REMAINING USEFUL LIFE PREDICTIONS FOR BEARINGS USING SPECTROGRAM AND SCALOGRAM-BASED CONVOLUTIONAL NEURAL NETWORKS

Botao Wang

A thesis submitted in partial fulfillment of the requirements for the
Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa

Abstract

Bearings are critical in today's mechanisms, and their reliability is continuously improving. Yet, working under high loads for long periods, bearings will degrade and eventually fail. An unpredicted bearing failure can lead to total and catastrophic failures of machines and may even lead to human injuries that result in substantial economic losses and reductions in production. Determining a bearing's remaining useful life (RUL) has become an important topic in many industrial fields.

Vibration signals are the most used representation for understanding a bearing's health status. Using different algorithms, time-domain vibration signals can be transformed into time-frequency domain signals that help indicate a bearing's status. For instance, this thesis investigates spectrograms and scalograms to visually represent a bearing's health condition using a short-time Fourier transform (STFT) and a continuous wavelet transform (CWT). Both representations are plotted as a function of time and frequency and can detect the bearing's working condition. However, spectrograms are advantageous in revealing frequency changes along the time axis, while scalograms facilitate the detection of abrupt changes.

Combined with a convolutional neural network (CNN), these plots can be used to interpret bearing RUL. The strength of CNNs lie in their ability to identify and detect features in images, including such tasks as image classification, using share-weight architectures, convolutional layers, and kernels. This thesis explores CNNs combined with spectrograms and scalograms using the PRONOSTIA dataset to perform

bearing RUL predictions and explore relationships between prognosis and diagnosis for bearing faults analysis.

Acknowledgments

I would like to thank the following people who have helped me through my studies and in completing my thesis, and without whom, I would not be able to perform well in my master's degree.

My supervisor, Patrick Dumond, has helped me through my entire studies and provided enormous support in my personal life when I was having a hard time figuring out how my life should be.

My parents have supported me through my years of pursuing my studies abroad, both financially and emotionally.

My friends and colleagues that have helped me tackle problems I encountered while writing this thesis.

Once more, I thank all of you for appearing in my life and being such sunshines. As an old Chinese saying says: I would be happy to meet all of you again in any life I have in another hundred years.

Table of Contents

List of Figures

List of Tables

Nomenclature

2D: Two dimensional

AE: Acoustic emission

AR: Autoregressive

ARMA: Autoregressive moving average

ARIMA: Autoregressive integrated moving average

BPFO: Ball pass frequency outer race

BPFI: Ball pass frequency inner race

BSF: Ball spin frequency

C1: Channel 1 data from PRONOSITA dataset

C2: Channel 2 data from PRONOSITA dataset

CNN: Convolutional neural network

CWT: Continuous wavelet transform

FCF: Faulty characteristic frequencies

FFT: Fast Fourier transform

FT: Fourier transform

FTF: Fundamental train frequency

f_r: Bearing rotating frequency

HI: Health indicator

PRONOSTIA: Test rig created by FEMTO institute

RNN: Recurrent neural network

RPM: Revolutions per minute

RUL: Remaining useful life

STFT: Short-time Fourier Transform

SK: Spectral kurtosis

WT: Wavelet transform

Chapter 1 Introduction

1.1 Roller bearing faults

Machines are commonly used today, and the reliability of these devices has become a significant concern. Roller bearings carry and transfer loads as important connecting components inside most rotating machines. However, after working under high loads for long periods, bearings will degrade and eventually fail. Bearing failures can include damage to the inner or outer race and the ball itself [1]. Bearing defects can happen due to various reasons: misalignment of the bearing components, incorrect mounting positions or processes, manufacturing defects on bearing components, careless lubrication or loading conditions, fatigue, or an improper working environment, etc.

As bearings transfer forces among components, when the bearing defects appear in the middle of a transmission system, desired forces or rotations may not be delivered, causing a malfunction in the whole mechanism. An unexpected bearing failure can lead to consequential damage to adjacent regions, a total shutdown of the machine, and even human injuries. Each one of these failures can result in substantial economic losses due to delayed production, repair costs, and liabilities.

Therefore, bearing failures can be devastating if no precautionary measures are considered in advance.

1.2 Motivation

In real scenarios, companies rely on the "L_{10} life" rating to estimate a given bearing's life, which was first described by Palmgren [2]. It indicates how much time 90% of a group of bearings can work before experiencing metal fatigue and defects. Companies tend to use L_{10}, or other similar indicators (e.g. "L_n life"), to estimate their bearing's minimum expected life.

Replacing bearings before they fail seems reasonable, but it is hard for one to tell when a bearing should be replaced, especially when L_{10} does not deliver a guaranteed estimation of its useful life. In some cases, bearings are even designed to have a longer life than the equipment in which they are installed, but are not immune to statistical premature failures. According to a bearing manufacturing company (SKF)'s report, 10% of bearings do not outlive their equipment [3]. However, approximately 9.5% are replaced for preventive reasons, while only 0.5% are replaced because of failures. Moreover, roughly 10 billion bearings are manufactured worldwide yearly, which means that about 50 million bearings are replaced due to failure prevention every year. Not only does replacing a bearing require human resources, but it also usually means that an entire machine must be shut down for the replacement to occur. With such a vast number of bearings that may get replaced every year, it can be assured that the unnecessary replacement of bearings can be inconvenient and bothersome to

many companies. Moreover, within the 10% of bearings being replaced annually, only one-third are replaced due to fatigue. In contrast, the rest are replaced due to careless bearing maintenance or rare bearing quality issues. Since bearings are protected by machine casings, it is difficult to visually inspect and assess a bearing's health condition. Therefore, developing on-site and non-invasive bearing fault detection methods has been a topic of great interest over the last few decades.

Since failures are difficult to detect or observe outside the mechanism, engineers have developed many techniques for conducting machine fault diagnosis and prognosis for these failures. When initial defects occur, the repetitive impacts on the affected area of the bearing lead to the final failure of the bearing, called flaking or spalling [4]. As the defective areas are impacted repeatedly and grow, the vibration that the bearing experiences increases. Specifically, rolling element bearings are designed with finite rolling elements, which cause the bearing to generate a periodic vibration signal during operation. This periodic increase in vibration signal can be detected in an accelerometer signal collected over time due to the effects of the load changing in the defect zone [5]. According to Nectoux et al. [6], different degradation behavior patterns result in differences in the raw vibration signal data. Similarly, acoustic signals in a low-noise environment and current signals at a low critical frequency rate also indicate convincing defect signals [7]–[9]. A bearing's thermal reading has been used as well [10]. Furthermore, various methods for analyzing machine data have been developed for classifying and categorizing bearing failure types (diagnosis). Common methods include the fast spectral kurtosis (SK) and the

wavelet transform (WT) on vibration signals [11], as well as envelope analysis on acoustic emission (AE) signals [12]. For example, in this thesis, a novel method described by Kim et al. that combines several traditional methods, is modified and used on the PRONOSTIA dataset and shown to be helpful in identifying bearing failure types [13].

Most of these traditional methods were developed to find abrupt changes caused by defects in the bearing. Although caused by many underlying factors, bearing failures tend to have four major categories: inner race failures, outer race failures, rolling elements failures, and cage failures. Therefore, by applying different mathematical processes, raw time domain signals can be transferred to the frequency domain to identify distinctive frequency behaviors. Relying on the insight from experts in the field, who use different methods to assess situations on a case-by-case basis, the problem appears to be solved: a bearing's running signal is collected and sent regularly to experts for analysis, or alternatively, experts measure the bearing's health situation on site until a defect is found.

Although bearing fault diagnosis is a well studied field, companies have less interests in only knowing a bearing's specific failure type, as only identifying how a bearing has failed does not allow them to save costs or prevent sudden machine shutdowns. In addition, even if a bearing failure type is successfully identified, if bearings are only replaced when severe defects are detected, more time is needed for maintenance crews to act. Production shutdowns and expensive human resources are a primary concern for most organizations.

Unlike bearing fault diagnosis, which only detects bearing faults, prognosis provides knowledge of the bearing's remaining useful life (RUL). When provided with an estimate of a bearing's RUL, organizations have a more cost-effective way of maximizing the bearing's lifetime, while preventing sudden failures. This lets organizations make optimal bearing replacements during planned machine maintenance shutdowns.

This thesis aims to advance bearing RUL prediction methods based on improving signal interpolation and reducing expertise requirements using machine learning.

1.3 Signal preprocessing methods

Due to unavoidable noise pollution during data acquisition, and the total amount of data processed, raw data sets carry massive amounts of irrelevant information. Time can easily be wasted while picking out helpful representative features. Preprocessing the measured data is vital in transforming the raw data into more meaningful and understandable information. Preprocessing can also help reduce unwanted noise that collected during the data collection process. Different preprocessing methods can be used on different signals depending on what information is desired from the input data. In trying to find the RUL of bearings, researchers have developed different preprocessing methods to help in extracting a bearings' health condition.

Many methods have been applied to preprocess raw vibration signals for bearing fault-related monitoring. Since vibration signals are the most popular raw signals

used in the field of condition monitoring, this thesis will focus on preprocessing vibration signals. Popular methods include the fast Fourier transform (FFT) [14], the discrete wavelet transform (DWT) [15], envelope analysis [16], and the empirical wavelet transform (EWT) [17]. Each of these preprocessing methods modifies the raw vibration signal in different ways.

Two preprocessing methods are used in this thesis: spectrograms and scalograms. The spectrogram is a time-frequency domain signal computed using the short-time fast Fourier transform (STFT). By taking raw vibration signals segmentally and applying the Fourier transform (FT), time-domain signals are converted to the frequency domain. Each segment of the original signal is now represented as a Fourier spectrum, and plotting these spectrums as a function of time results in a spectrogram. The original vibration signal is transformed into a time-frequency domain signal and can be used to interpolate the frequency and time properties of the signal simultaneously. For example, Figure 1(a) provides a time domain representation of a car engine vibration amplitude as it changes over time. Figure 1(b) plots the spectrogram and indicates both changes in frequency and amplitude (colour variations) over time [18]. Therefore, the spectrogram provides an advantage when the input signal is unsteady, and time and frequency information is of interest simultaneously.

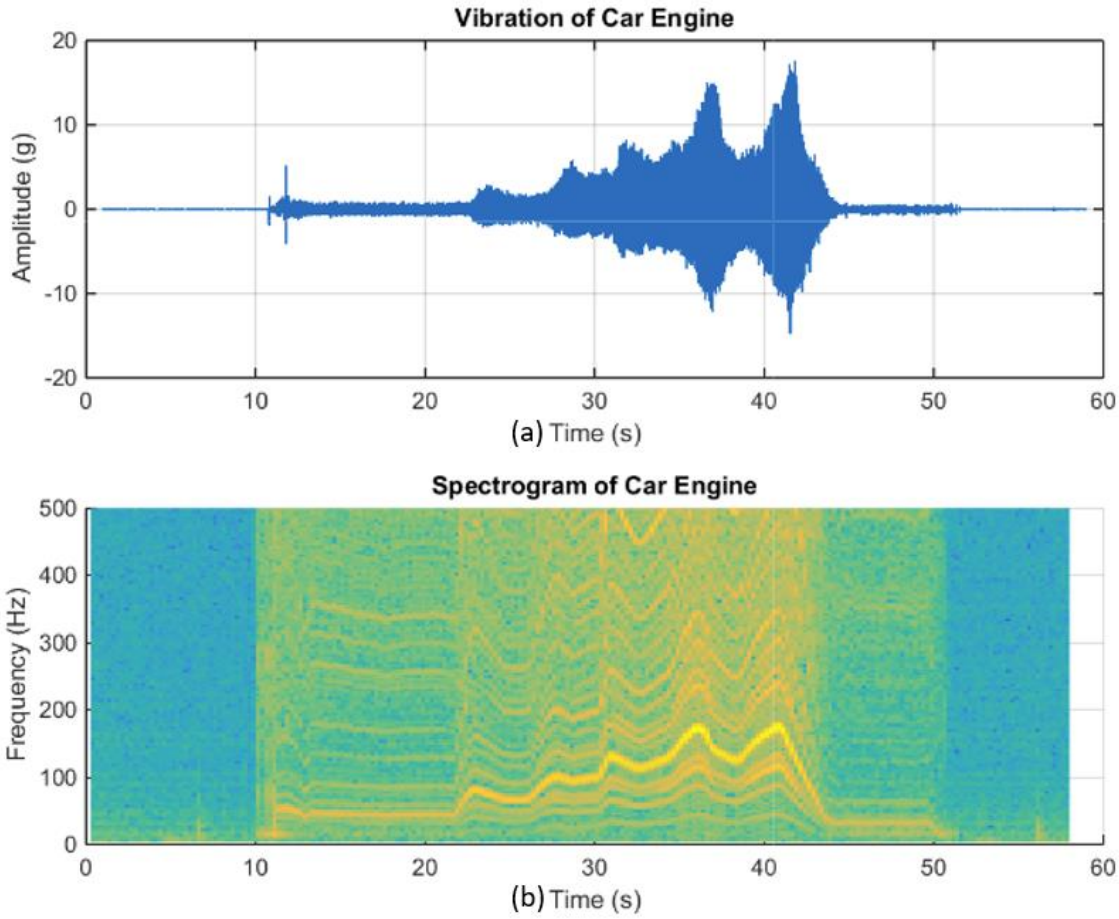


Figure 1 Signal diagrams of a car engine (a)raw vibration signal (b)spectrogram of raw vibration signal (modified from [18])

Scalograms, on the other hand, are computed by plotting the absolute values of the continuous wavelet transform (CWT). Like spectrograms, scalograms represent the time-frequency domain of signals. However, unlike applying the FT to time-series signals segmentally, the CWT multiplies a pre-set filter wavelet to the time-series signal. This multiplication is moved gradually along the whole signal. By introducing a pre-set wavelet, abrupt changes in the original signal can be emphasized, leading to a more distinct representation of signal variations. While the STFT uses a constant window size when applying the FT segmentally, resulting in a uniform frequency and

time scaling, the filter wavelet used when applying a CWT can be scaled and shifted in time. Therefore, by adapting the wavelet width, the CWT provides a better time and frequency resolution trade-off, which means that either frequency or time resolution can be emphasized in different scenarios. For example, Figure 2(a) shows two original signal peaks around the 400-time step that are emphasized as two red areas in the scalogram, Figure 2(b), with their corresponding frequency readings [19]. An important detail in the scalogram is that the frequency resolution in the low-frequency range is worse than the time resolution. However, the frequency resolution improves as the frequency range increases, providing a more detailed reading for high-frequency changes. On the other hand, lower frequencies typically represent ambient noise and do not carry as many informative features as high-frequency information.

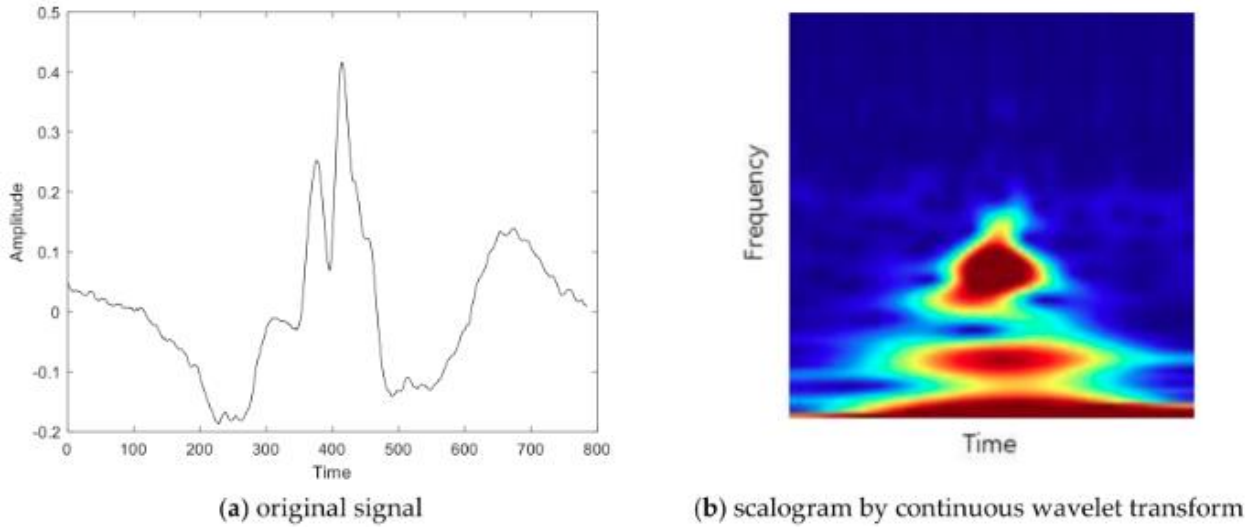


Figure 2 Original signal and scalogram for an electrocardiogram [19]

1.4 Bearing RUL estimations using machine learning

With auto-collected data, machine learning can analyze a bearing's health condition using artificial intelligence once the analytical model is built. When done correctly, the amount of human time and resources saved can be significant, and the need for maintenance engineering expertise can be reduced.

In supervised shallow learning, experts extract features from the input signals, then machine learning uses these features as inputs and learns their patterns. The algorithm is then able to predict similar signal features. This method requires field experts, who have deep knowledge of bearing faults and their signals, to spend a great deal of time extracting valuable features and feeding them into the algorithms. However, experts can also make mistakes or miss critical failure features. Moreover, more input data requires more time extracting features, which can prolong the prediction process. Nonetheless, well-designed algorithms can help save time by making decisions after taking in data. However, the requirement for expert knowledge and long analyzing times do not justify this as a universal solution.

Conversely, unsupervised deep learning was developed to overcome these obstacles. Deep learning takes in large amounts of raw measurement data and learns essential features without requiring assistance from bearing experts. Instead of maintenance engineers looking at the signals and distinguishing features using their past experiences, deep learning uses multiple hidden layers to extract key features of the input data and eliminate redundant or useless information. With this process, deep learning can deal with massive input data, lowering the chances of missing

failure features and discovering features unknown to experts. Shallow and deep machine learning work based on a similar concept, taking in and analyzing raw data and then making predictions on new data. However, shallow machine learning requires experts to help extract features and also requires experts to evaluate and assess the predictions. On the other hand, deep learning has the potential to evaluate outcomes by itself and provide feedback to its algorithm, thereby improving feature predictions on its own. Nonetheless, a great deal of work remains to be done if bearing fault diagnosis and prognosis are to meet expectations in the field.

Convolutional neural networks (CNNs) are a set of powerful deep learning methods proven to work for image classification. They use filters to detect the patterns in given pictures, for example edges, corners, curves, shapes, or even specific objects. By applying multiple filters, a concise pattern can be extracted and used for future comparison. CNNs have also proven to be helpful in solving bearing RUL prediction problems [20]. For example, Lei et al. used a feature extraction method that generates the spectrum's eigenvector and combines it with a CNN to predict a bearing's RUL [21]. Jun et al. used a multiscale CNN based on time-frequency representation to predict a bearing's RUL and also got valid results [22].

CNNs have several advantages for studying bearing RUL:

- CNNs work with images as input data. When raw vibration signals are transformed into more informative image representations, algorithms can use these representations as a better starting point to study bearing failure patterns. Also, by reading input images, CNNs can be used to simulate the

human visual system, which is similar to the way field experts would look at spectrograms in the field.

- CNNs, as a deep learning method, do not need human interactions in the feature extraction process. The algorithms learn features on their own, saving human resources and avoiding missing possible failure features. Depending on the training data, algorithms can develop different filtering methods for different failure patterns.
- Bearing vibration signals normally contain massive amounts of data. Not only can CNNs function with such large data, but they become better at prediction as the data set size increases. With more input data, a greater number of hidden features are exposed to the CNN's filtering layers and it becomes easier to learn.

