21	18	22	0.77	0.80	0.85	0.68
VAE-S	Recon	0.34	0.56	0.68	0.47	28	18	12	20	0.72	0.81	0.85	0.66
VAE-L	Recon	0.49	0.62	0.69	0.46	22	17	12	20	0.76	0.83	0.86	0.65
C-decoder	Recon	0.43	0.58	0.70	0.48	21	15	13	20	0.72	0.79	0.84	0.68
	Recon	0.46	0.60	0.65	0.50	20	15	13	19	0.73	0.79	0.83	0.68
PCA-C-LR-S	Latent	0.10	0.11	0.16	0.04	31	31	30	31	0.57	0.60	0.59	0.35
	Recon+Latent	0.48	0.61	0.69	0.50	20	15	13	19				
	Recon	0.49	0.49	0.61	0.46	22	19	16	20	0.73	0.80	0.85	0.68
PCA-C-LR-L	Latent	0.52	0.54	0.50	0.52	22	20	22	22	0.76	0.79	0.78	0.69
	Recon+Latent	0.60	0.57	0.63	0.58	20	17	15	16				
	Recon	0.37	0.56	0.69	0.48	22	16	12	20	0.69	0.78	0.86	0.65
iVAE-S	Latent	0.48	0.49	0.41	0.41	22	22	31	22	0.71	0.75	0.77	0.57
	Recon+Latent	0.50	0.59	0.69	0.49	22	16	12	20				
	Recon	0.45	0.59	0.69	0.47	22	16	12	19	0.73	0.81	0.85	0.67
iVAE-L	Latent	0.46	0.47	0.48	0.45	23	22	24	22	0.77	0.81	0.81	0.68
	Recon+Latent	0.50	0.60	0.72	0.49	22	16	12	19				

Table E.3: Phenomenological model dataset analysis results: OT-1-revolution pre-processing (green: top 10 %, red: below average)

Outer race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.22	0.35	0.60	0.50	31	31	15	23	0.65	0.72	0.80	0.79
PPCA-L	Recon	0.24	0.36	0.62	0.53	31	27	14	21	0.66	0.73	0.81	0.78
PCA-Z-LR-S	Recon	0.43	0.57	0.61	0.54	22	18	14	18	0.69	0.76	0.78	0.70
PCA-Z-LR-L	Recon	0.29	0.14	0.13	0.07	31	31	31	31	0.70	0.68	0.40	0.65
VAE-S	Recon	0.20	0.32	0.52	0.42	31	31	21	26	0.62	0.73	0.79	0.74
VAE-L	Recon	0.36	0.52	0.60	0.47	26	19	15	19	0.75	0.81	0.79	0.79
C-decoder	Recon	0.54	0.56	0.61	0.53	21	18	14	18	0.78	0.81	0.79	0.73
	Recon	0.52	0.59	0.61	0.55	19	15	15	18	0.79	0.82	0.79	0.72
PCA-C-LR-S	Latent	0.01	0.13	0.24	0.02	31	31	31	31	0.57	0.59	0.67	0.41
	Recon+Latent	0.52	0.64	0.70	0.55	19	15	15	18				
	Recon	0.50	0.60	0.64	0.58	18	14	15	16	0.74	0.78	0.81	0.71
PCA-C-LR-L	Latent	0.42	0.18	0.09	0.14	22	29	31	30	0.75	0.72	0.62	0.48
	Recon+Latent	0.51	0.61	0.66	0.59	18	14	15	16				
	Recon	0.42	0.53	0.60	0.50	21	17	15	18	0.70	0.76	0.78	0.76
iVAE-S	Latent	0.32	0.35	0.06	0.24	27	27	31	29	0.62	0.67	0.53	0.55
	Recon+Latent	0.45	0.57	0.64	0.51	21	17	15	18				
	Recon	0.45	0.56	0.60	0.50	24	18	15	19	0.79	0.82	0.79	0.78
iVAE-L	Latent	0.24	0.21	0.08	0.17	28	30	31	31	0.61	0.58	0.61	0.44
	Recon+Latent	0.47	0.57	0.64	0.51	22	18	15	19				
Inner race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.20	0.37	0.66	0.41	31	29	13	25	0.62	0.71	0.84	0.72
PPCA-L	Recon	0.21	0.39	0.70	0.43	31	28	12	24	0.63	0.72	0.85	0.73
PCA-Z-LR-S	Recon	0.45	0.60	0.69	0.49	22	17	12	19	0.70	0.78	0.83	0.68
PCA-Z-LR-L	Recon	0.27	0.10	0.13	0.05	31	31	31	31	0.68	0.66	0.40	0.62
VAE-S	Recon	0.20	0.33	0.55	0.40	31	31	20	26	0.59	0.71	0.79	0.68
VAE-L	Recon	0.32	0.53	0.63	0.48	28	19	13	19	0.72	0.80	0.82	0.69
C-decoder	Recon	0.47	0.57	0.69	0.48	21	17	13	19	0.74	0.81	0.84	0.68
	Recon	0.49	0.61	0.68	0.48	20	15	13	19	0.77	0.83	0.83	0.69
PCA-C-LR-S	Latent	0.01	0.11	0.26	0.00	31	31	31	31	0.57	0.59	0.68	0.29
	Recon+Latent	0.49	0.64	0.76	0.48	20	15	13	19				
	Recon	0.50	0.64	0.71	0.52	18	15	13	17	0.74	0.79	0.86	0.68
PCA-C-LR-L	Latent	0.42	0.28	0.28	0.26	22	28	31	28	0.74	0.72	0.67	0.50
	Recon+Latent	0.50	0.65	0.75	0.53	18	15	13	17				
	Recon	0.38	0.55	0.66	0.49	25	16	13	19	0.72	0.79	0.82	0.68
iVAE-S	Latent	0.34	0.36	0.10	0.17	28	28	31	31	0.66	0.69	0.59	0.45
	Recon+Latent	0.43	0.59	0.68	0.49	23	16	13	19				
	Recon	0.43	0.58	0.66	0.49	25	18	13	19	0.77	0.84	0.83	0.70
iVAE-L	Latent	0.25	0.30	0.10	0.21	28	29	31	31	0.62	0.64	0.63	0.48
	Recon+Latent	0.45	0.59	0.69	0.51	25	18	13	19				

Table E.4: Phenomenological model dataset analysis results: OT-1-tooth pre-processing (green: top 10 %, red: below average)

Outer race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.23	0.37	0.62	0.47	31	28	15	22	0.65	0.73	0.80	0.77
PPCA-L	Recon	0.29	0.44	0.61	0.57	31	26	15	19	0.68	0.76	0.83	0.80
PCA-Z-LR-S	Recon	0.33	0.50	0.63	0.47	31	24	15	21	0.69	0.76	0.84	0.75
PCA-Z-LR-L	Recon	0.46	0.55	0.60	0.52	31	21	16	22	0.71	0.78	0.83	0.74
VAE-S	Recon	0.34	0.54	0.62	0.50	27	19	15	19	0.75	0.82	0.81	0.71
VAE-L	Recon	0.47	0.58	0.63	0.49	21	18	14	18	0.77	0.82	0.82	0.71
C-decoder	Recon	0.51	0.50	0.52	0.41	23	18	18	20	0.75	0.71	0.58	0.66
	Recon	0.47	0.57	0.61	0.52	18	15	15	18	0.73	0.78	0.82	0.73
PCA-C-LR-S	Latent	0.07	0.08	0.07	0.02	31	31	31	31	0.48	0.58	0.63	0.49
	Recon+Latent	0.48	0.60	0.63	0.53	18	15	15	18				
	Recon	0.61	0.61	0.59	0.59	18	18	15	16	0.82	0.85	0.84	0.75
PCA-C-LR-L	Latent	0.41	0.34	0.03	0.34	23	24	31	24	0.68	0.68	0.52	0.62
	Recon+Latent	0.63	0.62	0.61	0.59	18	18	15	16				
	Recon	0.54	0.58	0.61	0.48	21	18	15	18	0.79	0.83	0.81	0.71
iVAE-S	Latent	0.38	0.30	0.11	0.27	25	24	31	25	0.59	0.44	0.37	0.52
	Recon+Latent	0.61	0.59	0.67	0.49	20	16	15	18				
	Recon	0.53	0.58	0.63	0.49	20	15	15	18	0.78	0.81	0.81	0.72
iVAE-L	Latent	0.37	0.40	0.07	0.32	24	23	31	23	0.63	0.71	0.67	0.62
	Recon+Latent	0.56	0.61	0.66	0.49	20	15	15	18				
Inner race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.21	0.38	0.69	0.43	31	29	12	24	0.62	0.72	0.84	0.72
PPCA-L	Recon	0.26	0.47	0.69	0.49	31	26	13	22	0.65	0.75	0.86	0.77
PCA-Z-LR-S	Recon	0.26	0.46	0.69	0.45	31	25	13	22	0.64	0.73	0.86	0.71
PCA-Z-LR-L	Recon	0.39	0.51	0.67	0.49	31	22	15	20	0.68	0.76	0.87	0.75
VAE-S	Recon	0.33	0.56	0.68	0.45	31	18	12	20	0.71	0.81	0.85	0.66
VAE-L	Recon	0.40	0.57	0.69	0.47	24	17	12	19	0.74	0.83	0.86	0.64
C-decoder	Recon	0.45	0.52	0.58	0.42	25	18	17	20	0.73	0.73	0.61	0.60
	Recon	0.46	0.60	0.67	0.48	20	15	13	19	0.72	0.79	0.84	0.68
PCA-C-LR-S	Latent	0.07	0.10	0.10	0.03	31	31	31	31	0.47	0.58	0.65	0.32
	Recon+Latent	0.48	0.62	0.69	0.48	20	15	13	19				
	Recon	0.60	0.64	0.65	0.56	20	17	13	16	0.81	0.86	0.86	0.74
PCA-C-LR-L	Latent	0.39	0.37	0.35	0.38	22	23	26	24	0.67	0.71	0.70	0.63
	Recon+Latent	0.61	0.65	0.67	0.57	20	17	13	16				
	Recon	0.48	0.59	0.68	0.46	22	17	12	20	0.76	0.82	0.85	0.69
iVAE-S	Latent	0.39	0.33	0.08	0.28	26	24	31	25	0.63	0.48	0.33	0.56
	Recon+Latent	0.58	0.61	0.72	0.47	21	17	12	20				
	Recon	0.47	0.58	0.68	0.49	22	16	12	19	0.74	0.81	0.85	0.68
iVAE-L	Latent	0.38	0.46	0.38	0.38	24	22	31	22	0.67	0.74	0.78	0.60
	Recon+Latent	0.51	0.63	0.71	0.49	22	16	12	19				