1.5 The relationship between diagnosis and prognosis

To improve prognosis, information about a bearing's developing fault type may help inform prognosis. For this reason, this thesis will explore the relationship between diagnosis and prognosis. It is believed that prognosis could be improved during the bearing degradation process when faults can be identified by diagnosis. The possible relationship between diagnosis and prognosis can potentially include:

- Diagnosis can inform when to start the prognosis process, rather than doing prognosis from the beginning of the bearing's life.

- Diagnosis results could potentially inform how to perform prognosis, or enlighten the appropriate methodology to apply.
- By using diagnosis, bearing failures could be categorized into different groups, and specific prognosis methods could be identified as optimal for each group, making the process more efficient and accurate for each failure type.

1.6 Contributions

This thesis compares existing methods for predicting bearing RUL and develops new methods based on knowledge gained in this process. The main thesis contributions include:

- Applying diagnosis to the PRONOSTIA dataset, identifying different bearing failure types for each bearing.
- Processing raw vibration signals into STFT-based spectrograms and CWT-based scalograms and comparing their effectiveness at both bearing diagnosis and prognosis.
- Developing a custom CNN-based algorithm using an autoregressive integrated moving average (ARIMA) for estimating bearing RUL based on the results of diagnosis.
- Comparing prediction results to determine the best preprocessing method for using the proposed algorithm.

Chapter 2 Literature Review

2.1 Traditional bearing fault classification approaches

2.1.1 Signals used for bearing fault classification

Before considering common fault detection methods, available data types and their characteristics should be understood. While bearings operate, various signals can be detected to represent their health states and failure potentials. Motor current signals, AE, temperature readings, and variations in acceleration (vibration signals) are used to help people obtain bearing diagnosis and prognosis [8], [9], [23], [24]. Each of these signals has been shown to carry sufficient information for particular fault cases.

For example, as early as 1990, acoustic signals were already used broadly in bearing failure detection. This method's accuracy is considered to be proportional to defect sizes and rotation speeds [23]. Environmental noise is a vital factor affecting the accuracy of the AE method and AE root-mean-square values have been found to perform better for ball faults than inner race faults. By applying the SK technique and Kurtogram theory to acoustic data, impulsiveness in the spectrum, even when it is exposed to additive noise becomes clear [24]. After years of development, the AE method has become a reliable way of providing early detection in bearing failures.

Myeongsu et al. completed a bearing failure detection test in which the failures were found to be cracks on the outer-race (COR), cracks on the inner-race (CR), and cracks on the roller (CIR) [25]. The experiment was carried out in a noisy environment with a sampling frequency set to 1 MHz. In this case, the average detection accuracy was found to be above 95%. The authors used a diagnostic procedure based on fault signature extraction on time-frequency domains and decision-making with using a support vector machine (SVM). After putting the raw AE signal into a Daubechies mother wavelet function family containing 5 levels, the authors separated the signal into 32 evenly distributed frequency regions, called terminal nodes, based on the total number of mean-peak ratios and then plotted processed data as spectrums. Finally, the authors used relative wavelet packet energy (RWPE) and wavelet packet node kurtosis (WPNK) to determine the faults from the signals. The authors also showed the potential of their AE-based multicore system for shorter execution times and higher energy reductions by applying multicore analysis.

On the other hand, Hoang et al. successfully completed bearing fault diagnosis using motor current signals and a deep learning method [26]. In this case, the authors suggested that even though vibration signals carry richer information, the expensive accelerometers and difficulty in installing them onto a machine could be troublesome. In contrast, motor current monitoring systems are built-in for many systems so that current-based detection methods have significant economic and implementation advantages. The limitations of current signals were also highlighted in that bearings can be installed inside or outside of a motor and due to the signal transfer process

(transmitted by torque vibration) and noise from powered processes, the external bearing signals were found to be hard to use. Hoang et al. claimed that Lessmeier et al. and Karatzinis et al., who used external bearing current signals, did not obtain results that had satisfactory classification accuracy [27] [28]. Using a CNN, Hoang et al. achieved an accuracy of 98.3% by using current signals.

Yilmaz et al. found that temperature signals could be a valid factor for bearing fault classification as well, especially for lubricant related cases [9]. Temperature signals were found to benefit diagnosis when the cracks and failures inside a bearing increased temperature by as little as 5℃. However, the authors also suggested that the temperature signal alone is not precise enough to determine fault type and may only be used for general or quick examination purposes. On the other hand, Choudhary et al. used thermal imaging instead of temperature itself to classify bearing faults [10]. The DWT and SVM combined with thermal radiation images were found to result in promising predictions. Even though the practicality of using such a method would need to be tested for larger data sets, it still represents an interesting path for future bearing fault classification investigations.

Vibration signals are still the most common type of information used today since they carry more accurate and detailed information regarding bearing operating conditions. Moreover, with appropriate mathematical transformations, vibration signals can be transferred from the time domain to the time-frequency domain, uncovering additional interesting features. By transforming raw vibration signals into the frequency domain, four major faulty characteristic frequencies (FCFs) for

ball bearings are typically considered: ball pass frequency outer race (BPFO), ball pass frequency inner race (BPFI), ball spin frequency (BSF) and fundamental train frequency (FTF) [29]. Using the FCFs, possible fault types can be determined.

Ali et al. mention how early-stage failure signals can be used for bearing fault analysis, but also mention how these same signals can be strongly affected by noise [30]. Therefore, the authors developed an artificial neural network (ANN) combined with an energy entropy-based empirical mode decomposition method and obtained result of 93% accuracy when examining bearing states. In the authors' work, vibration signals are converted into intrinsic mode functions (IMFs) using empirical mode decomposition (EMD). With this step, non-linear and non-stationary raw vibration signals can be interpreted as signals with trends. In Li et al.'s work, an FFT is used to convert time-domain vibration signals to frequency-domain signals [31]. In this case, failure frequencies were determined and combined with a neural network-based algorithm to detect bearing faults.

Different types of signals can be used for particular situations. For example, AEs are used when the component of interest's volume level is higher than its surroundings, temperature readings can be advantageous when looking at lubrication related problems, motor currents are more suitable when lower cost is required or as an early ambiguous estimation, and vibration signals serve as the richest form of information that can be used to detect hidden features. Since vibration signals are the most common signals for bearing fault classification, this thesis will focus on using vibration signals.

2.1.2 Vibration signal processing methods

Raw vibration signals can be mathematically processed to represent certain characteristics for different needs. Most processing methods serve the purpose of reconstructing the raw time domain signal into frequency or time-frequency domains to reveal behaviors that cannot be seen in the time domain alone. Two signal processing families closely related to this thesis work and some commonly used methods are listed as follows:

(a) The Fourier transform (FT)

As most time domain signals carry insufficient information, it is reasonable to decompose them and analyze them from a different perspective. Most time domain signals can be seen as numerous sinusoids combined into a composite signal. Therefore, decomposing original signals to individual sinusoids helps in analyzing their features and eliminating undesirable noise. The FT is one of the most critical mathematical concepts created to solve these problems in the past century. It is used to transfer any periodic time domain signal into frequency domain data, so original signal frequency behaviors can be revealed. For instance, decomposing music signals provides pitch information instead of intensities. The mathematical expression of FT is shown in equation (1).

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} \partial t \tag{1}$$

where $x(t)$ represents the time-series function, and $e^{-i\omega t}$ is the complex representation of sinusoids [32].

The discrete Fourier transform (DFT) is the finite version of the FT. It has become the basis of many basic algorithms as it serves numerous industries, including data transfer, sound and image suppression, circuit designs and most valuable for mechanical engineers – signal processing and analysis [33]. Unlike the FT, the DFT takes only a finite range of samples of a function and converts them into a complex value-involved function of frequencies. While the DFT establishes the relationship between the time and frequency domain representations, it is also meant to perform the FT on computers using a finite input. However, with the increase in computational needs and technology development, the FFT has mostly taken over. The FFT is an algorithm explicitly designed to compute the DFT faster, because the DFT requires substantial calculation power and by categorizing the original time domain signal into the same signal product with sparse factors, it is time-consuming [34]. Overall, the FFT is an efficient algorithm designed to execute the DFT faster on large datasets. Strömbergsson et al. applied the FFT to faulty bearing diagnosis and successfully determined the BSF and the BPFI in bearings [35]. As shown in Figure 3, f_{IR1} represents the first BPFI, indicating an inner race defect, where f_{IR2} and f_{IR3} are the harmonics of f_{IR1}; f_{SB1} and f_{SB2} are BSFs indicating ball defects. However, the FFT requires that the signal be continuous and periodic at the same time, which means some continuous, yet aperiodic signals cannot be computed using the FFT. Using the FFT alone introduces the trouble of dealing with complex output for real-valued inputs, and it cannot represent the signal's power information.

Also, the FFT can be unreliable for identifying inner race (IR) defects during a bearing's early failure stages, when vibration signals are much weaker [36].

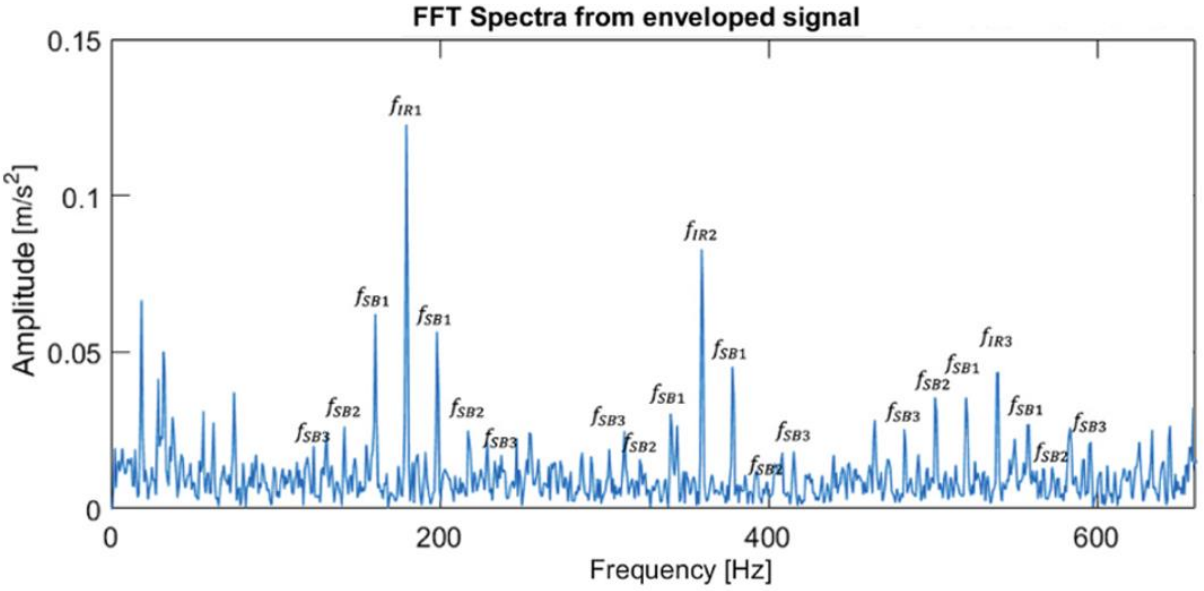


Figure 3 FFT spectrum for a BPFI and BSF fault bearing (modified from [35])

With time domain and frequency domain information provided simultaneously, bearing health states can be determined further. For example, in Figure 4, Pham et al. use spectrograms created using the STFT to represent bearing inner race faults, outer race faults and roller element faults at various RPMs [37]. Based on different fault types, frequency readings and intensities in the time domain can vary. It may be hard for human eyes to identify key characteristics, but with the help of artificial intelligence, informative images can describe bearing health states more accurately. In their paper, using a VGG16 algorithm, classification with an accuracy of 98% and above was achieved.

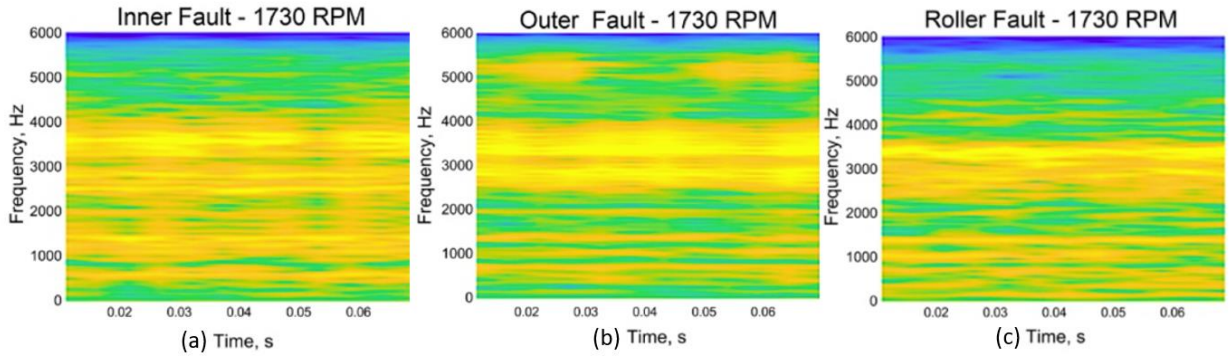


Figure 4 Example spectrograms of (a) inner race defects (b) outer race defects and (c) roller defects (modified from [37])

In summary, a time series or FT series representation can give only time or frequency information for one signal, respectively. To obtain two pieces of information at the same time, the STFT can be used to create a spectrogram that represents time and frequency in the same diagram. This is done by convolving the FT with a chosen mother wavelet moving along the raw time series signal [38]. However, the spectrogram is fixed within one resolution per diagram, and more is needed in some cases.

(b) The wavelet transform (WT)

To improve upon the resolution of the STFT, the wavelet transform (WT) can be used instead. The WT is a time-frequency-based representation method like the spectrogram, but provides multi-resolution on time and frequency domains. The principle that drives the method is that low-frequency tends to last longer and stays more inactive compared to high-frequency signals. Therefore, by adding an increasing resolution gradient, higher frequency signals can be emphasized and provide a better representation of the signal's behavior. An example of the differences between the FT, STFT and WT can be seen in Figure 5, where time and frequency have a constant

resolution for the STFT [39], but for WT, signals are sectioned more often in the higher frequency region when compared to the lower frequency regions to provide a finer time resolution. This can help identify behaviors that are more likely to happen in the higher frequency regions.

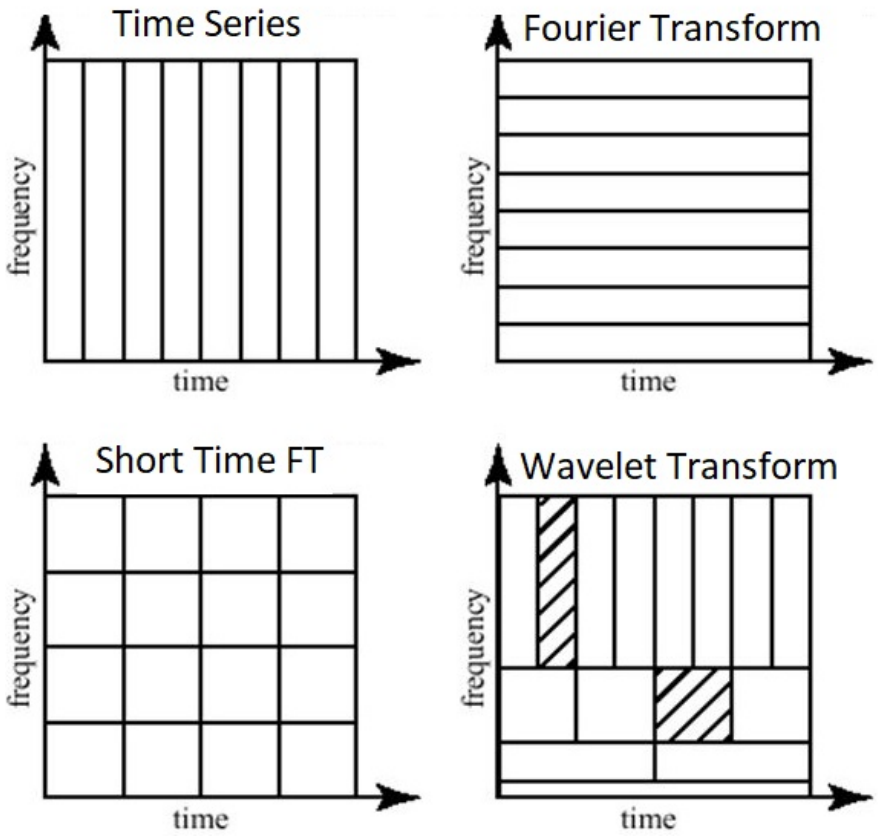


Figure 5 Different resolution representations among time series, FT, STFT, WT diagrams [39]

Both the continuous wavelet transform (CWT) and the discrete wavelet transform (DWT) can be used for bearing fault analysis. García-Prada et al. used the DWT for feature extraction during bearing fault analysis [40]. In their work, the DWT is applied to the 5^{th} level, meaning the original signal is decomposed five times for denoising. By inputting that result into an ANN, their method gave a moderately

successful result, but their success rate in identifying inner race and outer race faults was lower. Nonetheless, the CWT and the DWT have essentially the same working procedures; however, with today's computational power, the CWT can compute the signal with a finer resolution with denser scaling and shifting parameters, which makes the CWT perform better for signal analysis in today's environment [41].

Yoo and Baek used the CWT for the whole life of test the bearings and created plots for their bearings' health conditions, as shown in Figure 6 [42]. From the graph, healthy bearing CWT Figure 6(b) has low wavelet coefficients (colder colours) in the middle-frequency region. In contrast, unhealthy bearing CWT Figure 6(d) has high wavelet coefficients (warmer colours) in the middle-frequency region, indicating bearing defects. In addition, the periodic peaks shown in the time domain signal of Figure 6(c) are also represented as repetitive warm colour regions. Gao et al. used a method that combined the complex Morlet CWT with a CNN to identify bearing fault types [43]. The authors applied the CWT on the original vibration signal and used the output, which is represented by a scalogram, as the input for the CNN. The results are promising and indicated that people without bearing knowledge may still be able to perform bearing fault analysis. In Konar and Chattopadhyay's work, a SVM with a CWT was used for bearing fault detection of an induction motor. The authors stated that the best mother wavelets for the CWT in their case were the 'morlet' and 'daubechies10' and also suggested that combined SVM-CWT detecting structure works better than the DWT-ANN method [44]. However, the authors also mentioned how tedious it was to find the best mother wavelet for the CWT for different cases.

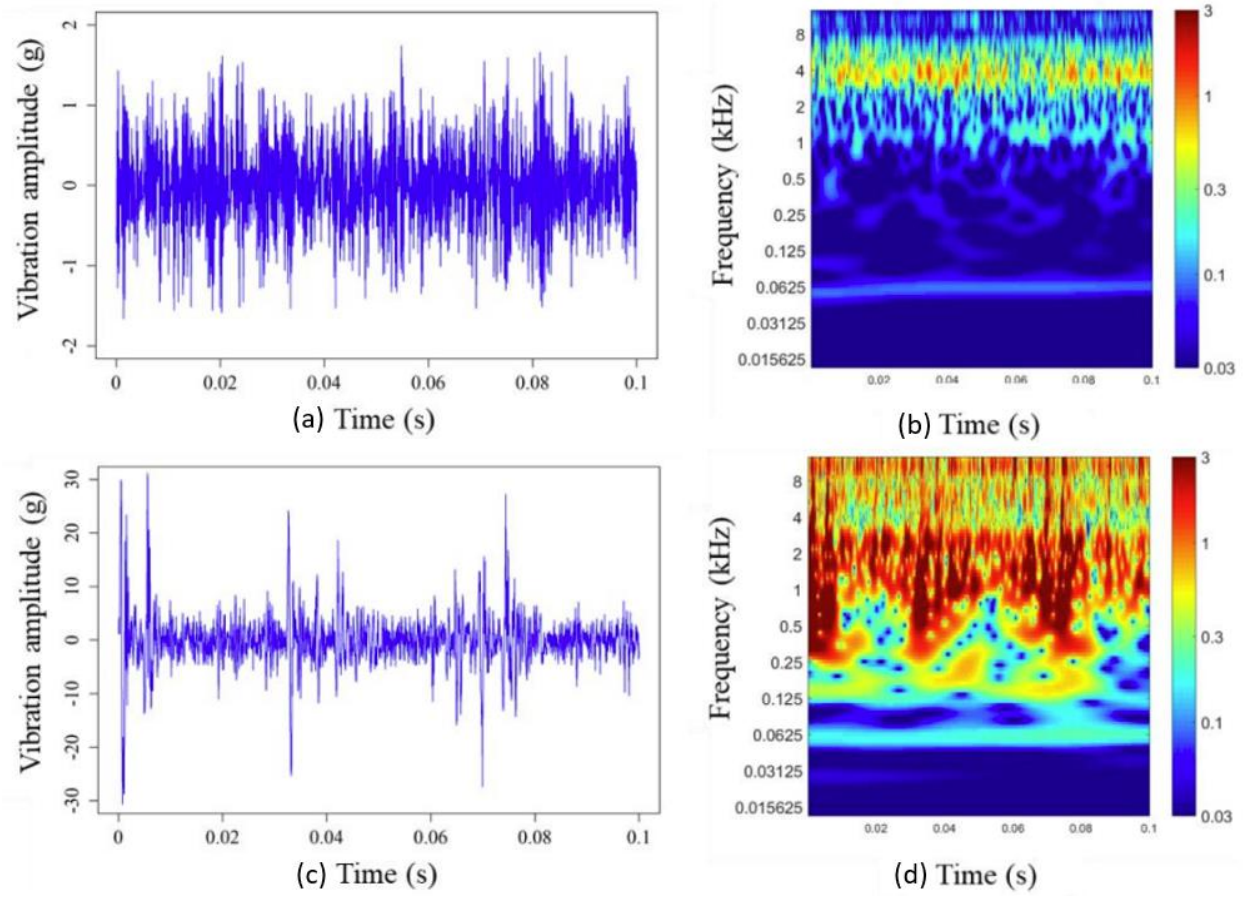


Figure 6 Health states of a bearing as (a) healthy raw vibration signal (b) healthy scalogram (c) unhealthy raw vibration signal (d) unhealthy scalogram [42]

(c) Other popular signal processing methods

The Hilbert transform (HT)

The HT is used to translate one-dimensional signals into two dimensional representations in complex form. The real part indicates the original signal's behavior and the imaginary part represents the signal's phase and shift. As the HT isolates the frequency domain signal, it is helpful when identifying faulty frequency types. Liu uses the HT to compare the frequency behaviors for different defect sizes and

working rotation speeds [45]. From his work, he states that the severity of faulty frequencies is not proportional to defect size, but it increases as the defect grows and smooths out as the defect forms completely. This severity decrease is due to the edge of the fault cavity growing more polished and decreasing the strength of sharp impulses, which can be used as a sign to replace bearings.

Spectral kurtosis (SK)

Kurtosis is used to represent the deviation of a probability function away from its Gaussian distribution of the historical trend, and the SK further represents the most non-stationary behavior of all frequency bands in the signal. Since the values of SK rely on the choice of frequency resolution, finding the optimal frequency resolution for different frequencies is important. Solving the SK results in a kurtogram, which helps determine the bandwidth where the fault has a dominant effect [46]. The SK expresses low values for stationary frequencies and high values for the transients of the frequencies [47]. In bearing analysis, non-stationary or abrupt changes in the signal can be seen as a hint of defects, and many researchers use SK and the kurtogram for analyzing bearing failures [48], [49].

As many methods can be used to convert raw vibration signals into plots with frequency and time representations, researchers find that there is too much information for human eyes to handle. Therefore, discriminating approaches are developed and artificial intelligence (AI) is introduced. AI is designed to think and perform tasks like a human being, while able to handle much larger volumes of information, making it perfect for discriminating overwhelming amounts of data.

Finally, machine learning (ML) is the algorithm that trains the AI to achieve those goals.

2.1.3 Machine learning algorithms for classification

Based on vibration signals, different machine learning technics have been applied historically been applied to the study of bearing fault classification. The two main categories of ML are supervised ML and unsupervised ML. Supervised ML is based on comparing previous experiences or the "corrected answers" to the predicted outcome, categorizing outputs into pre-set groups. Before performing supervised ML, it is essential to use known labeled datasets to train algorithms, leading to outcomes that can only classify within labeled groups. Unsupervised ML is used to determine unknown features based on distinctive data behaviors compared to those present in the general signal [50]. Unsupervised ML algorithms must set their own new labels and groups for their use. In terms of bearing fault-related topics, supervised ML is mainly used for identifying bearing faults based on characteristic fault frequencies (BPFI, BPFO, BSF, FTF) because the fault frequencies are identified and can be labeled before classification. Moreover, unsupervised ML is commonly used to determine a bearing's RUL, where the remaining life of a bearing would always be unknown to researchers beforehand.

Many valid signal processing methods/algorithms have been developed for bearing fault classification. The most common methods are briefly introduced below.

(a) Support vector machine (SVM) based methods

SVM is a commonly used supervised ML method for bearing fault diagnosis. As shown in Figure 7, it identifies a hyperplane that best splits the processed signals into two categories, blue and red dots here. Using the hyperplane for identification, signals are separated into two groups: one represents healthy signals, and the other represents signals containing defects. Therefore, whenever a new data point lands in the plot, it would be classified as one of the two groups and be identified as such. Optimizing the hyperplane so that is best separates the two categories of interest is important. A way of doing this is to create a margin representing the closest distance between the hyperplane and the support vectors. By maximizing the margin value, the two groups of data can be divided as widely as possible, which means the optimal hyperplane can be retrieved. The strength of a SVM is that it can result in a reasonable prediction even if the sample data point is not following the perfect prediction pattern. By using such a straightforward principle, SVMs are very efficient and easy to implement. The hyperplane in Figure 7 is a straight line in a 2-D plane, but it can be a curved line in a 2-D plane, or any form in n-1 dimensions for more precise analysis. The SVM method can be used when data is not preprocessed and still performs well for smaller sample sizes [51]. Fernández-Francos et al. used a one-class variant-SVM over a group of lab-based run-to-failure and a pre-seeded bearing fault test dataset. A SVM is only applied for bearing failure classification before going into specific fault frequency classification, which is found to be more effective. They achieve bearing fault diagnosis with 100% accuracy when they discriminate faulty

bearing signals from healthy bearing signals [52]. However, SVMs do not perform as well in situations where the data is too noisy or the amount of data is too large, causing training time to be too long. Unfortunately, this often represents a typical real-life bearing operating environment [53]. In addition, SVMs can only classify two categories at once. Therefore, it is best suited for identifying if a bearing is faulty or not.

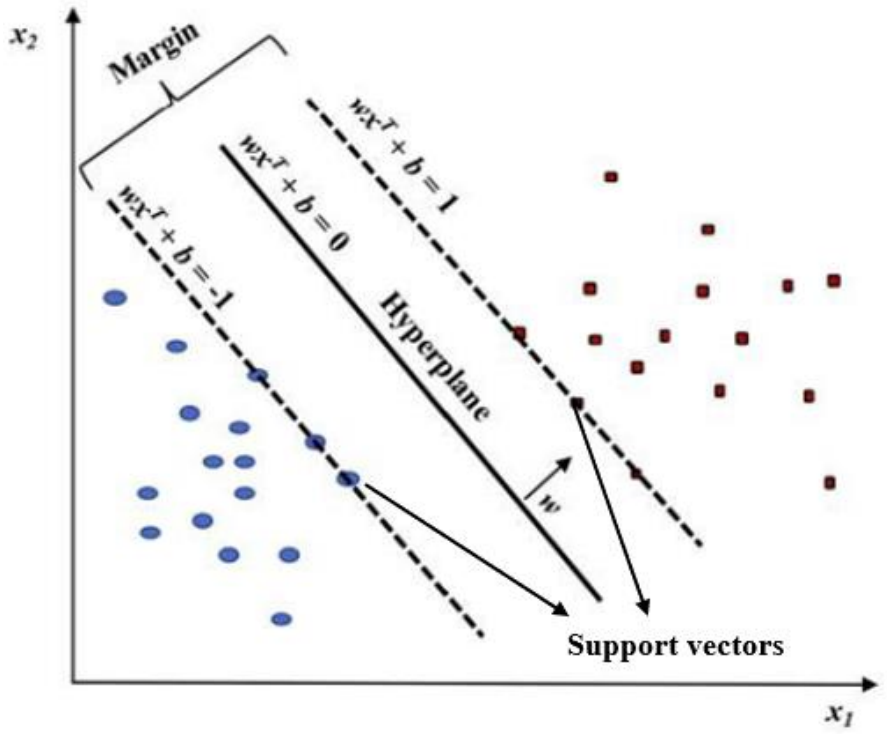


Figure 7 SVM demonstration diagram (modified from [54])

(b) K-nearest Neighbor (KNN)

KNN, another supervised ML method, is also a commonly used decision-making algorithm. The main advantage of KNN when compared to SVMs is that it can be used to classify multiple categories. Known datasets are separated into multiple

groups, and then one can set a value for k indicating how many known data points will be used to classify the unknown data point. The algorithm will find the k nearest known categorized datapoints to the tested point. By simply counting the most shown annotated points, the unknown data point can be associated with the category having the most points within the range of k. For example, in Figure 8, k is set to five, meaning that the five nearest points are chosen to define the new data point. Three out of the five points are in category A. Thus, the new data point is defined as being in category A. Tian et al. successfully detected bearing faults for a motor using the semi-supervised KNN method [48]. In their work, KNN is not solely used for categorizing, but by comparing how far away the test points are from the healthy group centroid, they can set a threshold for identifying the test point that comes from a faulty bearing. However, KNN's prediction accuracy highly relies on the k factor chosen, which remains a human decision. Furthermore, KNN works best with a clean dataset, which means the reliability of the prediction decreases when the data are contaminated.

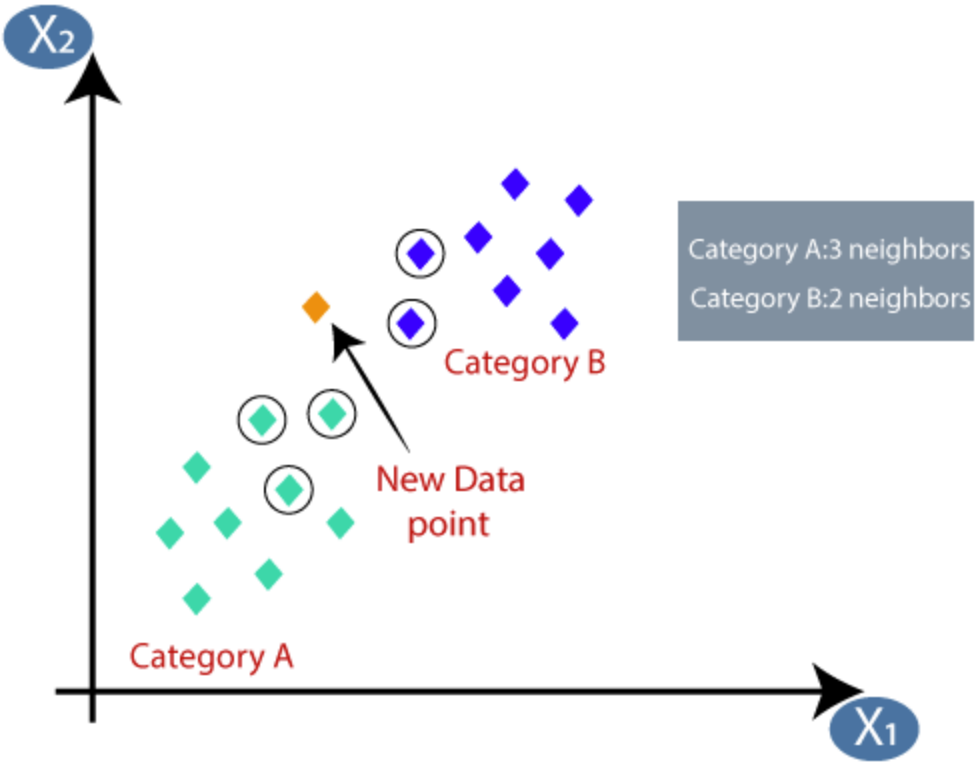


Figure 8 Example of a KNN diagram when k=5 [55]

Combining signal processing methods with ML algorithms, bearing fault classification can be achieved. Various needs can be fulfilled by using different signal processing methods with specific ML algorithms. However, ML has some downsides as well:

- Professional knowledge requirements: during the feature-extraction stage of supervised ML, experts must manually label the processed signals for ML algorithms to learn from. This step requires deep professional knowledge. However, with more and more data, the rate of human mistakes typically increase, and hidden features can be missed.

- Simple but not "smart" enough: ML algorithms struggle to deal with multiple bearing defects that appear simultaneously, which can confuse algorithms. Some non-characteristic fault types cannot be identified beforehand, for example, bearing lubrication, assembly looseness and external environment changes.
- Non-transferable: the pre-set labels are fixed for each circumstance, and if the bearing or its working condition change, new labels must be made and applied again.

To overcome these challenges, deep learning (DL) methods are introduced.

2.2 Deep learning

While ML requires humans to be involved in feature extraction, allowing ML algorithms to learn extracted features and make decisions on incoming data, DL can skip this step. Instead of being fed with organized categories, DL algorithms can extract features by themselves and use those features to train the algorithms for predictions.

Three major types of DL algorithms are commonly used for bearing fault analysis, including: artificial neural networks (ANNs), recurrent neural networks (RNNs), and convolutional neural networks (CNNs). The ANN is the simplest form of neural network. It is composed of layers of interconnected nodes that process and transform input data through mathematical operations, producing output predictions. ANNs are generally used for classification and regression purposes. The RNN deals with

sequential data and is widely used to analyze time-series data. The "recurrent" aspect of the RNN is characterized by a persistance of information over time and can be helpful when dealing with raw vibration signals or time-series related bearing signals. The CNN, on the other hand, is designed for image analysis problems. Since a major goal of this thesis is to predict bearing RUL through preprocessed image-like information, CNNs will be used as the primary algorithms in this thesis.

CNNs are a type of machine learning method that are designed to emulate the human neural system [56]. CNNs were originally designed for image recognition and restoration and have been proven very successful [57], [58]. For bearing fault classification using several different signal processing methods, 1-D raw vibration signals can be converted into coloured 2-D representations (such as spectrograms, scalograms, or kurtograms). Since bearing signals can be converted into input types that CNNs can understand, many studies have been conducted using CNNs for bearing fault classification.

For bearing fault diagnosis, Eren et al. used a 1-D CNN for bearing fault diagnosis [59]. The original vibration signals were fed directly into their 1-D CNN and they were able to achieve 93% accuracy for two distinct datasets. In their work, they described the disadvantages of traditional classification methods, which is that human-designed features are not optimal for representing most underlying characteristics in many circumstances. They also point out that other researchers who used KNN with the WT, and a SVM with the wavelet packet transform (WP) reported high accuracy on similar tasks due to small train/test datasets [60], [61].

Wang et al. compare the accuracy differences of using both 1-D and 2-D CNNs on bearing fault diagnosis [62]. In their work, the FFT was used to convert 1-D signals into 2-D signals, and by applying CNNs to these inputs, they stated that 1-D CNNs had better accuracy than using 2-D CNNs alone when dealing with waveform signals. But when 1-D CNNs were combined with 2-D CNNs, an accuracy of 98% was achieved, and 2-D CNNs were shown to extract features that 1-D CNNs could not. Therefore, increasing CNN dimensions does not simply increase the overall prediction accuracy, but high-dimensional CNNs can extract more features than low dimensional CNNs.

For bearing fault prognosis, Sutrisno et al. summarized three methods they used to detect bearing failures and estimate their remaining life [63]. Data were used in either the time domain or converted to the frequency domain, and then methods like root mean square, peak amplitude and kurtosis were used to analyze for possible bearing failures. Then, a least squares signal regression method was used to refine and normalize the vibration data to provide a prognosis on the remaining useful life of a bearing. Finally, they used vibration frequency signature anomaly detection, along with survival time ratios to estimate the remaining life.

On its own, ML is incapable of determining a bearing's RUL without known labels to represent a bearing's life. But CNNs can extract certain features that represent the bearing's health state as a health indicator (HI), and by using those newly defined features, CNNs can learn and train themselves to estimate the bearing's RUL. Ren et al. used CNNs with the FFT to perform bearing RUL predictions [21]. They could identify the degradation pattern of the bearing, and by comparing it with traditional

feature extraction methods, their method was shown to provide generally better results as well. Li et al. combined CNNs and long short-term memory (LSTM) to predict the RUL of bearings and achieved better results than using CNNs alone [20].

2.3 The autoregressive integrated moving average (ARIMA) model

For bearing RUL predictions, the main goal is to determine the bearing's remaining life after bearing faults are detected. Therefore, a forecasting method is required to make these predictions. The ARIMA model is a forecasting method for time series problems, for example, predicting future stock prices and COVID propagtion [64], [65]. Since the bearing degradation process can be described as a decaying behavior in the time domain, the ARIMA model can be used in predicting bearing RUL.

The ARIMA model works by using autoregression (AR) and by applying a moving average (MA). The ARMA model can also be used to forecast data with a stationary mean value, but since bearing degradation is a non-stationary process, the ARIMA model is preferred in this situation. The AR expresses the relationship between current values and past values, while the MA is used to eliminate the random error fluctuation generated during the AR process. But ARMA models are restricted to stationary data, which implies that the mean and variance of the data do not change over time. On the other hand, the "I" in the ARIMA model indicates integration,

which helps convert non-stationary data to stationary data. As for bearing RUL predictions, the entire degradation process can be treated as a linear descending process, which makes the ARIMA model better than the ARMA model. A more detailed working principle will be represented in section 3.3.5.

In Sun et al.'s work, the ARIMA model is applied to predict the load and stress changes in bearings [66]. Their results indicate that ARIMA predictions are accurate when the prediction steps are small, and as the prediction steps increase, the ARIMA model results get worse. Nonetheless, they state that the ARIMA model is still useful, as the bearing degradation process has fewer prediction steps to make when the bearing is about to fail. Luo et al. used an ARIMA model combined with the recurrent process (ARIMA-R) to help analyze bearing health states and make predictions [67]. Their work indicates that their model has superior performance compared to using the autoregressive neural network (ARNN) on bearing RUL predictions. Their model was shown to be useful in scenarios where calculating predictions based on full sets of historical training samples is not available, or when using non-recursive ML methods. They also found that when long-term predictions were desired, the ARIMA model-based algorithm did not perform as well as for short-term predictions. Lu et al. compared their physics-based wavelet neural network (WNN) with the ARIMA model for bearing RUL predictions [68]. In their work, the ARIMA model outperformed their model for one test bearing, but their model had better overall prediction results. They also stated two major characteristics of the ARIMA model: the ARIMA model

performs worse when abnormal degradation patterns appear, and the ARIMA model works better when significant damage is detected.

Chapter 3 Methodology for Diagnosis and Prognosis

This chapter will present and explain the datasets and methods used in this thesis. The methods can be grouped into two sections, diagnosis, and prognosis. Their relationship will be explained at the end of the chapter.

3.1 PRONOSTIA dataset

Before introducing methodologies for analysing bearing signals, it is important to describe the dataset used for this thesis, and how it will be processed to help in the analysis procedures. The dataset used in this thesis was retrieved from the FEMTO-ST Institute in France. The data were captured from an experimental platform called PRONOSTIA. The dataset was collected based on accelerated life tests to simulate real bearing degradation. As shown in Figure 9, the test rig consisted of three major parts:

(a) The rotating part

An asynchronous motor with a power of 250 W was used to power the system, and a gearbox was used to transfer the load with desired values of

revolutions per minute (RPM) and torque. A shaft, supported by two pillow blocks, was used to deliver required motion to the test bearing.