Table E.5: Phenomenological model dataset analysis results: SO-1-revolution pre-processing (green: top 10 %, red: below average)

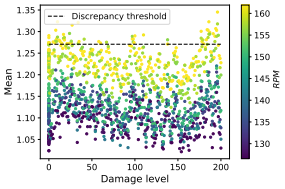
Outer race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.13	0.29	0.50	0.51	31	31	18	22	0.64	0.70	0.63	0.79
PPCA-L	Recon	0.26	0.38	0.63	0.59	31	27	13	20	0.67	0.73	0.83	0.82
PCA-Z-LR-S	Recon	0.11	0.11	0.12	0.20	31	31	31	31	0.66	0.70	0.69	0.76
PCA-Z-LR-L	Recon	0.39	0.34	0.01	0.27	31	31	31	31	0.73	0.72	0.28	0.70
VAE-S	Recon	0.13	0.35	0.60	0.59	31	24	15	18	0.55	0.71	0.78	0.79
VAE-L	Recon	0.13	0.46	0.58	0.58	31	22	15	17	0.52	0.71	0.78	0.79
C-decoder	Recon	0.45	0.55	0.60	0.55	26	21	15	18	0.75	0.78	0.79	0.75
	Recon	0.25	0.42	0.49	0.45	28	23	18	21	0.66	0.75	0.66	0.78
PCA-C-LR-S	Latent	0.07	0.05	0.04	0.07	31	31	31	31	0.43	0.40	0.49	0.42
	Recon+Latent	0.32	0.46	0.51	0.49	28	23	18	21				
	Recon	0.60	0.65	0.61	0.63	18	15	15	14	0.84	0.85	0.82	0.82
PCA-C-LR-L	Latent	0.30	0.23	0.04	0.16	27	28	31	30	0.67	0.60	0.54	0.56
	Recon+Latent	0.62	0.67	0.62	0.65	16	15	15	14				
	Recon	0.15	0.42	0.63	0.54	31	21	15	18	0.52	0.73	0.79	0.79
iVAE-S	Latent	0.31	0.39	0.17	0.30	27	27	31	28	0.61	0.67	0.55	0.68
	Recon+Latent	0.34	0.49	0.69	0.58	27	21	15	18				
	Recon	0.13	0.44	0.59	0.55	31	21	15	17	0.44	0.69	0.78	0.77
iVAE-L	Latent	0.27	0.36	0.13	0.25	27	28	31	30	0.61	0.65	0.54	0.61
	Recon+Latent	0.34	0.49	0.64	0.58	27	21	15	17				
Inner race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.11	0.29	0.59	0.48	31	31	16	24	0.62	0.69	0.66	0.80
PPCA-L	Recon	0.24	0.40	0.70	0.63	31	28	11	20	0.64	0.72	0.87	0.82
PCA-Z-LR-S	Recon	0.11	0.11	0.11	0.15	31	31	31	31	0.63	0.68	0.73	0.72
PCA-Z-LR-L	Recon	0.33	0.29	0.02	0.17	31	31	31	31	0.70	0.69	0.29	0.68
VAE-S	Recon	0.16	0.38	0.66	0.42	31	25	14	21	0.56	0.74	0.82	0.73
VAE-L	Recon	0.14	0.49	0.67	0.45	31	22	14	20	0.49	0.72	0.81	0.74
C-decoder	Recon	0.43	0.60	0.64	0.38	27	20	14	22	0.74	0.80	0.81	0.71
	Recon	0.21	0.43	0.57	0.58	31	21	17	19	0.66	0.77	0.69	0.77
PCA-C-LR-S	Latent	0.05	0.05	0.06	0.05	31	31	31	31	0.43	0.45	0.50	0.47
	Recon+Latent	0.26	0.46	0.60	0.60	31	21	17	19				
	Recon	0.62	0.69	0.68	0.68	20	15	13	15	0.85	0.87	0.86	0.86
PCA-C-LR-L	Latent	0.34	0.30	0.04	0.16	25	24	31	31	0.67	0.67	0.55	0.63
	Recon+Latent	0.65	0.70	0.69	0.69	18	15	13	15				
	Recon	0.17	0.47	0.64	0.45	31	22	14	21	0.52	0.75	0.81	0.75
iVAE-S	Latent	0.40	0.44	0.15	0.49	28	27	31	27	0.68	0.72	0.54	0.75
	Recon+Latent	0.42	0.51	0.69	0.58	28	22	13	20				
	Recon	0.10	0.49	0.68	0.43	31	20	14	21	0.43	0.75	0.81	0.73
iVAE-L	Latent	0.37	0.40	0.13	0.34	24	28	31	28	0.68	0.72	0.55	0.73
	Recon+Latent	0.44	0.54	0.72	0.48	24	20	14	21				