(b) The loading part

To perform the accelerated test, an external load was introduced using a pneumatic jack. The external force generated was indirectly applied to the test bearing through a clamping ring. The radial force was set to 4000 N, which is the bearing's maximum dynamic load.

(c) The measurement part

Two types of signals were collected in this test rig using vibration and temperature sensors with 25.6 kHz and 10 Hz sampling, respectively. The vibration sensors, placed as close to the external race of the bearing as possible, consisted of two accelerometers recording the vertical and horizontal vibration signals of the bearing. This resulted in two signal channels called "Channel 1" for the vertical vibration reading and "Channel 2" for the horizontal vibration reading. The radial force, rotation speed and torque applied to the shaft were collected at a sampling frequency of 100 Hz and were used together to determine the bearing's degradation. Finally, the data acquisition system collected the data.

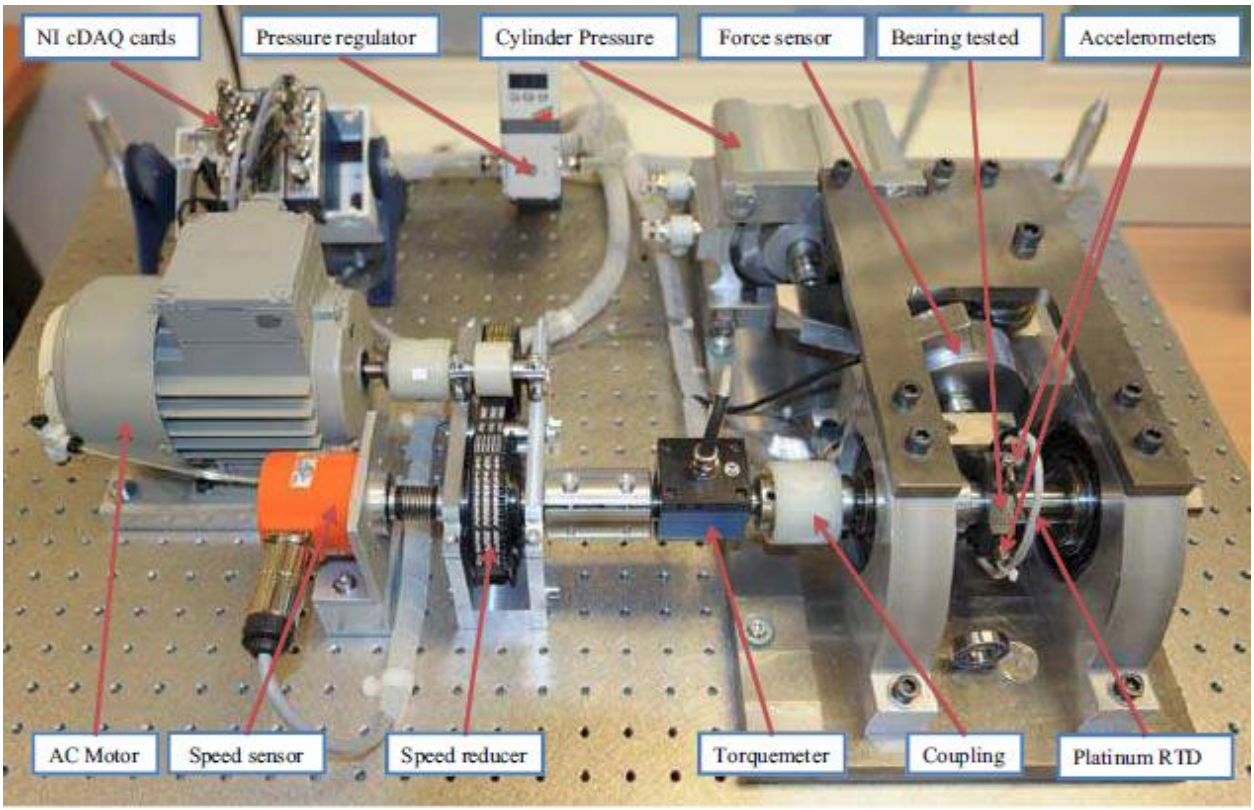


Figure 9 PRONOSTIA test rig [6]

Three different bearing working condition were provided to be used for diagnosis and prognosis purposes, as described in Table 1.

Table 1 PRONOSTIA test bearing working conditions

	RPM	Load
Condition 1	1800 rpm	4000 N
Condition 2	1650 rpm	4200 N
Condition 3	1500 rpm	5000 N

Researchers were originally provided with six run-to-failure datasets based on these conditions and were challenged to estimate the RUL of other bearings with only part of the total life readings given. Equations (2), (3), and (4) are used to estimate the accuracy of the final results [6].

$$\%E_{r_i} = 100 \times \frac{ActRUL_i - \widehat{RUL_i}}{ActRUL_i} \tag{2}$$

$$A_i = \begin{cases} e^{-\ln(0.5)\cdot(E_{r_i}/5)}, & E_{r_i} < 0 \\ e^{+\ln(0.5)\cdot(E_{r_i}/5)}, & E_{r_i} \geq 0 \end{cases} \tag{3}$$

$$Score = \frac{1}{11}\sum_{i=1}^{11} A_i \tag{4}$$

where $ActRUL$ represents the actual remaining useful life, measured from the test point to the actual point of failure. Moreover, RUL represents the predicted remaining useful life, E_{r_i} is the percentage error between the $ActRUL$ and RUL, and A_i represents the score for each bearing that has been tested. Finally, the score indicates how the overall prediction performs, where higher score values represent higher accuracy.

3.2 Diagnosis for bearing fault classification

The traditional method of determining bearing fault types is by using a FFT. By plotting the bearing FCFs and their harmonics while using the envelope spectrum of the FFT and simply comparing the results to known behaviors can lead to a diagnosis. Example plots can be found in Figure 3. However, the PRONOSTIA dataset was collected at a vibration sampling rate of 25.6 kHz and recorded for 0.1s every 10s. This indicates that each 0.1 s time interval only contains 2560 data points, which is considered scarce for FFT analysis. Moreover, by only using such a small number of data points, the FFT analysis does not deliver a clear output. As shown in Figure 10

(a) and (b), the 20 combined recordings result in a richer FFT representation. The peaks shown in plot (b) are more isolated than the ones shown in plot (a) and a much higher resolution is obtained. Therefore, 20 recordings were combined to provide a richer data stream, totaling 2 s of recorded data with 51200 data points.

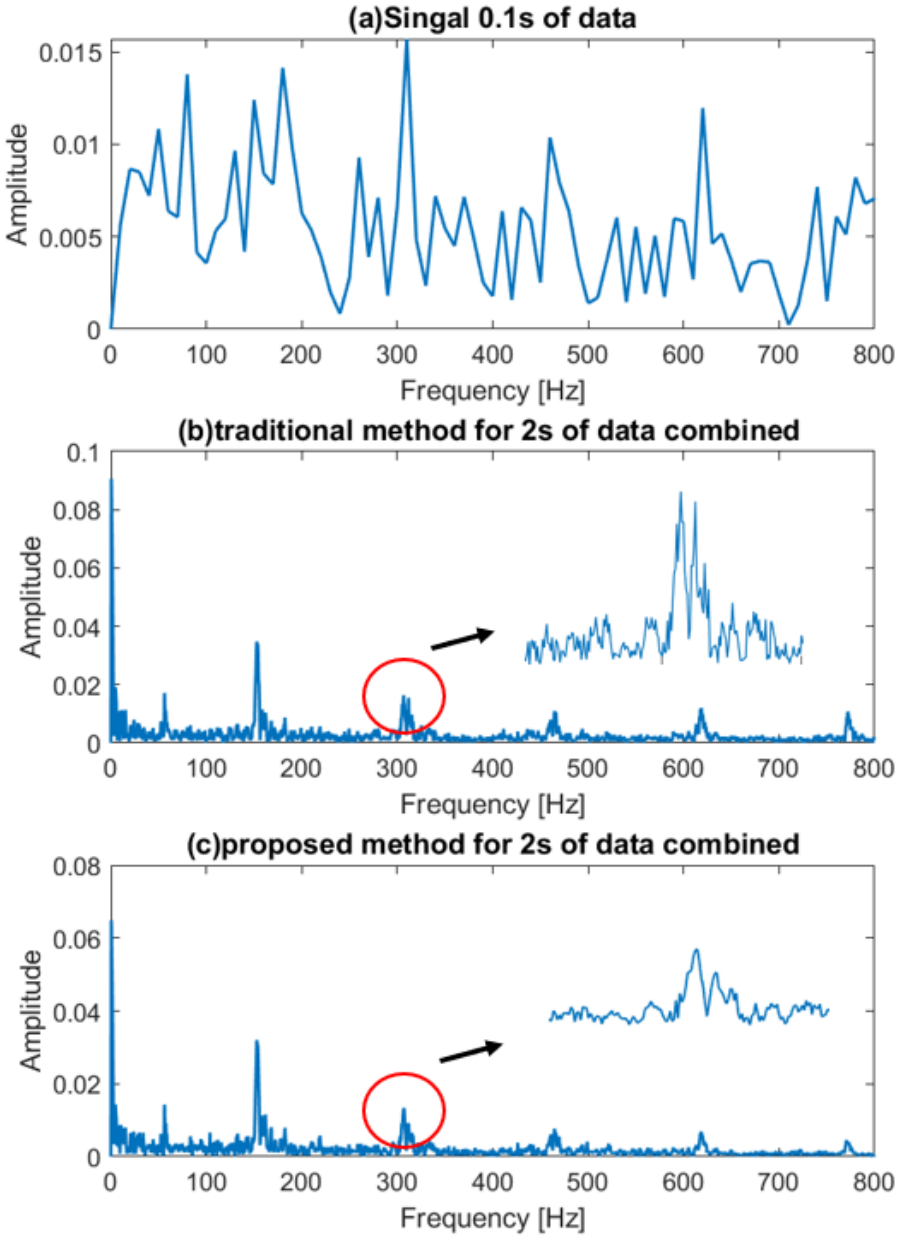


Figure 10 Comparison among (a) FFT of 0.1s of recorded data; (b) traditional FFT of 2s of recorded data; (c) proposed method with additional steps and a FFT of 2s of recorded data

The FCFs are divided into four distinct categories and their mathematical equations are:

$$BPFO = \frac{n \times RPM}{2 \times 60} \times (1 - \frac{d}{D} \times \cos \emptyset) \quad (5)$$

$$BPFI = \frac{n \times RPM}{2 \times 60} \times (1 + \frac{d}{D} \times \cos \emptyset) \quad (6)$$

$$BSF = \frac{D \times RPM}{60 \times d} \times (1 - (\frac{d}{D})^2 \times \cos^2 \emptyset) \quad (7)$$

$$FTF = \frac{RPM}{2 \times 60} \times (1 - \frac{d}{D} \times \cos \emptyset) \quad (8)$$

where n is the number of balls, d is the ball diameter, D is the pitch diameter, and $\emptyset$ is the contact angle.

The PRONOSTIA bearing FCFs are calculated using Equations (5)-(8) using the values provided in Table 2. Results of the FCF calculations are provided in

Table 3, where f_r is the natural frequency of the bearing.

Table 2 PRONOSTIA dataset configuration

n	d (mm)	D (mm)	Rotation speed (rpm)	$\emptyset$
13	3.5	25.6	1500/1650/1800	0

Table 3 PRONOSTIA bearing characteristic faulty frequencies

	Condition 1 (1800 rpm)	Condition 2 (1650 rpm)	Condition 3 (1500 rpm)
f_r (Hz)	30	27.5	25
BPFI (Hz)	221.66	203.19	184.72
BPFO (Hz)	168.34	154.31	140.28
BSF (Hz)	215.33	197.38	179.44
FTF (Hz)	12.95	11.87	10.79

In addition to combining 2 s of signals, a novel diagnosis procedure is used based on the method suggested by Kim et al [13]. In their work, three steps are taken to obtain the final fault frequency plots:

Step 1: AR model

The AR model is a mathematical model for time series data used to predict values based on previous readings.

$$y_p = \sum_{p=1}^{n} \emptyset_p \, x_{t-p} + c \tag{9}$$

where y_p is the predicted value at the nth time step, $\emptyset_p$ is the AR coefficient, x_{t-p} is the $(t-p)th$ data, and c is white noise. Note that c is only going to change the scale of the final results rather than the patterns. Therefore, it has no influence in figuring out repetitive peaks and is ignored in this thesis. $\emptyset_p$ can be determined by many methods. For example, the partial autocorrelation function (PACF) can be used, where it sets a region to decide whether the x_{t-p} value is directly correlated to y_p [69]. The AR model is efficient and simple to use because the only parameter that needs to be optimized is p and several methods can be used to do so. Sawalhi et al. state that

for rolling element bearing analysis, the AR model can be used to separate impulses from the original signal [70]. To optimize the order of p, it is reasonable to consider when the kurtosis of the residual signal is at a maximum. This optimal order of p should be less than the two nearest peaks so that impulses stay distinctive in the residual signal. Finally, the residual signal is obtained after removing the discrete signal from the original signal.

Step 2: SK and STFT used to select the final frequency band

The SK is plotted by applying the STFT to step 1's result. The STFT is used to extract time-frequency domain information for bearings. Instead of using time and frequency as axes, the SK uses the spectral kurtosis value and frequency as axes. The STFT-based estimator of the SK can be written as [20], [45]:

$$K(f) = \frac{\langle |S(t,f)|^4 \rangle}{\langle |S(t,f)|^2 \rangle^2} - 2 \, , f \neq 0 \tag{10}$$

where $\langle \cdot \rangle$ is the time-average operator and $S(t,f)$ is the STFT of the residual signal. A visual representation is provided in Figure 11. If one looks from the top and takes frequency and time as axes, a 2D spectrogram is obtained. If one takes frequency and magnitude as axes, the SK is observed [71]. The SK plot is used to determine the demodulation bands. Since the STFT can be plotted based on various window lengths, the optimal SK can be obtained by comparing different window length-based STFTs. For example, in Figure 12, by using different window lengths (32, 64, 128 and 256), the demodulation frequency band lands in the range of [200 1000] Hz for a maximum SK value (i.e. the determinate fault frequency range).

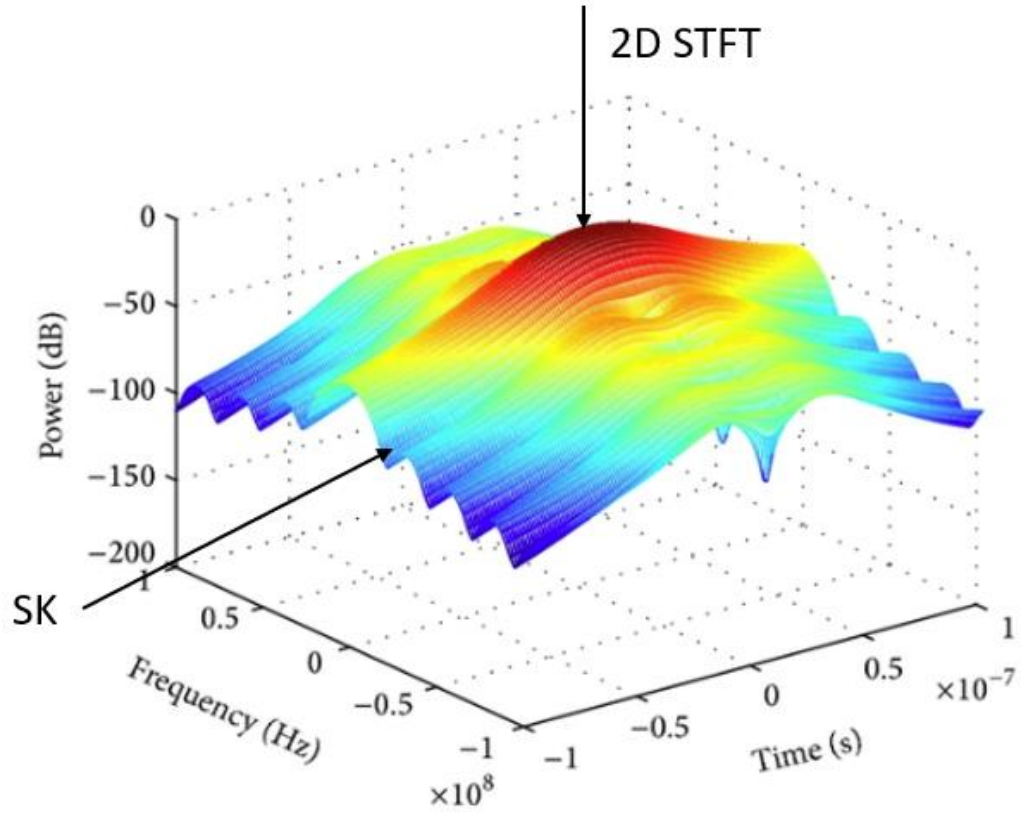


Figure 11 Representation of a 3D STFT (modified from [71])

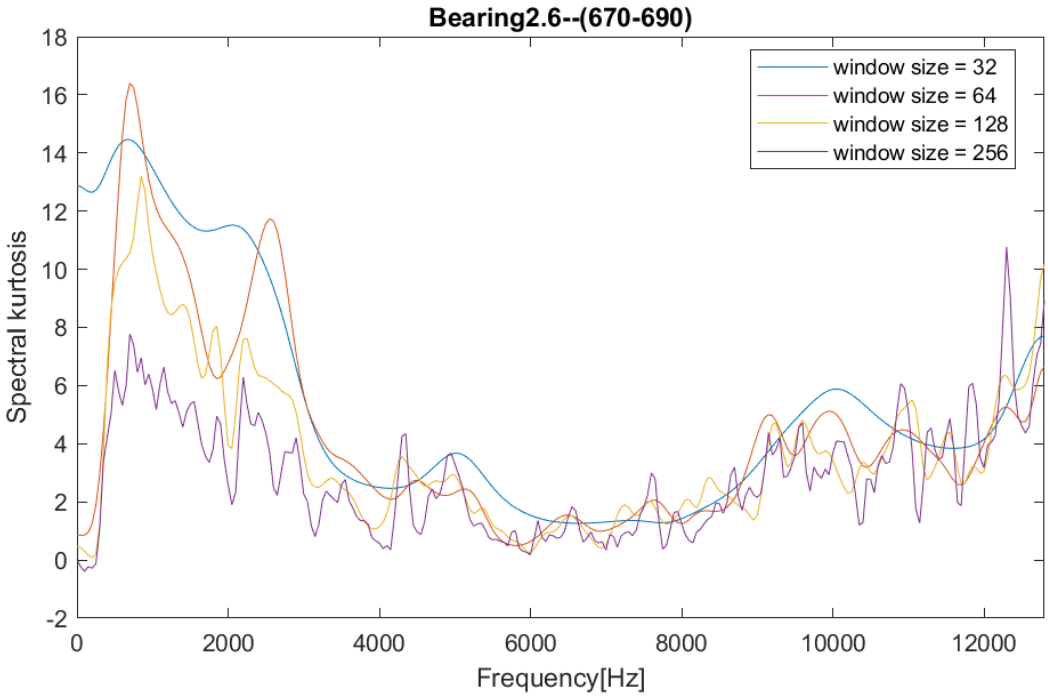


Figure 12 SK plots based on different window lengths

Step 3: HT and FFT used to obtain the final envelope spectrum

Finally, the traditional HT and FFT are applied to the resulting signal. The final plot is shown in Figure 10 (c), showing less noise when compared to plot (b). As shown in the magnified area, not only does plot (b) have more noise in general, but it also contains an extra peak, whereas plot (c) identifies it as less important than the first peak.

Now, combined with precalculated bearing FCFs (drawn as line on the FFTs), the resulting plots can be used to identify bearing fault types. For example, bearing 2_6 is shown in Figure 13. In this case, the solid blue lines are the envelope spectrum, the black dotted lines are corresponding FCFs, and the red dotted lines are the sidebands for the BPFI and BSF, which help in identifying these faults. Note that sidebands for the BPFI have the value of $BPFI \pm MRF$, and sidebands for the BSF are $BPFI \pm FTF$. As the black BPFO lines line up with the envelope spectrum peaks and their harmonics in this example, while no other FCFs show similar alignment, it is safe to determine that bearing 2_6 has an outer race defect.

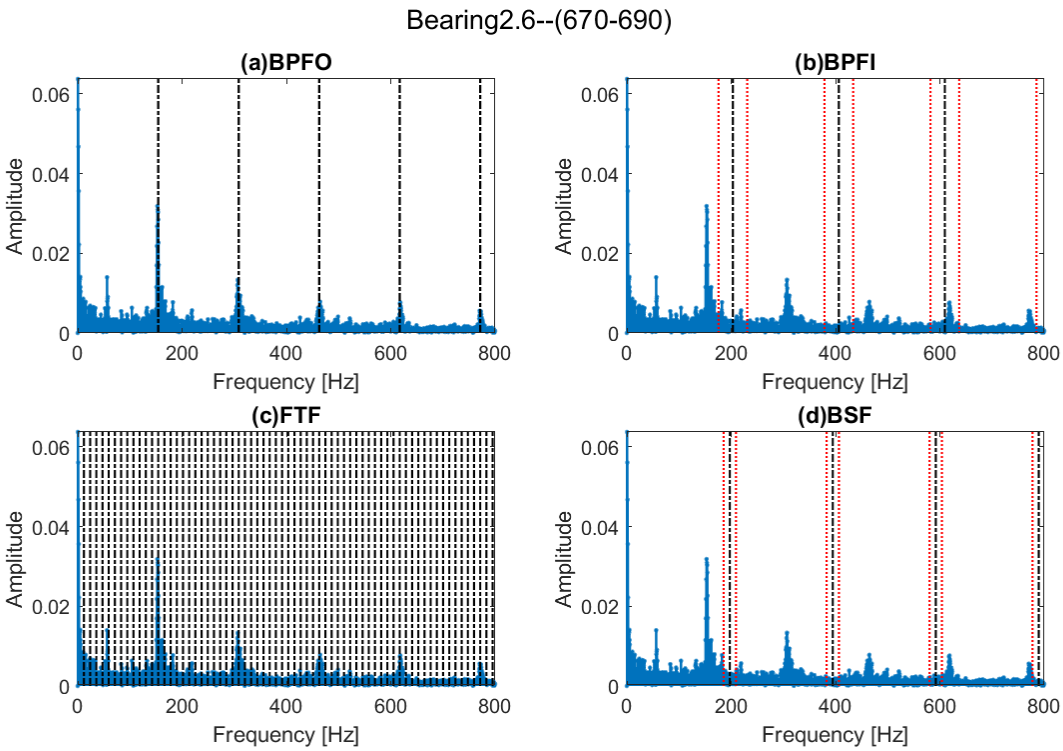


Figure 13 Example diagnosis result of PRONSOTIA data for Bearing 2_6

3.3 Prognosis for RUL predictions

In this thesis, CNNs are used for estimating bearing RUL. Since the PRONOSTIA dataset contains two accelerometers, a CNN is applied to each channel's dataset for comparison, and a channel fusion CNN is also used for overall performance evaluation. In addition, spectrograms and scalograms are used to generate input images for the CNNs. The process flow chart is shown in Figure 14.

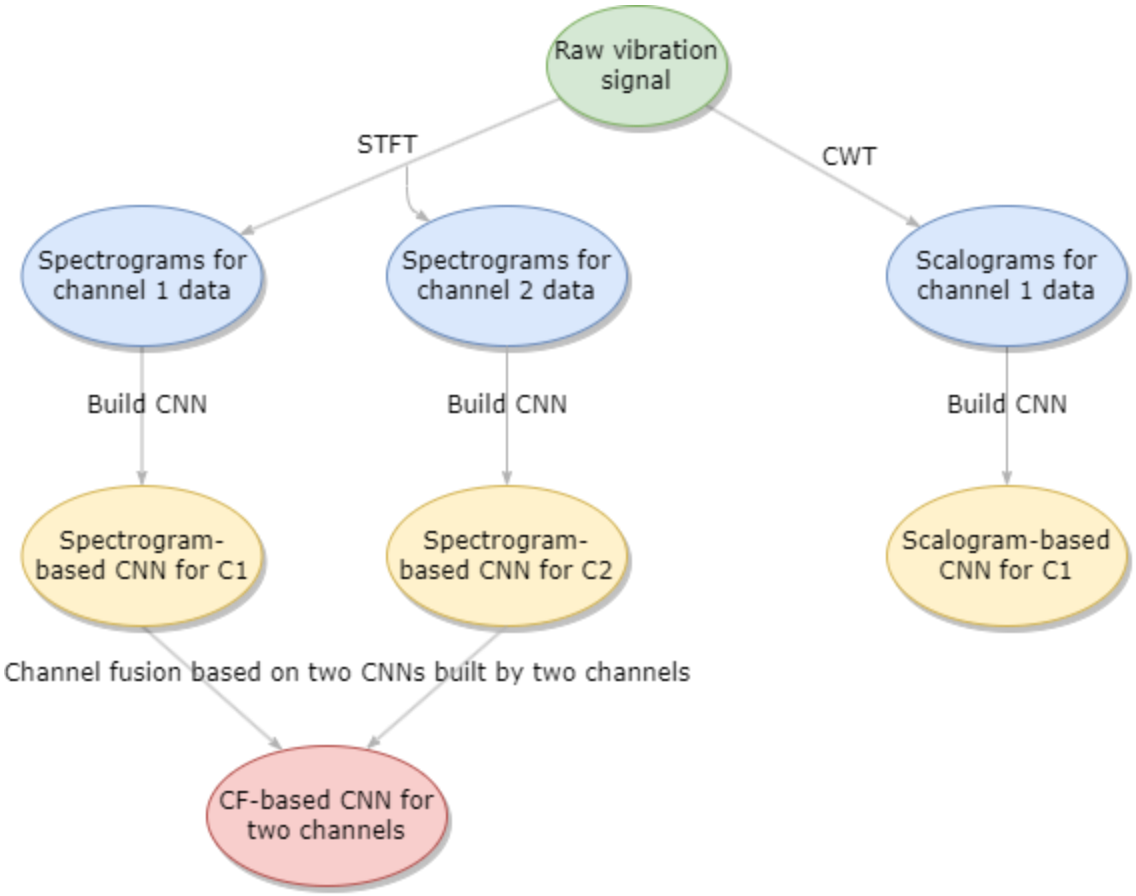


Figure 14 Flow chart of the prognosis procedure used in this thesis

3.3.1 Signal preprocessing

As 2D CNNs typically require their inputs to be 2D images, raw vibration signals must be preprocessed to transform 1D time-series representations into image-like 2D representations. Two preprocessing methods are used in this thesis: spectrograms created using the STFT, and scalograms using the CWT. When using CNNs, standard input images are normally in ".png" or ".jpg" formats, using either greyscale or RBG channels to convert authentic images to mathematical matrixes. But for bearing analysis in this thesis, since preprocessing already converts 1D arrays to 2D

matrixes, the subsequent plots would be matrixes from the outset. Therefore, the STFT and CWT functions are set to provide good representations of the bearing states, as well as suitable output dimensions for a CNN input.

(a) Spectrograms

The STFT is used in scenarios where the FFT provides enough information for expressing bearing health conditions. The STFT is a FT-related method that can determine sinusoidal frequencies and local intensities of a section of the signal as the whole signal changes over time [72]. The principle of how it works is that STFT separates the signal into smaller periodic domains and multiplies it by a window function (e.g. Hann or Gaussian window), then applies the FFT of the resulting section signal. A vivid image representation can be found in Figure 15 [73]. Let $x(t)$ and $X_w(\tau, \omega)$ represent a real sequence and its STFT, respectively. The mathematical equation for the STFT can be expressed as:

$$X_w(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-i\omega t} dt \tag{11}$$

where τ represents the time localization, and $w(t - \tau)$ represents the window function [74]. Note that the only difference between the STFT and the FT is that equation (11) includes a window function "$w(t - \tau)$" when compared to equation (1).

By adjusting the window size and overlap, different time and frequency resolutions can be achieved to provide detailed signal information. More precisely, wide windows lead to poor time resolution but good frequency resolution, and narrow windows perform in the opposite way. Finally, the combined domains can be used as a general signal representing the overall transformation. In addition, by sectioning

signals, non-stationary signals can be analyzed using the STFT as well. The whole process generates a time-frequency domain representation called a spectrogram that visually displays the signal's time and frequency behaviors at the same time.

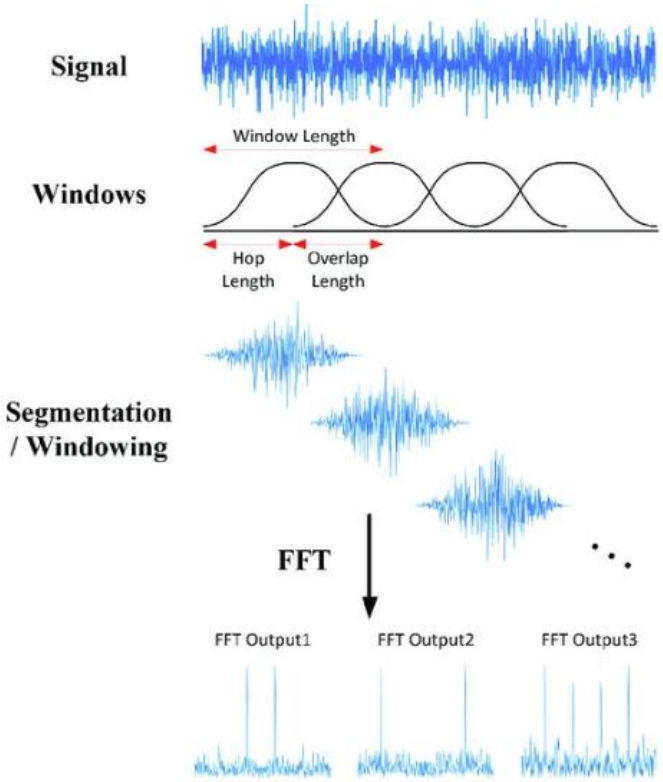


Figure 15 Short-time Fourier transform (STFT) overview [73]

For a CNN to receive an $n \times n$ matrix as input, the STFT window is chosen as "tukey" with a shape parameter of 0.25. The length of each segment, nperseg, is chosen to be 22 and the number of points to overlap between segments is chosen to be 0. Also, the length of the FFT is chosen to be 230. As a result of using these values, a final matrix of 116×116 is achieved. A 116×116 input matrix carries enough information to describe bearing behaviors, and is small enough for CNNs to be processed over a reasonable time period.

A visual representation of bearing 2_3's health state using a spectrogram can be seen in Figure 16. The percentage numbers on the top represent the bearing state and go from healthy (100%) to total failure (0%), and the red circled areas are the indicators of potential frequency and time ranges of failures. Note how distinct the spikes appear and grow as the bearing begins to fail. A raw vibration signal is plotted in Figure 17. In this case, the same periodic peaks also show up at the end, corresponding to a similar trend shown in Figure 16. However, it is clear that spectrograms provide "spikes" that are better defined and cleaner compared to only using time series vibration data. Spectrograms also indicate the frequency range and "spikiness" by color intensity. Therefore, spectrograms are shown to be sufficient in representing the bearing's health state throughout its whole lifetime, while also providing frequency information that raw vibration signals do not contain. A complete set of spectrogram plots for all bearings can be found in Appendix A .

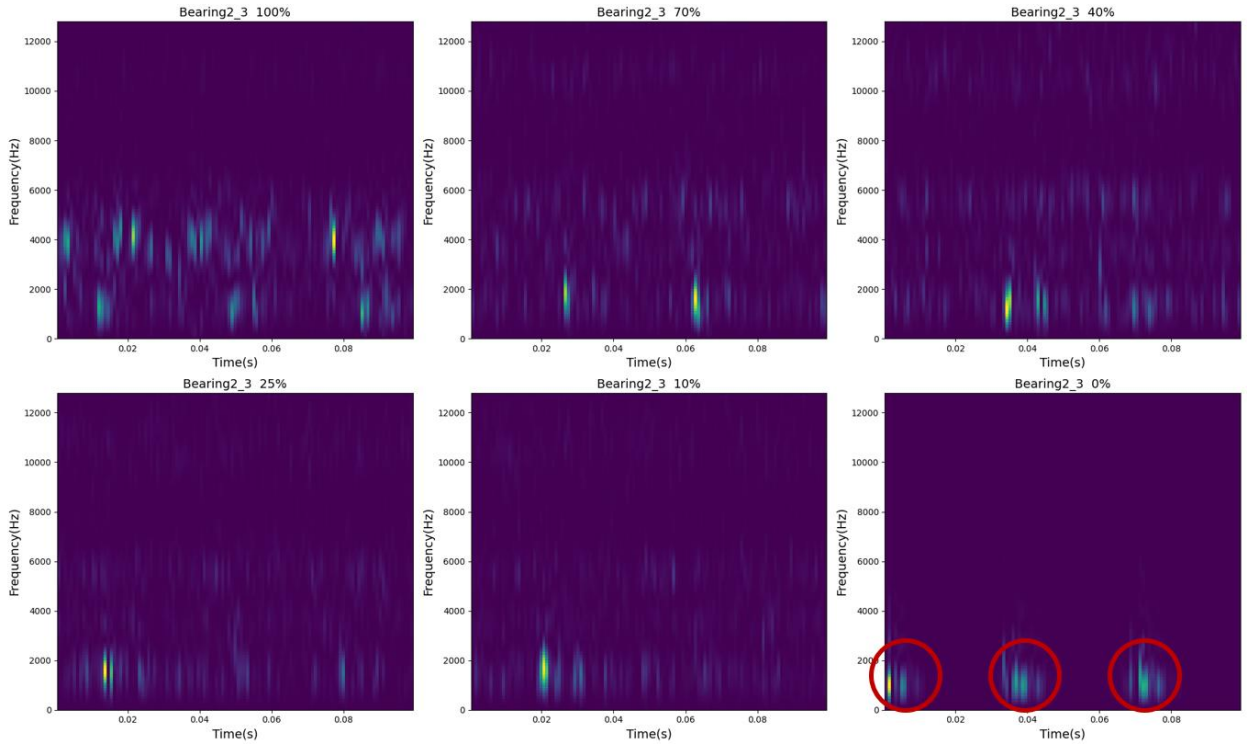


Figure 16 Example of a set of spectrograms for the whole life of Bearing 2_3

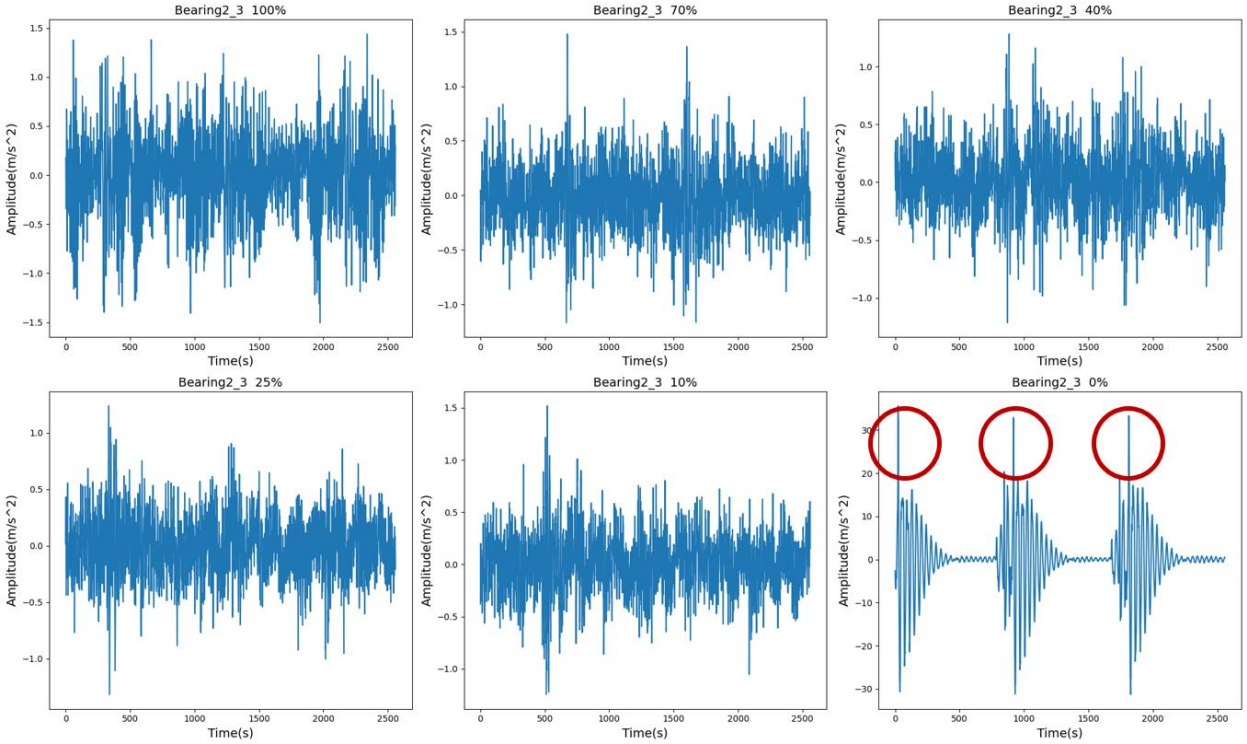


Figure 17 Raw vibration signal for the whole life of Bearing 2_3

(b) Scalograms

There are two main WT methods: the continuous wavelet transform (CWT) and the discrete wavelet transform (DWT), and both work based on the principle that pre-set wavelets are moved along the original time-series signal. The results of the convolutional multiplications form a time-frequency domain representation, which is also called a scalogram. The visual difference between a spectrogram and a scalogram can be seen in Figure 18 [75]. Since scalograms can decompose signals at different frequency and time resolutions, Figure 18(c) shows that the signal has high vibration intensities in the 0.031 to 0.086 Hz region between 100 and 200s, but Figure 18 (b) can only indicate a constant intensity of signal in the 0 to 0.1 Hz region from 0 to 200s. The ability of a scalogram to analyze signals at different frequency and time resolutions can benefit non-stationary signal processing.