Table E.6: Phenomenological model dataset analysis results: SO-1-tooth pre-processing (green: top 10 %, red: below average)

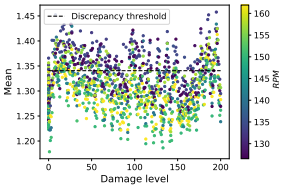
Outer race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.24	0.38	0.64	0.52	31	27	15	21	0.66	0.73	0.81	0.81
PPCA-L	Recon	0.24	0.39	0.68	0.62	31	27	13	20	0.66	0.74	0.86	0.84
PCA-Z-LR-S	Recon	0.40	0.52	0.65	0.59	31	24	15	20	0.70	0.77	0.84	0.81
PCA-Z-LR-L	Recon	0.03	0.09	0.60	0.52	31	31	31	29	0.53	0.67	0.83	0.79
VAE-S	Recon	0.47	0.59	0.69	0.59	31	21	13	18	0.75	0.81	0.85	0.82
VAE-L	Recon	0.58	0.61	0.65	0.60	23	18	13	15	0.81	0.84	0.84	0.81
C-decoder	Recon	0.40	0.48	0.60	0.54	24	19	15	18	0.68	0.72	0.78	0.78
	Recon	0.56	0.64	0.66	0.57	18	18	15	18	0.76	0.84	0.83	0.82
PCA-C-LR-S	Latent	0.09	0.04	0.07	0.10	31	31	31	31	0.50	0.47	0.46	0.49
	Recon+Latent	0.63	0.65	0.68	0.62	18	18	15	18				
	Recon	0.62	0.65	0.67	0.61	15	15	14	15	0.81	0.81	0.83	0.80
PCA-C-LR-L	Latent	0.59	0.63	0.31	0.65	15	14	31	14	0.81	0.86	0.54	0.82
	Recon+Latent	0.64	0.68	0.69	0.66	14	13	14	14				
	Recon	0.52	0.64	0.66	0.60	21	16	13	15	0.81	0.83	0.85	0.79
iVAE-S	Latent	0.13	0.29	0.26	0.27	31	26	27	31	0.55	0.74	0.61	0.74
	Recon+Latent	0.55	0.66	0.68	0.65	21	16	13	15				
	Recon	0.61	0.62	0.66	0.62	21	16	13	15	0.83	0.86	0.85	0.80
iVAE-L	Latent	0.24	0.32	0.05	0.35	25	23	31	24	0.54	0.67	0.53	0.65
	Recon+Latent	0.61	0.64	0.66	0.65	21	16	13	15				
Inner race fault													
Model	**Discrepancy space**	**TPR**				**DDL**				**AUC**			
		mean	var	kurt	OST	mean	var	kurt	OST	mean	var	kurt	OST
PPCA-S	Recon	0.21	0.40	0.68	0.56	31	28	14	22	0.63	0.72	0.82	0.80
PPCA-L	Recon	0.22	0.42	0.75	0.62	31	28	11	20	0.63	0.73	0.87	0.82
PCA-Z-LR-S	Recon	0.33	0.48	0.70	0.45	31	25	14	22	0.65	0.75	0.86	0.81
PCA-Z-LR-L	Recon	0.02	0.08	0.60	0.31	31	31	31	31	0.53	0.64	0.85	0.74
VAE-S	Recon	0.41	0.57	0.73	0.61	31	20	11	16	0.71	0.80	0.87	0.84
VAE-L	Recon	0.51	0.64	0.72	0.63	31	17	11	16	0.77	0.84	0.87	0.85
C-decoder	Recon	0.38	0.51	0.63	0.60	23	18	14	17	0.68	0.73	0.81	0.82
	Recon	0.58	0.66	0.68	0.66	19	17	14	16	0.78	0.84	0.85	0.83
PCA-C-LR-S	Latent	0.09	0.06	0.13	0.04	31	31	31	31	0.48	0.48	0.49	0.46
	Recon+Latent	0.61	0.68	0.70	0.68	18	17	14	16				
	Recon	0.64	0.71	0.72	0.65	16	13	13	15	0.83	0.85	0.86	0.80
PCA-C-LR-L	Latent	0.62	0.69	0.66	0.67	15	12	13	13	0.80	0.85	0.65	0.83
	Recon+Latent	0.68	0.75	0.73	0.72	15	11	12	13				
	Recon	0.48	0.65	0.71	0.62	20	15	11	16	0.77	0.83	0.87	0.86
iVAE-S	Latent	0.20	0.31	0.37	0.12	29	24	24	30	0.61	0.73	0.66	0.64
	Recon+Latent	0.51	0.65	0.71	0.62	20	15	11	16				
	Recon	0.57	0.66	0.71	0.67	20	15	11	16	0.80	0.86	0.86	0.87
iVAE-L	Latent	0.27	0.37	0.34	0.34	24	22	31	22	0.62	0.71	0.65	0.66
	Recon+Latent	0.57	0.66	0.71	0.68	20	15	11	16				