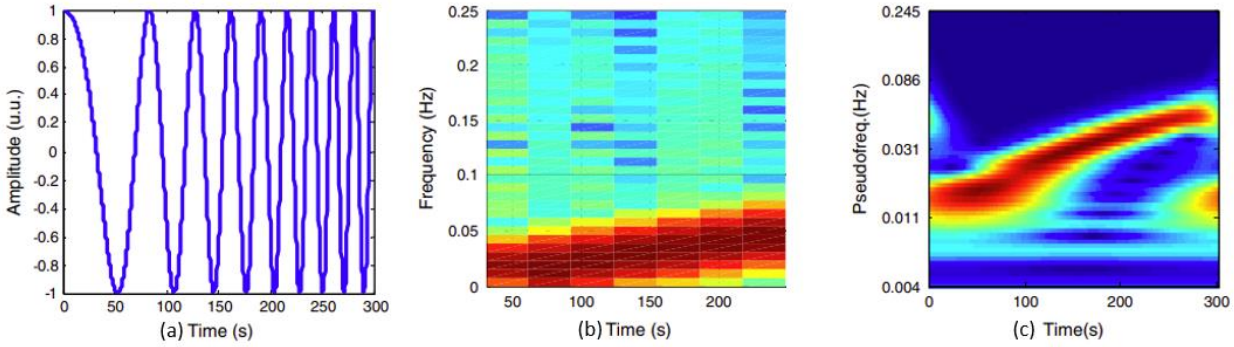


Figure 18 Chirp signal represented as (a) the original time domain waveform (b) a spectrogram (c) a scalogram [75]

The mathematical expression for a CWT is [76]:

$$X(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} x(t) \psi^*(\frac{t-\tau}{s}) dt \tag{12}$$

where, s is a scale parameter (1/frequency) and $\psi^*(\frac{t-\tau}{s})$ is the complex conjugate of the wavelet ψ. Note the difference between the CWT (equation (12)) and the STFT (equation (11)) is that instead of multiplying by an exponential $e^{-i\omega t}$, $x(t)$ is multiplied by $\psi^*(\frac{t-\tau}{s})$ (i.e. the complex conjugate of the wavelet ψ). Moreover, instead of frequency ω being one of the parameters, s (the inverse of frequency) is used in the equation. The wavelet is some basic function, just like the sinusoid is a basic function for the FT acting as window function, but with more complex shapes (e.g. Morlet and Mexican Hat wavelets, etc.). By increasing s, the wavelet can be expanded to help solve low-frequency signals with bad time resolution, and by decreasing s, the wavelet can be shrunk to help solve high-frequency signals with good time resolution. As τ helps shift the wavelet along the original signal $x(t)$ and s helps scale the wavelet, the output $X(\tau, s)$ (wavelet coefficient) reaches its maximum value if the wavelet's frequency matches with part of the original signal's frequency.

The mathematical expression for the DWT is [76]:

$$D(\tau, s) = \frac{1}{\sqrt{s}} \sum_{m=0}^{p-1} x(t_m)\, \psi\left[\frac{t_m - \tau}{s}\right] \tag{13}$$

where $\tau = k2^j$ and $s = 2^j$ are now dyadic, with j representing the scale index and k representing the wavelet transformed signal index. Note that instead of integrals $\int$, a sum $\sum$ is used to represent the idea of discreteness, note in that in the CWT, $s_{CWT} =$

$2^{1/v}$, where v is an integer greater than one and can be referred to as "voices per octave". Compared with $s_{DWT} = 2^j$, the DWT and CWT differ based on how they discretize the scale and the translation parameters, where the DWT normally uses an exponential scale with a base of two. In contrast, the CWT mostly uses an exponential scale with a base smaller than two [77]. This leads to finer discretized scales for the CWT when compared to the DWT. Therefore, if one calculates the wavelet coefficients at every possible s and τ, tremendous data will be generated. As such, by setting s and τ to be discrete, the amount of data generated by the DWT can be limited, resulting in an efficient and computationally friendly process.

Like the spectrogram, the scalograms also represents time and frequency domain signals at the same time. However, since the scalogram can deliver information based on different time and frequency resolutions, it is able to detail more accurate behaviors. In this thesis, the "Morlet" wavelet is used as the mother wavelet, and the wavelet scale is chosen to be 129. The resulting scalogram is a 128×128 2D matrix, as shown in Figure 19. Using a scale of 129, the 128×128 matrix carries enough information and remains small enough for CNNs to process in a reasonable time span.

Since each plot has their own color map scaling, it is hard to tell the trend based on a whole bearing life perspective. Therefore, the color bar is unified by using the maximum value that appeared during the whole bearing life instead. Figure 20 is created to represent the bearing behavior from a more general perspective. Note that the y-axis is in log scale, and the log-scaled frequency range can be used to emphasize behaviors that happen in certain frequency regions. From this, it is clear that the

defect frequency happens around 800Hz, which is hard to tell in the spectrograms. However, both the scalograms and spectrograms can indicate the periodic appearances.

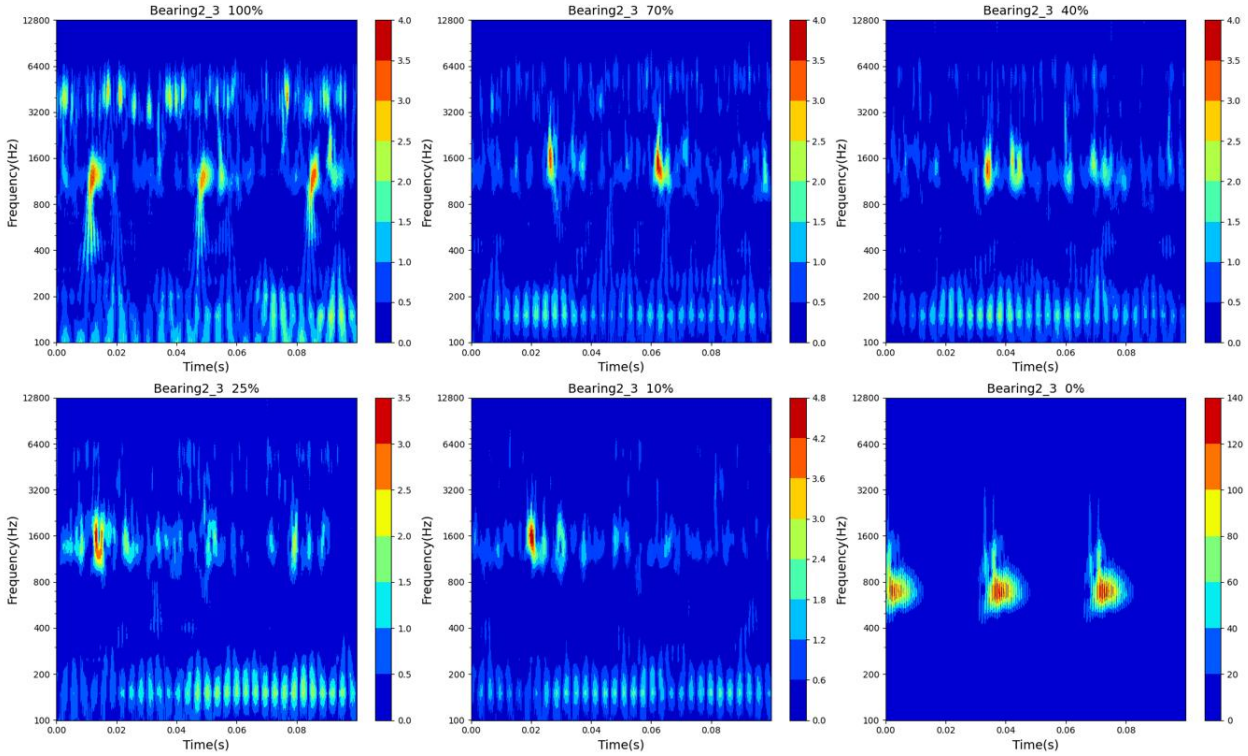


Figure 19 Example of a set of scalograms for the whole life of Bearing 2_3

In conclusion, by using spectrograms and scalograms, a defect frequency can be determined. For example, bearing2_3 has a defect frequency of around 800Hz that appears three times in 0.1s. A sudden failure is assumed based on the comparison of listed plots, shown in Figure 20.

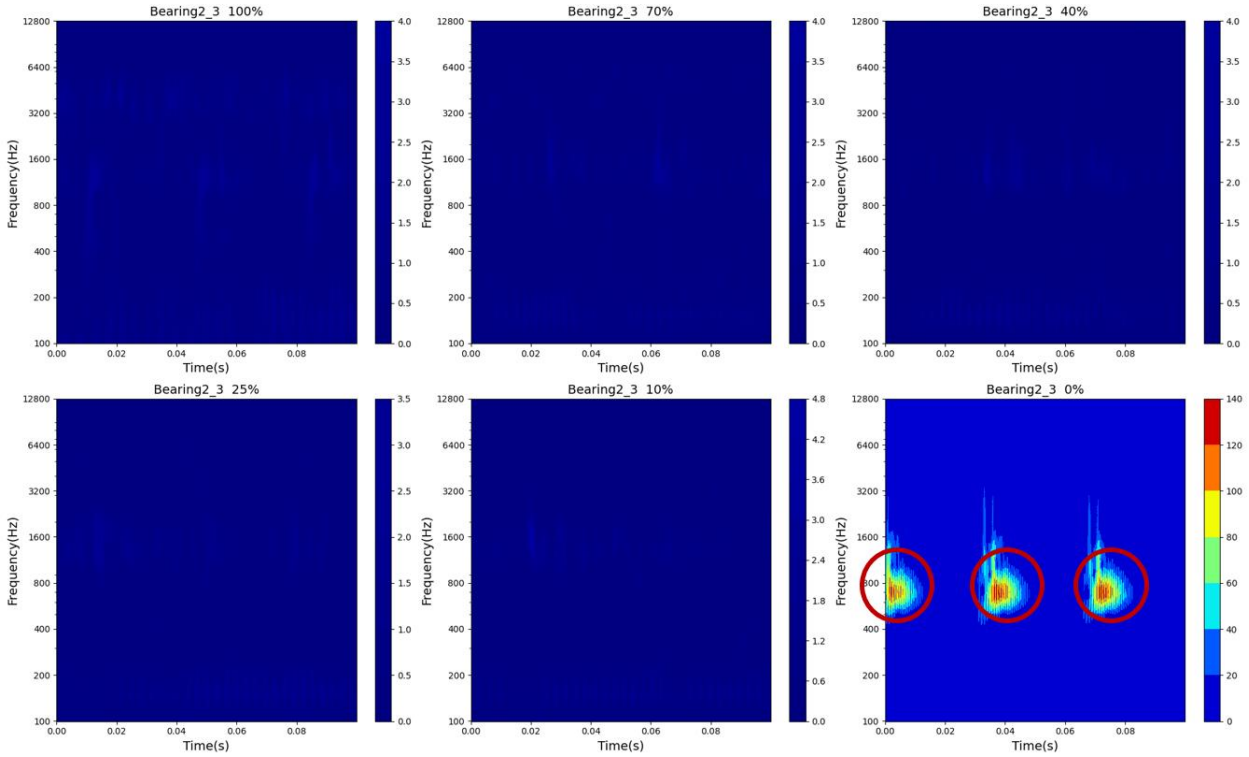


Figure 20 Unified color map based on Figure 19

Note that certain resolutions and image sizes for both spectrograms and scalograms are chosen based on a trial-and-error approach that picks the best fit images for the CNN to process. These resolutions are defined by the parameters chosen when applying the STFT and the CWT, for example the window length and wavelet scale. The parameters are chosen to serve as both a good demonstration of the bearing's health state, and to be readable by the CNN. The images sizes are also chosen to deliver both informative results and maintain sizes that can be processed by the CNN in a reasonable time by the equipment available.

3.3.2 CNN parameters

CNNs work on the principle that the algorithm learns patterns from a training dataset. These patterns then provide labels for identifying similar patterns later. By comparing test data patterns with learned labels, CNNs can determine whether test data belongs to a specific training set category. For example, if a CNN is trained with images labeled as "cats", any image input after that would be compared to the labeled cat images. If the images are found to share similarities, then they are likely to also be given the same cat label. To identify similarities, a CNN must extract features from the input images, a process that is explained in the following paragraphs.

As seen by a computer, any colored image combines three digital channels representative of the primary additive colors: red, green, and blue. Alternatively, colored images can be represented in grayscale using only one digital matrix. By transferring images to meaningful digits, CNNs can use mathematical processes to alter and reconstruct original images, thereby condensing the information. CNNs use convolutional layers and pooling layers as hidden core layers to learn input data patterns.

Convolutional layer:

The algorithm generates a kernel as a smaller matrix used to extract different features in a convolutional layer. As shown in Figure 21, a patch of the input matrix is multiplied by a kernel, forming an individual result as an element in the output matrix. By moving the patched section, the final output matrix is filled with processed

input data. Different features can be extracted from the input image depending on the kernel's settings.

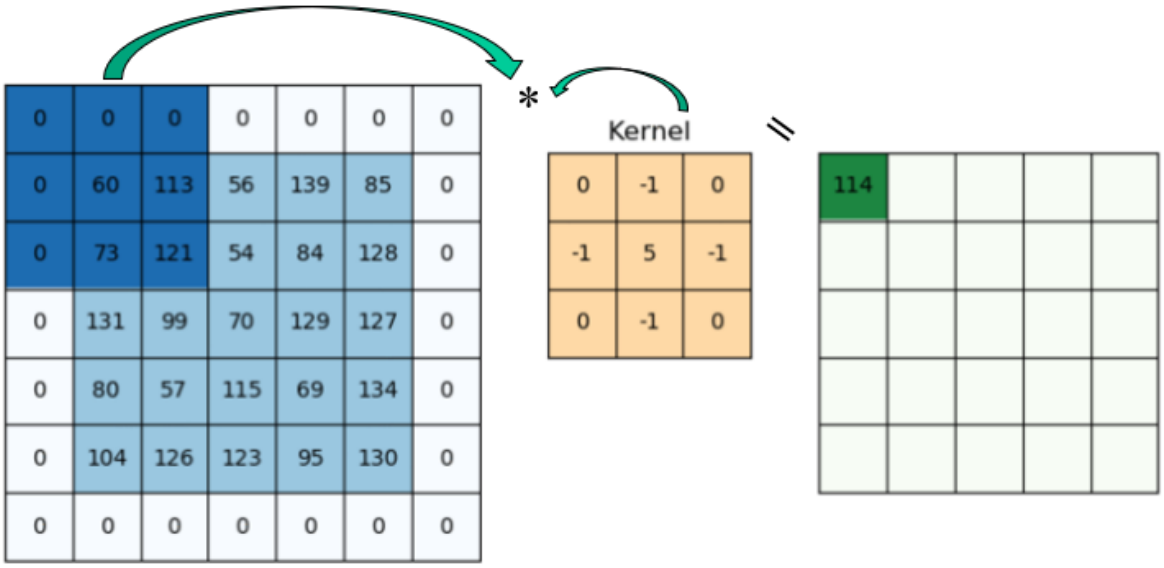


Figure 21 Example of a kernel-based filtering process in a convolutional layer (modified from [78])

Several convolutional layers terms must be defined to explain further:

(a) Filters and kernels

Kernels are small matrices, as shown in Figure 21, used to determine certain patterns in images. Each kernel can detect different patterns, such as horizontal edges or more complex objects, depending on the depth of the CNN. They can have square or normal rectangular shapes for 2D CNNs, and they have the same number of channels as the input. Numbers inside the kernel can be chosen by the CNN itself, and by using transfer learning, other CNN kernels can be used for new data sets. Filters, on the other hand, determine how many kernels are used in a single convolutional layer. Therefore, in a single convolutional layer, multiple patterns can

be detected. In many tutorials and papers, the use of the terms "filters" and "kernels" are often confounded. However, in this thesis, the terms are used distinctly as defined.

(b) Stride

Stride is how many steps occur each time a kernel convolves. For example, for a 2D CNN, a stride of size (1×1) means the kernel moves to the right or down by one element. Mathematically, with no padding, a 2D CNN with input size of $(n \times n)$, kernel size of $(f \times f)$, and a number n_c of filters, the convolving output is calculated to be:

$$((n - f + 1) \times (n - f + 1) \times n_c)) \tag{14}$$

Note that normally, a stride step has the same width and height. However, in some cases it can be different and s_w and s_h are denoted specifically for those cases. In addition, stride can also be applied in 3D CNNs as a 3D matrix with a representation of (s_w, s_h, s_d), where s_d represents the depth along the 3rd dimension.

(c) Padding

As stated in (b), convolving kernels on images essentially shrinks the output, and if the input is small, then only a few kernel convolving processes can be performed. Also, the very edge and corner elements in any image only get used a few times when convolving compared to other elements in the middle area. Therefore, much of the information near the edge or corners of the image is thrown away. So, padding is introduced to solve this problem by adding "borders" to the original image. The "borders" are empty elements that do not affect the final output, but help save edge information. The character p defines the amount of padding added on the border.

Therefore, the output can be written as $(((n + 2p - f + 1) \times (n + 2p - f + 1)) \times n_c)$. In this thesis, no padding will be used since the input size is large enough, and the CNN is not so deep that an image would shrink too much.

Considering all factors above, a 2D convolutional layer can be defined as:

$$(\left(\left(\frac{n+2p-f}{s}\right) + 1\right) \times \left(\left(\frac{n+2p-f}{s}\right) + 1\right) \times n_c) \quad (15)$$

where (s, s) is the stride step and n_c represents the number of filters used in each step.

Pooling layer:

A pooling layer is then applied to the output of the convolutional layer. The reason for introducing a pooling layer in a CNN is that pooling layers can help reduce the size of the image and speed up the computing process. Moreover, some feature extraction becomes more robust by using pooling layers. For example, with max pooling, the larger number that gets extracted and that gets added to the algorithm generally represents a more distinct behavior in that area, potentially the desired feature of interest.

The two main types of pooling layers are the max pooling layer and the average pooling layer. Both pooling layers follow a similar application concept: a chosen number is used to represent each patch of the original input matrix. For example, Figure 22 shows that if the patch size is set to be (2×2) with a stride of two, the largest number in the original matrix patch is chosen to be the output element in the max pooling layer. In contrast, the average patch matrix value is used in the average pooling layer. Different messages can be sent to the next level of analysis by applying

different pooling methods. However, they all serve to reduce the original matrix size. Due to this size reduction process, noise is reduced, and the convolutional layer's extracted information is condensed. Similar hyperparameters are used in pooling layers (i.e. f is the filter size and s is the stride). The pooling output size can be calculated using equation (16):

$$((\left(\frac{n-f}{s}\right)+1)\ \times(\left(\frac{n-f}{s}\right)+1)\times\ n_c) \tag{16}$$

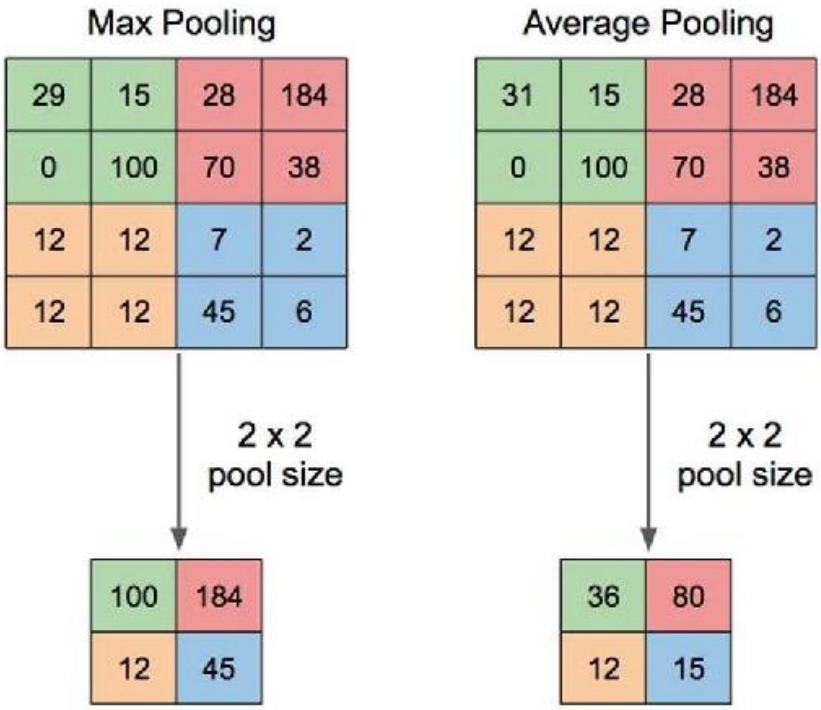


Figure 22 Example of a max pooling layer (left) and an average pooling layer (right) [79]

Flatten layer and dense layer:

The last steps of any neural network are to predict or classify the output of hidden layers with "labels", which are normally stored in 1D arrays for simplicity. Since the

outputs of a convolutional layer are multidimensional matrices and pooling layers do not change the nature of these outputs, a flatten layer is used to transfer multidimensional information into a 1D array representation. The flatten layer is usually set before the dense layer so that the dense layer can have a 1D array as input. Finally, the dense layer is used to compare the outputs with labels to predict or classify.

Dense layers are often used in common neural networks. In this case, they are simply layers where each neuron receives an input from every neuron in the previous layer. The output then becomes a result of previous weighted inputs. For example, if the previous layer is presented as a four-neuron layer ($x_1,\ x_2,\ x_3,\ x_4$) and the dense layer is presented as a two-neuron layer ($y_1,\ y_2$), with different weights w and a bias b, their relation can be expressed as:

$$y_1 = w_{1\_1}x_1 + w_{1\_2}x_2 + w_{1\_3}x_3 + w_{1\_4}x_4 + b_1 \quad (17)$$

$$y_2 = w_{2\_1}x_1 + w_{2\_2}x_2 + w_{2\_3}x_3 + w_{2\_4}x_4 + b_2 \quad (18)$$

Note that weights w and bias b are trainable parameters. Therefore, for this layer, there are ten trainable parameters. By using different weights and biases, weighted information can be transferred into the next layer. Key input data patterns can be retrieved by combining multiple convolutional layers and pooling layers. In the end, dense layers (or fully connected layers) are used to connect each input to each output for categorizing important features and make predictions based on pre-set labels.

Activation function:

The connection between two layers can be given as a linear expression:

$$Y^{[n]} = W^{[n]}X^{[n]} + B^{[n]} \tag{19}$$

where $Y^{[n]}$ is the output of the n^{th} neuron, $X^{[n]}$ is the input from the previous neuron, $W^{[n]}$ and $B^{[n]}$ are the weights and biases for this connection, respectively. As it is a linear expression, no matter how many layers are included, the final output would be a linear expression of the input, which is not the case for real-life problems. Therefore, introducing non-linearity between steps is essential, and the way of doing this is by using an activation function in each layer. The most basic activation functions include the sigmoid, tanh, and relu activation functions. Their mathematical expressions are provided in equations (20) to (22), respectively, and their plots are shown in Figure 23.

$$G(z) = \frac{1}{1 + e^{-z}} \tag{20}$$

$$G(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{21}$$

$$G(z) = \max(0, z) \tag{22}$$

where z is the input and $G(z)$ is the output after applying the activation function.

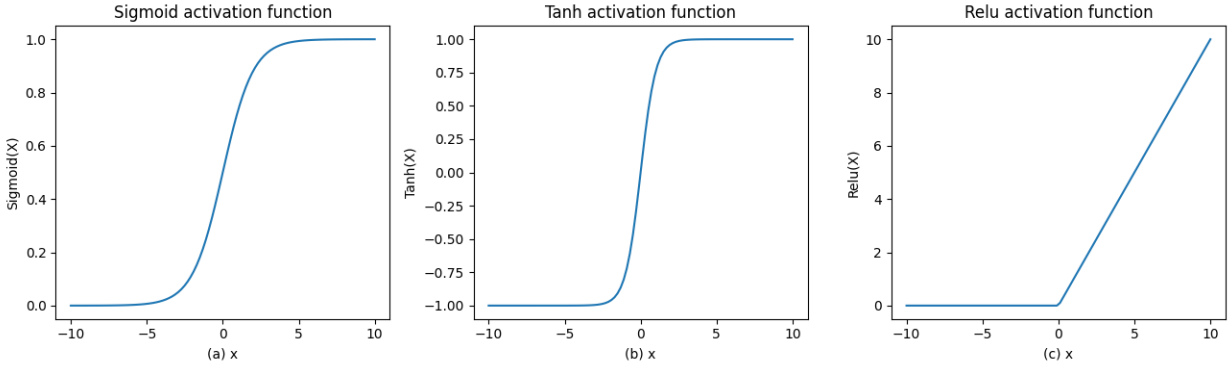


Figure 23 Activation function plots

After applying the activation function, the outputs keep their characters, meaning they still carry specific pattern information. To better understand this concept, consider the sigmoid function and take z as a very large number when compared to other inputs. The activation function output is in the range of $(0, 1)$, so a large number in that range would be close to one, which is also the case after the activation function is applied. Therefore, the character of "being a large number in its range" stays, but its linearity is removed. This linearity is now more closely expressed as a binary function. As a result, the extracted features are more distinctly divided to better serve the purpose of expressing feature identities.

The sigmoid activation function is mostly used in binary classification problems since its value is in the range of $(0, 1)$ and best describes whether a prediction is zero or one. The tanh activation function is mostly used in RNNs, and relu is the most common activation function used for neural networks as it performs the best for most cases. The relu function makes any negative input zero, and to perform such an easy task, it requires less computing power, making it very fast to train with.

3.3.3 Health indicator (HI) for unsupervised learning

Unlike other classification CNNs containing distinct categories for predictions to land on, predicting a bearing's RUL has no pre-defined outcome. This represents an unsupervised learning process. Therefore, instead of manually inputting labels for distinct features, the algorithm must determine another way of presenting health states along the whole life of a bearing's degradation. A simple way of developing

labels is by introducing a HI. By analyzing the original data sets, the first and last recordings of the bearings can be retrieved, meaning the bearing start and end health states can be assumed, then a linear function can be used to represent the whole degradation process. Specifically, the first recording point represents the healthiest state of the bearing, and the last recording point represents the least healthy state, which are assigned to be one (100% health) and zero (0% health), respectively. Any recorded point in between is assigned a percentage representing its health state. In Figure 24, the x-axis represents the total number of recordings for bearing 1_1, and the y-axis represents the HI values for the bearing's health state. Although it seems cursory, this kind of setting method can be used for scenarios where the real degradation trend is hard to collect. Finally, the basic label setting task for bearing RUL is complete.

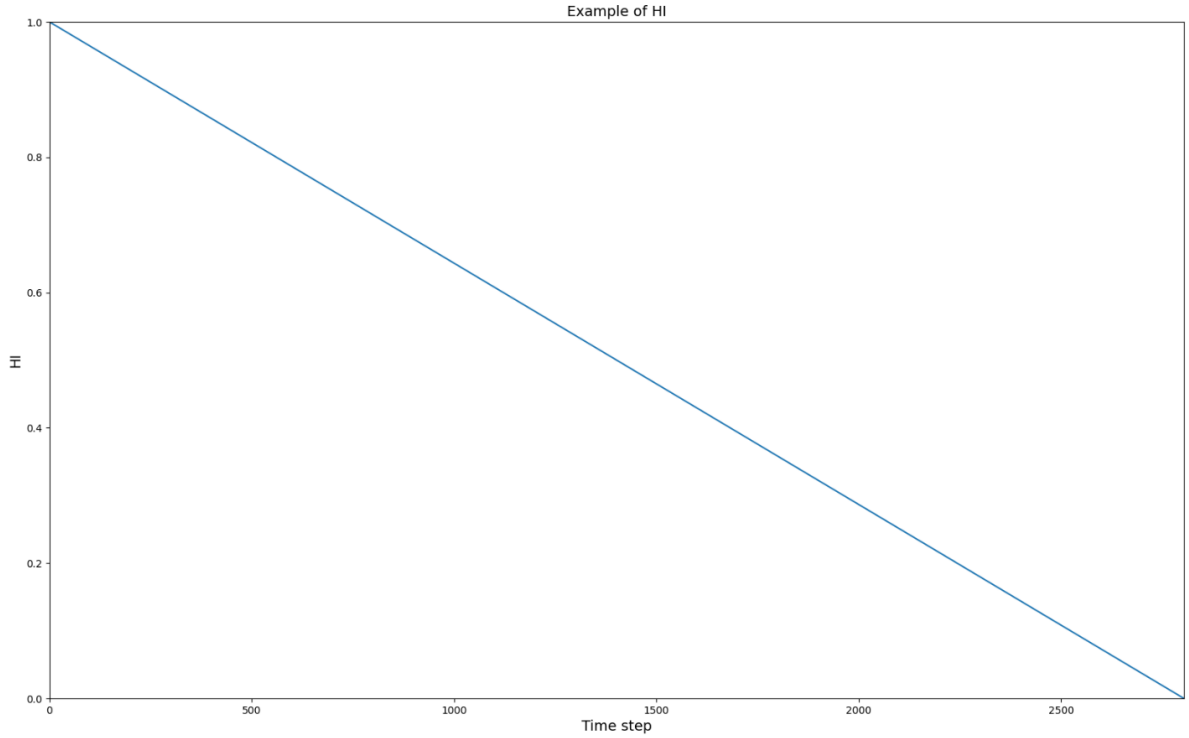


Figure 24 Example of a HI setting for Bearing 1_1

3.3.4 CNN construction

After creating spectrograms and scalograms in matrix form (as in section 3.3.1), the input size for the CNNs used in this thesis is 116×116 and 128×128, respectively, for each 0.1s of recorded data. The sizes of the inputs are chosen for the simplicity with which they govern the size of the CNN plots. These input sizes can transfer clear information and are a reasonably small size for CNNs to compute. A bigger matrix contains more information, but the computing power required for bigger CNN inputs can be difficult to manage with the equipment available.

Note that data for bearing 1_4 is missing in the original dataset used for this thesis, and was not found to be available during this project. Moreover, each test set is used directly from the original test file, which represents the time span from healthy to when the PRONOSTIA testers decide to pause the recording. However, for bearing 2_4 and bearing 2_7, each of their sample sizes are added to be 95% of their real life to present how final prediction accuracy can change with the amount of data used as input to the CNN. Therefore, the final data set is split into train sets and test sets as described Table 4.

Table 4 PRONOSTIA dataset train/test split

	Condition 1	Condition 2	Condition 3
Train set	Bearing 1_1	Bearing 1_1	Bearing 3_1
	Bearing 1_2	Bearing 2_2	Bearing 3_2
Test set	Bearing 1_3	Bearing 2_3	Bearing 3_3
	Bearing 1_3_95p	Bearing 2_4	
	Bearing 1_5	Bearing 2_5	
	Bearing 1_6	Bearing 2_6	
	Bearing 1_7	Bearing 2_7	
		Bearing 2_7_95p	

To demonstrate the process, only one CNN model built with spectrograms using channel 1 data will be demonstrated. Other CNNs used in this thesis follow similar logic but with some alterations in hyperparameter choices.

By combining all recording files, the train set and test set are built as two 3D arrays with sizes of $(7534, 116, 116)$ and $(15927, 116, 116)$ respectively, where the first number represents the total number of files contained in a given data set. Training set labels are created using a HI for each bearing. The CNN is then programmed to have six 2D convolutional layers and three fully connected layers. The mathematical computation process for each layer is detailed as follows. Note that the reason for choosing certain hyperparameters will be explained later.

(1) Every 2D input matrix having a size of (116×116) (i.e. a 2D spectrogram) is multiplied by a set of filters with a kernel dimension of (5×5), and a total of 32 filters are used in the first layer. The stride is set to be $(1,\ 1)$, meaning that the kernel is convolved on only one element at a time. The first output shape can be calculated using equation (15) to be $(112 \times 112 \times 32)$. Then, another

convolutional layer is added with the same hyperparameters. The output of the first two convolutional layers measures $(108 \times 108 \times 32)$.

(2) The max pooling layer is then applied to the output of the convolutional layers. The hyperparameters of the max pooling layer are set to be a pool size of $(2, 2)$ and a stride of $(2, 2)$. By using equation (16), the final output of the first max pooling layer is $(107, 107, 32)$.

(3) Steps (1) and (2) are repeated three times before going into the flatten layer, where all the output parameters are flattened out to be a 1D series with a dimension of 51200. Note that after each combination of steps (1) and (2), batch normalization is applied to prevent gradient explosion.

(4) Lastly, three dense layers are applied for the final prediction and the predictions are plotted.

(5) The final prediction plot is smoothed by using a moving average to indicate the whole bearing behavior and help in future forecasting.

A summary of the CNN architecture is provided in Figure 25.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 112, 112, 32)      832

 conv2d_1 (Conv2D)           (None, 108, 108, 32)      25632

 max_pooling2d (MaxPooling2D  (None, 107, 107, 32)     0
 )

 conv2d_2 (Conv2D)           (None, 103, 103, 32)      25632

 conv2d_3 (Conv2D)           (None, 99, 99, 32)        25632

 max_pooling2d_1 (MaxPooling  (None, 49, 49, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 45, 45, 128)       102528

 conv2d_5 (Conv2D)           (None, 41, 41, 128)       409728

 max_pooling2d_2 (MaxPooling  (None, 20, 20, 128)      0
 2D)

 flatten (Flatten)           (None, 51200)             0

 dense (Dense)               (None, 128)               6553728

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 7,152,033
Trainable params: 7,152,033
Non-trainable params: 0
```

Figure 25 CNN architecture for spectrogram-based channel 1 datasets

The CNNs are built in Python using the "keras-tensorflow" package. This package is designed specifically to build and modify CNNs for Python users. It allows the construction of any CNN by picking the appropriate parameters. Optimal hyperparameters for CNNs are chosen by applying a built-in hyperparameter tuner

package (i.e. RandomSearch in tensorflow-keras), as shown in Figure 26. The package is used to run all combinations of pre-set hyperparameters, and each combination goes through the normal CNN process three times to evaluate the final accuracies. The whole process is executed 30 times to average the results and eliminate any unexpected outlier data. After 30 trials, the ten best combinations of hyperparameters are listed, and the best combination is shown in Figure 27, with the lowest validation loss found to be 0.0405. Complete codes for each CNN architecture are provided in Appendix A . The total amount of computation time is approximately five hours.

```
lr = hp.Choice('lr', [0.001, 0.005, 0.01])
cnn_filters_1 = hp.Choice('cnn_filters_1', [16, 32])
cnn_filters_2 = hp.Choice('cnn_filters_2', [32, 64])
cnn_filters_3 = hp.Choice('cnn_filters_3', [64, 128])
fc_units_1 = hp.Choice('fc_units_1', [256, 128])
fc_units_2 = hp.Choice('fc_units_2', [128, 64])
kernel_size = hp.Choice('kernel_size', [3, 5])
bn = hp.Boolean('bn')
activation = "relu"
tuner = kt.RandomSearch(hypermodel=build_model,
                        objective="val_loss",
                        max_trials=30,
                        executions_per_trial=3,
                        overwrite=True,
                        directory="SearchResults",
                        project_name="Spectrogram_C1")
```

Figure 26 Specific codes used to find the optimal hyper-parameters for building the spectrogram C1-based CNN

```
Best val_loss So Far: 0.0405295963088671 34
Total elapsed time: 03h 09m 16s
Showing 10 best trials
<keras_tuner.engine.objective
Trial summary
Hyperparameters:
lr: 0.001
cnn_filters_1: 32
cnn_filters_2: 32
cnn_filters_3: 128
fc_units_1: 128
fc_units_2: 64
kernel_size: 5
bn: False
Score: 0.040529596308867134
```

Figure 27 Best hyperparameter values for a spectrogram-based CNN for channel 1 data

3.3.5 ARIMA model

Results from section 3.3.4 are once more processed for the purpose of denoising and smoothing in an effort to improve further procedures. The methods used are the polynomial regression and the moving average. Their differences will be detailed in section 4.2.1.

Given the CNN results, the bearing RUL can be estimated using a time series forecasting method called ARIMA.

The ARIMA model consists of three main parts:

(1) Autoregression (AR)

Used to predict future values based on past time series values. AR assumes that future values are related to past values, and therefore, uses the behavior of past values to predict future values.

(2) Moving average (MA)

Used to improve prediction accuracy based on errors. Similar to AR, it assumes that future values connect with past errors.

(3) Integration (I)

Used to make time series data stationary. When time series data is stationary in time, the data has consistent statistical properties over time. Therefore, the stationary data is easier to predict.

When only AR and MA are introduced, they become an ARMA model that deals with stationary data. However, with the help of integration, the ARIMA model can use time series data that is non-stationary. The integration part of the ARIMA model allows the model to differentiate between each data point and remove non-constant trends. Therefore, the ARIMA model is expected to perform better for bearing RUL predictions when the whole bearing life is estimated to be a linear descending behavior.

An ARIMA model is typically denoted as ARIMA(p,d,q) where p, d and q are positive integers. p represents the order of the autoregressive part, d represents the degree of differentiation, and q represents the order of the moving-average. The basic mathematical equation is shown in equation (23) [80].

$$Y_t = \emptyset_0 + \emptyset_1 Y_{t-1} + \emptyset_2 Y_{t-2} + \cdots + \emptyset_p Y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \qquad (23)$$

where $\emptyset$ and θ are coefficients for the autoregressive part and the moving-average part of the model, respectively, and ε is the error term.

Choosing the order of p, q, and d is crucial to optimize the ARIMA model's accuracy.

(1) For AR, p refers to the number of lags of the time series that are used as predictors for the current value of the time series. For MA, q refers to the number of lagged forecast errors used to model the current value of the time series.

(2) I, as integration, refers to how many differentiations are applied to make the non-stationary time series data, stationary.

The typical way of finding optimum p and q orders is by applying the partial autocorrelation function (PACF) and autocorrelation function (ACF). PACF is used to determine the order p of the AR, revealing the direct effects of time series data and its lagged version. ACF is used to determine the order q of the MA, and determines correlations between time series data and a lagged version of itself at different lags. For example, if $p = 1$, it means the model uses the previous observation to predict the current one (i.e. m_{t-1} and m_t). If $p = 2$, the model will use two previous observations to predict the current one (i.e. m_{t-2} and m_t). Similarly, if $q = 1$, the model will use the error from the previous prediction to adjust the current prediction (i.e. ε_{t-1} and ε_t). If $q = 2$, the model will use the errors from two previous predictions to adjust the current prediction (i.e. ε_{t-2} and ε_t).

In this thesis, a Python package called Skforcast generates optimum p, q, and d orders to save calculation time. The AutoARIMA function in this package autoselects the most suitable ARIMA model for each CNN network. As shown in Figure 28 and

Figure 29, the best ARIMA model is chosen to be ARIMA(5,1,4) for scalogram-based CNNs. This means that the CNN outputs are differentiated once to become stationary. The prediction is made based on the fifth previous value of the time series, and the error is calculated based on the fourth previous error of the time series.