Appendix F C-AIM Gearbox Unfiltered Dataset Figures

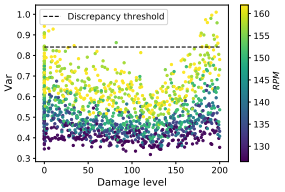
F.1 Homoscedastic vs. heteroscedastic models



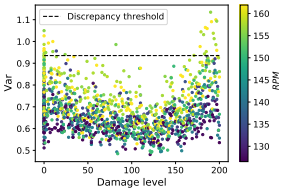
(a) PPCA: mean



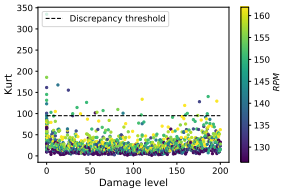
(b) PCA-C-LR: mean



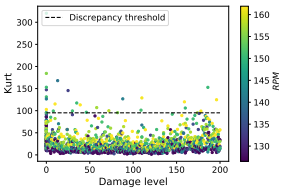
(c) PPCA: variance



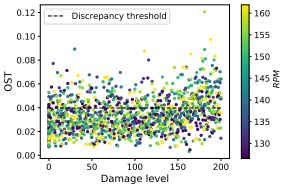
(d) PCA-C-LR: variance



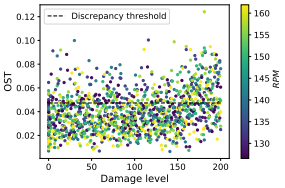
(e) PPCA: kurtosis



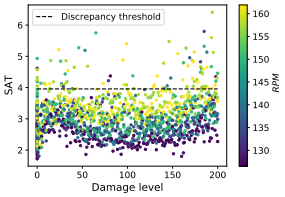
(f) PCA-C-LR: kurtosis



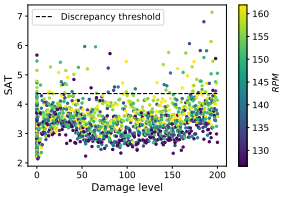
(g) PPCA: OST



(h) PCA-C-LR: OST



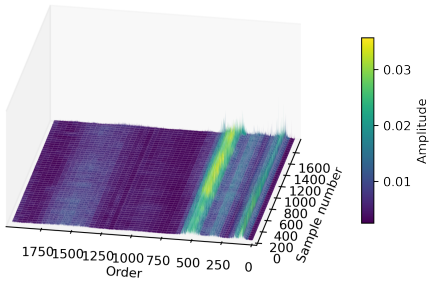
(i) PPCA: SAT



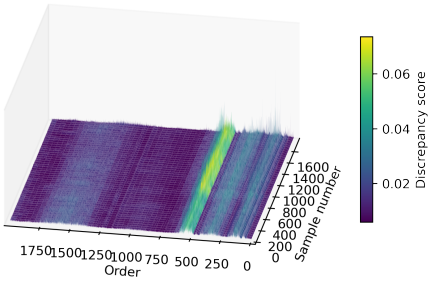
(j) PCA-C-LR: SAT

Figure F.1: PPCA-S vs PCA-C-LR-S: reconstruction space discrepancy signal features for fault detection and tracking (OTA-1-tooth, unfiltered dataset)

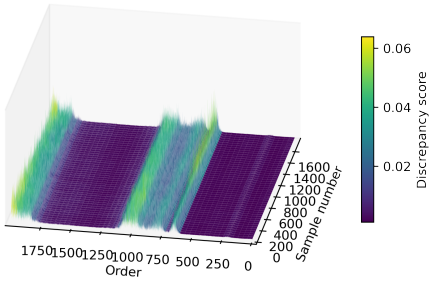
F.2 Linear PCA models



(a) Raw acceleration data

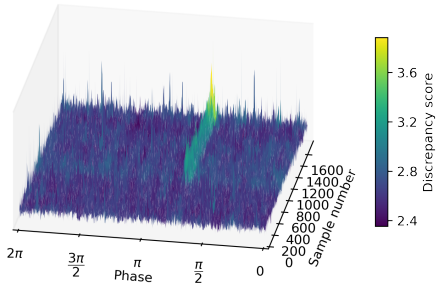


(b) S: Recon space

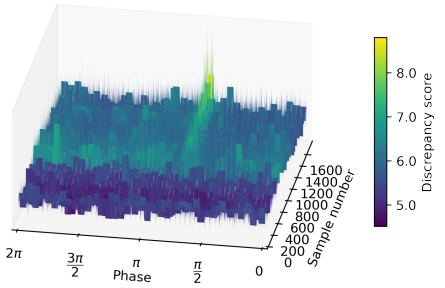


(c) L: Recon space

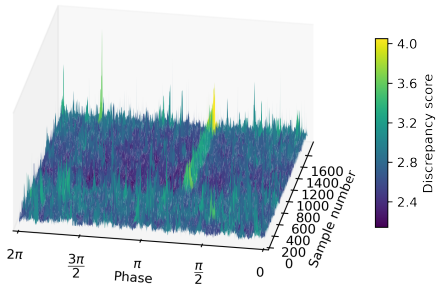
Figure F.2: PCA-C-LR-S vs PCA-C-LR-L: Order spectra of reconstruction space discrepancy signals before Hilbert transform is applied, compared to order spectra of raw acceleration data (OTA-1-tooth, unfiltered dataset)



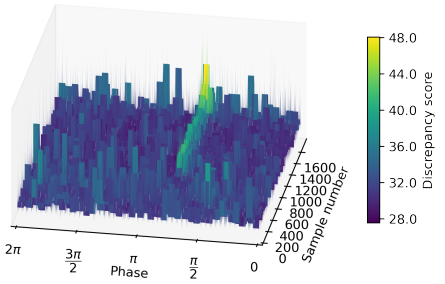
(a) S: Recon space



(b) S: Latent space



(c) L: Recon space



(d) L: Latent space

Figure F.3: PCA-C-LR-S vs PCA-C-LR-L: Synchronous average of reconstruction- and latent space discrepancy signals, as the fault develops (OTA-1-tooth, unfiltered dataset)

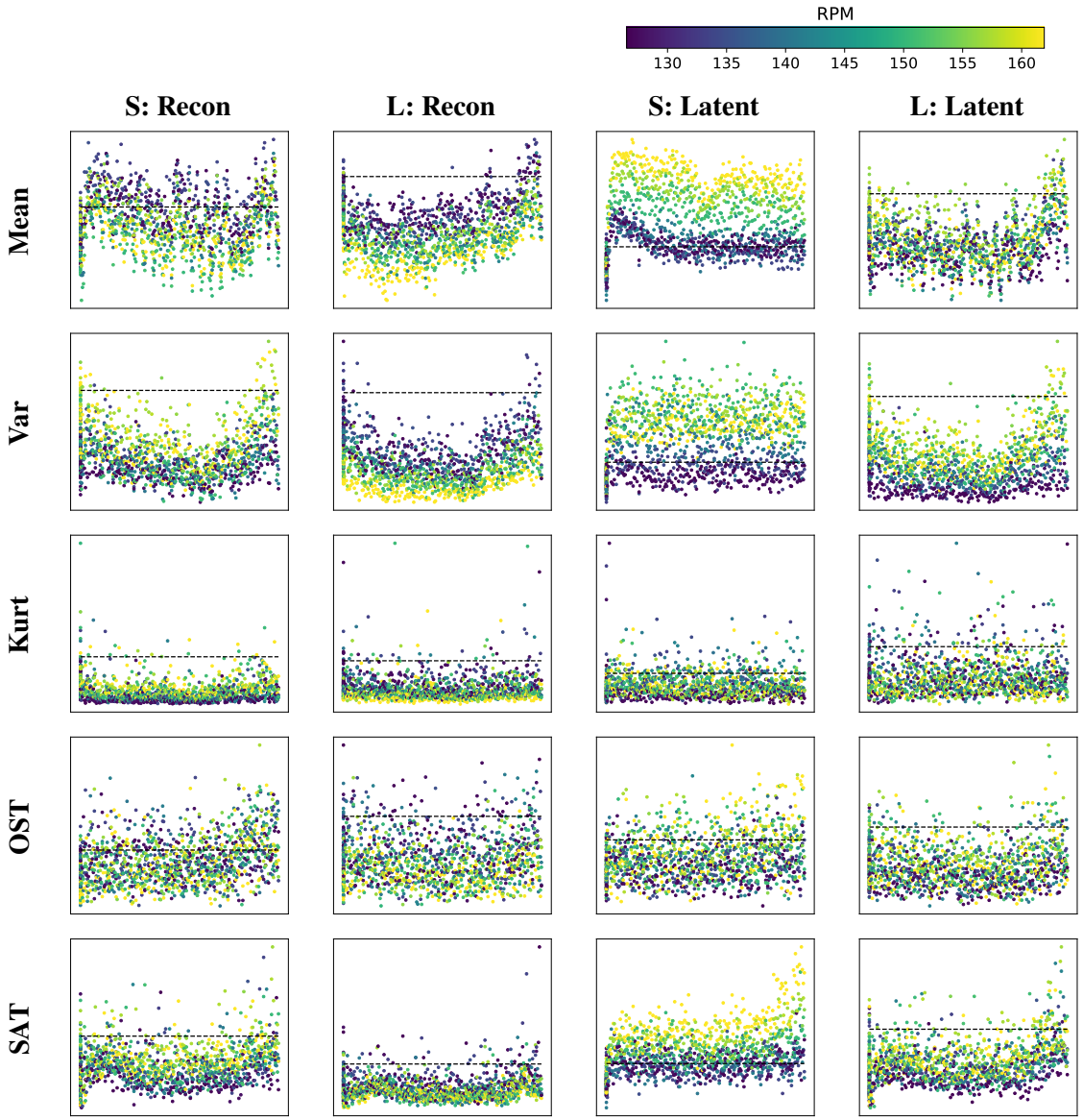
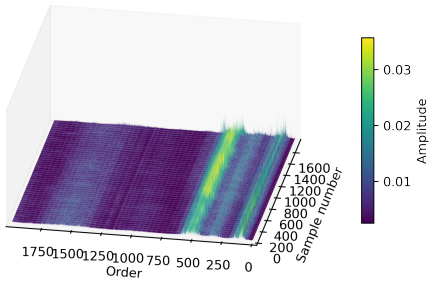
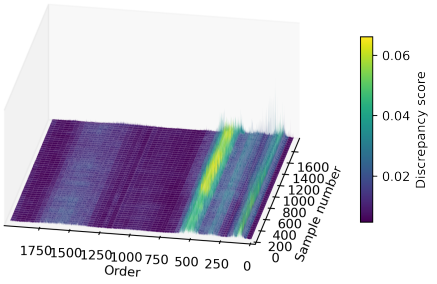


Figure F.4: PCA-C-LR-S vs PCA-C-LR-L: reconstruction space and latent space discrepancy signal features for fault detection and tracking (OTA-1-tooth, unfiltered dataset, x-axis: Damage level 0-200, y-axis: Feature value, dashed line: Discrepancy threshold)

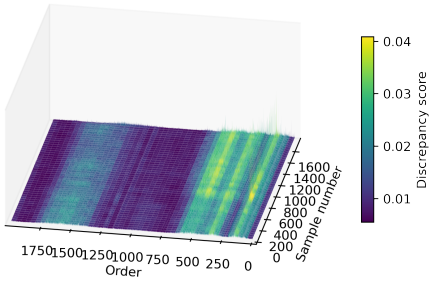
F.3 Nonlinear VAE models



(a) Raw acceleration data

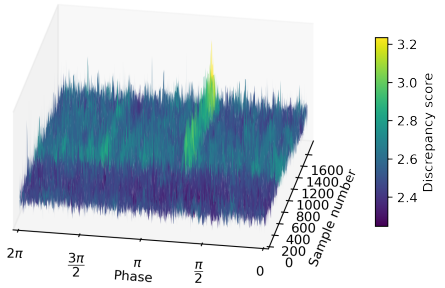


(b) S: Recon space

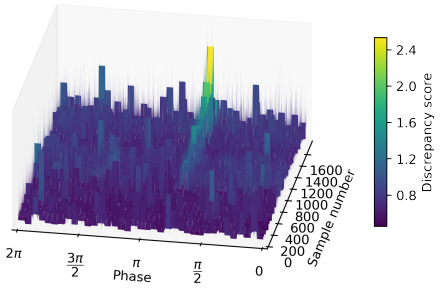


(c) L: Recon space

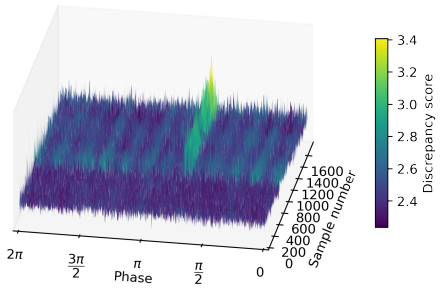
Figure F.5: iVAE-S vs iVAE-L: Order spectra of reconstruction space discrepancy signals before Hilbert transform is applied, compared to order spectra of raw acceleration data (OTA-1-tooth, unfiltered dataset)



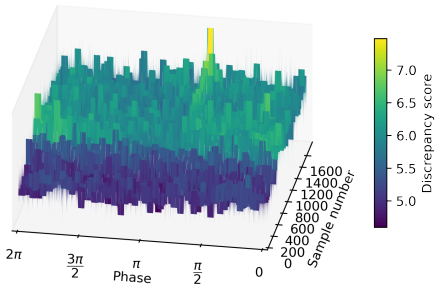
(a) S: Recon space



(b) S: Latent space



(c) L: Recon space



(d) L: Latent space

Figure F.6: iVAE-S vs iVAE-L: Synchronous average of reconstruction- and latent space discrepancy signals, as the fault develops (OTA-1-tooth, unfiltered dataset)

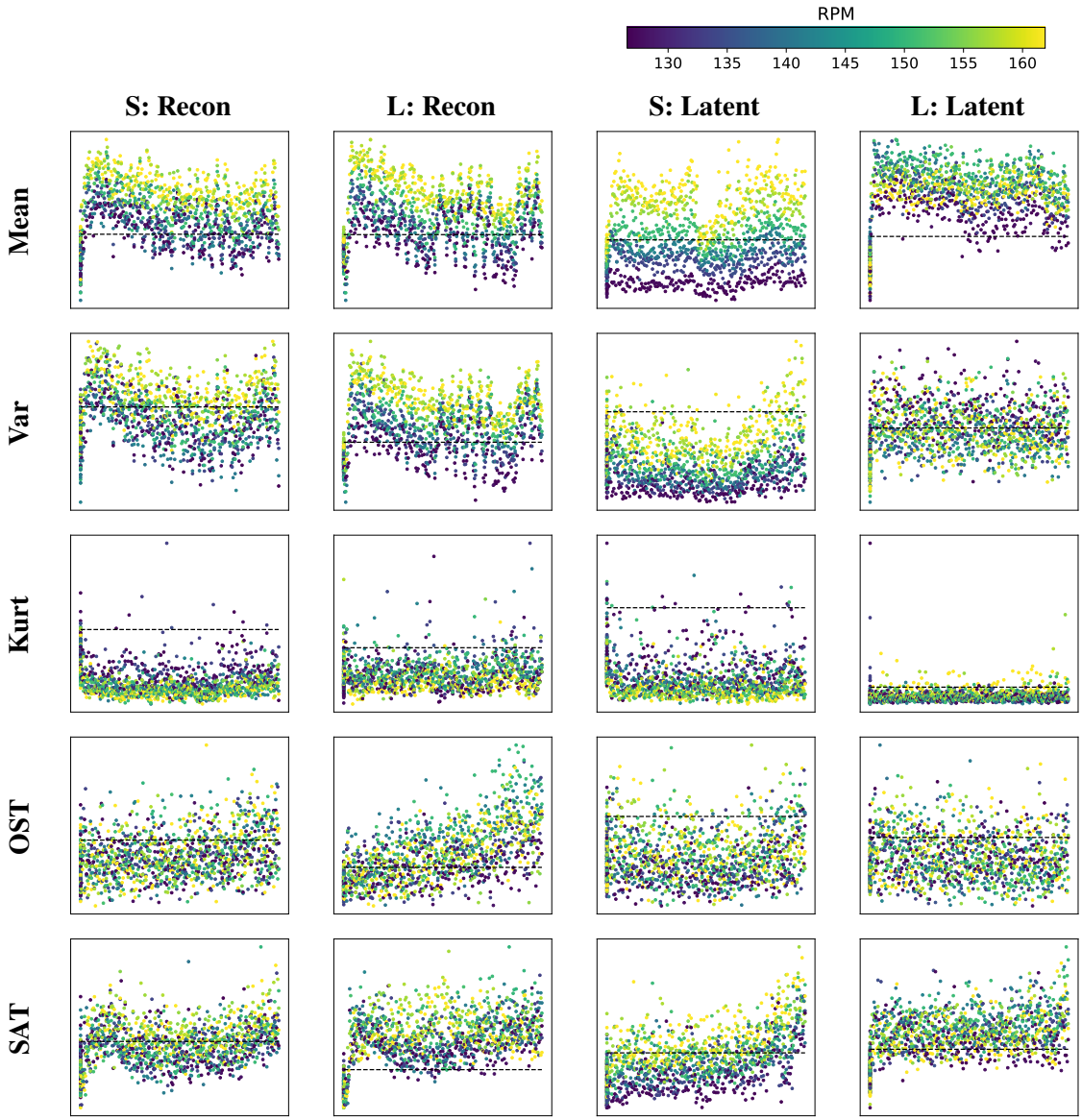


Figure F.7: iVAE-S vs iVAE-L: reconstruction space and latent space discrepancy signal features for fault detection and tracking (OTA-1-tooth, unfiltered dataset, x-axis: Damage level 0-200, y-axis: Feature value, dashed line: Discrepancy threshold)