```
for i in range(len(past_series)):
  y_train = Series(past_series[i].squeeze(), index = list(range(0, len(past_series[i].squeeze()))))
  fh = np.arange(1, int(10*len(y_est[i])))

  forecaster = AutoARIMA(suppress_warnings=True)
  forecaster.fit(y_train)
  y_pred = forecaster.predict(fh)
  y_pred_int = forecaster.predict_interval(fh, coverage=0.95)

  fh_train = ForecastingHorizon(y_train.index, is_relative=False)  # in-sample forecasting horizon
  y_fit = forecaster.predict(fh_train)
  y_fit_int = forecaster.predict_interval(fh_train, coverage=0.95)

  arima_y_trains.append(y_train)
  arima_y_preds.append(y_pred)
  arima_y_fits.append(y_fit)
  arima_fh_trains.append(fh_train)
  arima_fhs.append(fh)

  arima_y_pred_ints.append(y_pred_int)
  arima_y_fits_ints.append(y_fit_int)

  arima_forecasters.append(forecaster)
  arima_params.append(forecaster.get_fitted_params())
```

Figure 28 Example of the AutoARIMA model generation procedure code

```
Parameters of 0th ARIMA model:
{'intercept': -0.00012028800141460766, 'ar.L1': -0.9294459238279495,
'ar.L2': 0.35437402800011136, 'ar.L3': 0.8334069277290589, 'ar.L4':
0.40395098028297327, 'ar.L5': 0.017156325575737392, 'ma.L1':
1.064085817530383, 'ma.L2': -0.11346845094394897, 'ma.L3': -
0.6588857805660129, 'ma.L4': -0.33940932854836486, 'sigma2':
7.185145679518337e-07, 'order': (5, 1, 4), 'seasonal_order': (0, 0, 0, 0),
'aic': -20342.218624039815, 'aicc': -20342.071055565808, 'bic': -
20281.76155324288, 'hqic': -20319.902208354855}
Parameters of 1th ARIMA model:
{'intercept': -6.892965082866378e-05, 'ar.L1': 0.2409585636195264,
'ar.L2': 0.5341827970010907, 'ar.L3': 0.07089482868845846, 'ma.L1': -
0.1102085134627819, 'ma.L2': -0.3820869862783088, 'sigma2':
1.8404616706463872e-06, 'order': (3, 1, 2), 'seasonal_order': (0, 0, 0,
0), 'aic': -12284.64560389203, 'aicc': -12284.550608132911, 'bic': -
12249.091313129762, 'hqic': -12271.245382056073}
```

Figure 29 AutoARIMA function results showing the best two options for the ARIMA model

3.3.6 Channel fusion

The channel fusion method is explicitly used to help combine CNNs built based on channel 1 and channel 2 data. By establishing the combined CNN, it is possible to illustrate how different vibration sources can affect prognosis results.

Channel 1-based CNNs and channel 2-based CNNs are created separately, and their outputs are fed into a final fusion step. Every CNN's last steps include one flatten layer and three dense layers, as shown in the example provided in Figure 25. Specifically, the channel 1-based CNN's outputs form a branch after the flatten layer for channel fused CNNs, as shown in Figure 30. Then, a second branch is also created via a channel 2-based CNN following the same procedure. Lastly, the two branches are combined and fed into the last few dense layers to establish a channel fused CNN, as shown in Figure 31.

```
# the first branch hyperparameters
lr =  0.001
cnn_filters_1 =  32
cnn_filters_2 =  32
cnn_filters_3 =  128
fc_units_1 =  128
fc_units_2 =  64
kernel_size =  5
bn =  False
activation = 'relu'

# the first branch operates on the first input
x = layers.Conv2D(filters=cnn_filters_1, kernel_size=kernel_size, strides=[1,1], activation=activation, input_shape=input_shape_C1)(input_C1)
x = layers.Conv2D(filters=cnn_filters_1, kernel_size=kernel_size, strides=[1,1], activation=activation)(x)

x = layers.MaxPooling2D(pool_size=(2, 2),strides=(1,1))(x)
if bn:
  x = layers.BatchNormalization(axis=-1)(x)

x = layers.Conv2D(filters=cnn_filters_2, kernel_size=kernel_size, strides=[1,1], activation=activation)(x)
x = layers.Conv2D(filters=cnn_filters_2, kernel_size=kernel_size, strides=[1,1], activation=activation)(x)

x = layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2))(x)
if bn:
  x = layers.BatchNormalization(axis=-1)(x)

x = layers.Conv2D(filters=cnn_filters_3, kernel_size=kernel_size, strides=[1,1], activation=activation)(x)
x = layers.Conv2D(filters=cnn_filters_3, kernel_size=kernel_size, strides=[1,1], activation=activation)(x)

x = layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2))(x)
if bn:
  x = layers.BatchNormalization(axis=-1)(x)

x = layers.Flatten()(x)

branch_C1 = Model(inputs=input_C1, outputs=x)
```

Figure 30 Example of branch 1 for a channel 1 based CNN

```
combined = layers.concatenate([branch_C1.output, branch_C2.output])
# apply FC layers and then a regression prediction on the combined outputs
x = layers.Dense(fc_units_1, activation=activation)(combined)
x = layers.Dense(fc_units_2, activation=activation)(x)
x = layers.Dense(1)(x)
# then model will accept the inputs of the two branches and then output a single value
model = Model(inputs=[branch_C1.input, branch_C2.input], outputs=x)
```

Figure 31 Channel fusion function for channel 1 and channel 2 data

Chapter 4 Results

4.1 Diagnosis results

Diagnosis methods are applied to each bearing twice using channel 1 and channel 2 accelerometer data. Since this results in a very large number of plots, only interesting results are presented in the main body of this thesis. The remainder of the plots are provided in Appendix B. The final diagnosis results for all bearings are provided in Table 5. In this table, "IN" represents an inner race defect, "OUT" represents an outer race defect, "N/A" means no defect was detected using the FFT method, "f_r" represents a bearing that expressed frequency harmonics of the rotating frequency, and "Drift" represents a non-stationary rotating speed.

Table 5 Final diagnosis results for all bearings

	Channel 1	Channel 2
Bearing1_1	IN	IN
Bearing1_2	IN	IN
Bearing1_3	IN	N/A
Bearing1_5	IN	OUT
Bearing1_6	IN	IN
Bearing1_7	IN	IN
Bearing2_1	IN	N/A
Bearing2_2	OUT	OUT
Bearing2_3	N/A	f_r+Drift
Bearing2_4	OUT	OUT
Bearing2_5	N/A	f_r
Bearing2_6	OUT	OUT
Bearing2_7	N/A	N/A
Bearing3_1	IN	IN
Bearing3_2	IN	IN
Bearing3_3	OUT	OUT

4.1.1 Inner race defects

In this section, bearing 1_1 will be used to demonstrate bearing inner race defects. As shown in Figure 32 for bearing 1_1, all FCFs are used separately on the frequency plots to identify which defect a particular bearing is experiencing. As Figure 32 indicates, fault peaks line up perfectly with the inner race fault FCF in (b), and even though the first peak from (d) also lines up with the ball fault FCF, the other two peaks do not follow the same alignment. Therefore, an inner race defect appears to be shown for bearing 1_1.

Then, by comparing plots in Figure 33 and Figure 34, significant peaks can be observed at the BPFI frequency indication lines, and their sidebands are indicated as doted red lines whose values are calculated as $BPFI \pm MRF$. Sidebands are additional frequency components that appear in the frequency spectrum around the primary frequency of the bearing. These sidebands are typically caused by the presence of faults or defects in the bearing, such as localized wear, cracks, or spalling. As the bearing rotates, these defects can generate vibration signals that have additional frequency components that are related to the rotational frequency of the bearing. By analyzing the sidebands, it is possible to determine the type and severity of the fault or defect in the bearing. For example, the location and spacing of the sidebands can indicate the type of defect, while the amplitude of the sidebands can provide information about the severity of the defect.

For bearing 1_1, channel 2 data has a more evident plot for the final result than channel 1 data. Moreover, channel 2 data has higher peak values than channel 1. One more thing to notice is that the peaks do not align with the BPFI values exactly. This may happen when the rotating speed of a machine is not held completely constant during the machine's operation, or if the data resolution isn't high enough.

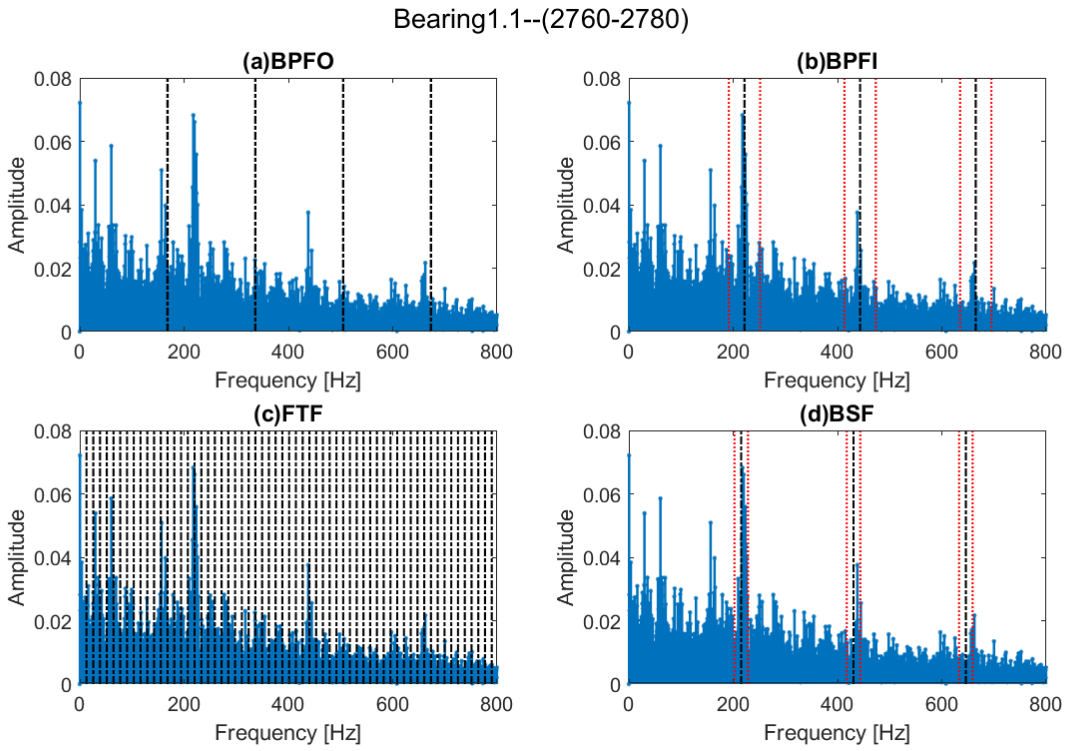


Figure 32 Bearing 1_1 diagnosis results for all four FCFs

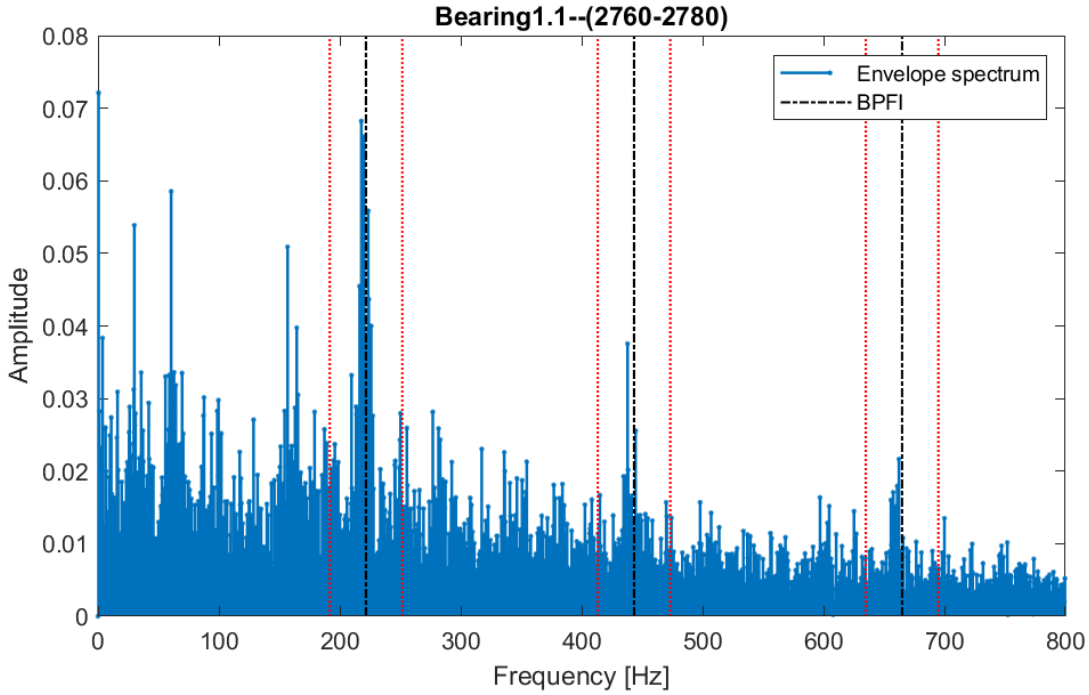


Figure 33 Bearing 1_1 detailed BPFI diagnosis result for channel 1

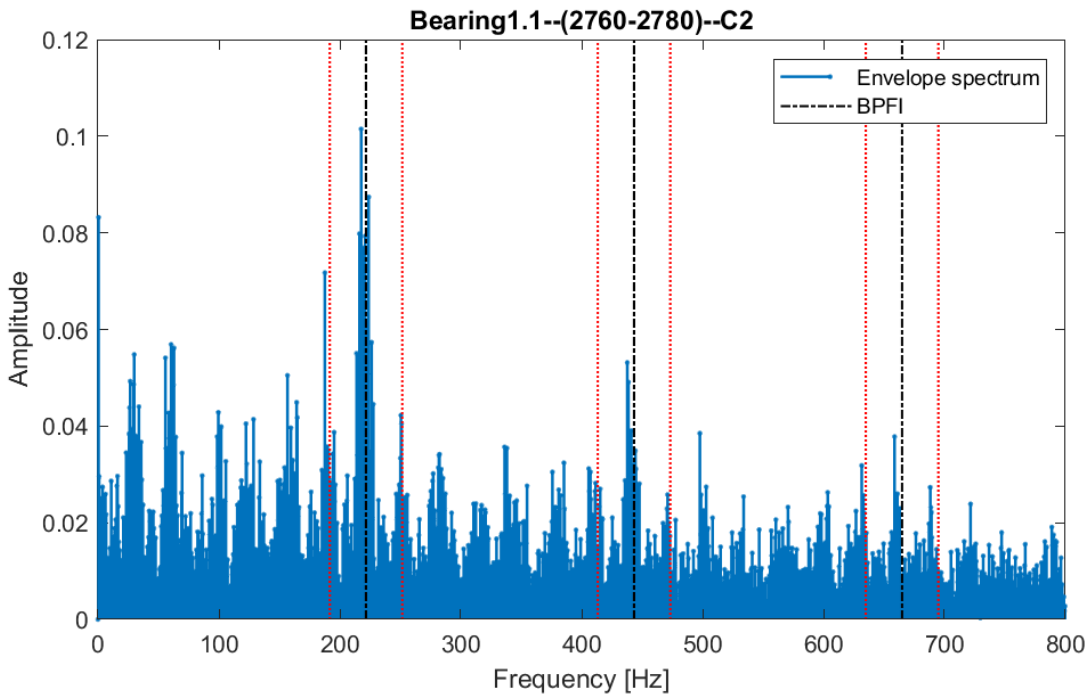


Figure 34 Bearing 1_1 detailed BPFI diagnosis result for channel 2

4.1.2 Outer race defects

As shown in Figure 35 and Figure 36, bearing 2_6 is identified as having an outer race defect. The faulty peaks align with the BPFO frequency lines over multiple harmonics, and no other features appear to be defective. The BPFO has no sidebands for bearings whose outer race is fixed and where the inner race rotates. However, for bearings whose inner race is fixed during operation, their BPFO would have sidebands just like BPFI.

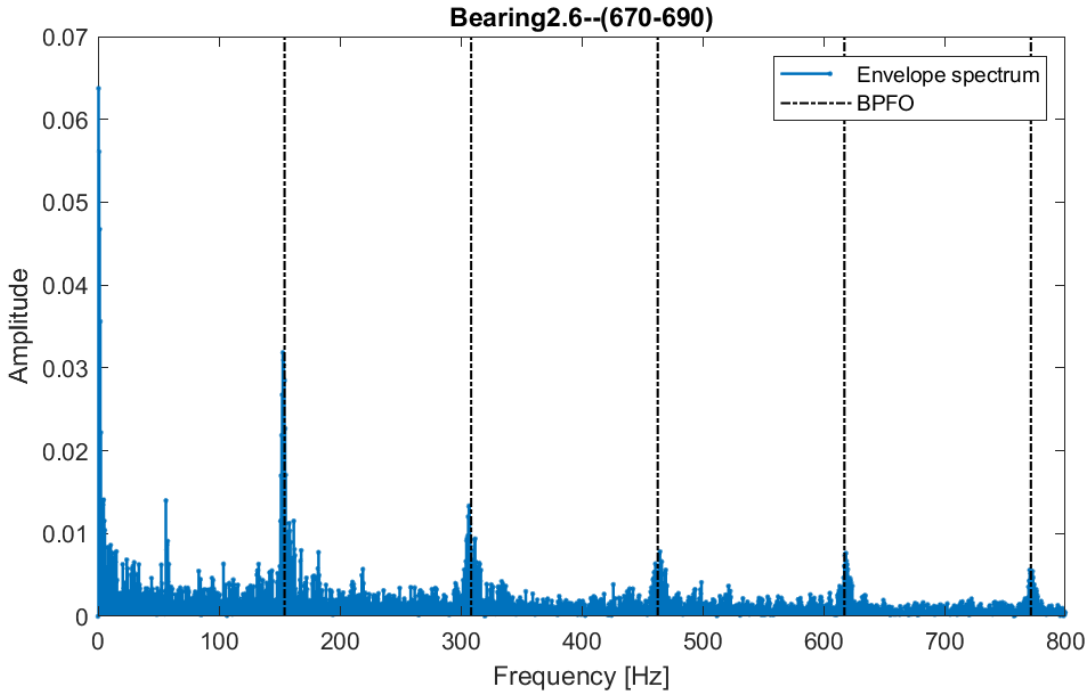


Figure 35 Bearing 2_6 detailed BPFO diagnosis result for channel 1

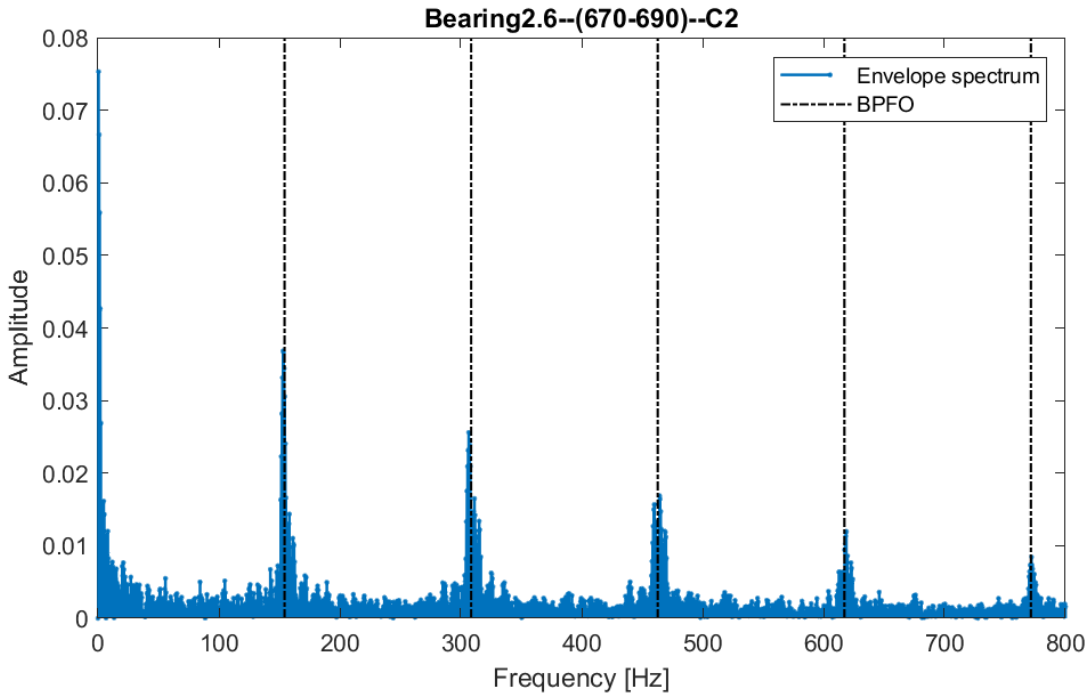


Figure 36 Bearing 2_6 detailed BPFO diagnosis result for channel 2

Although most bearings with inner race or outer race defects represent a similar behavior on channel 1 and channel 2, bearing 1_5 shows an inner race defect on channel 1, and an outer race defect on channel 2, as shown in Figure 37. This can happen when a damaged bearing is allowed to run for longer periods of time as the damaged component can lead to damage of other components.

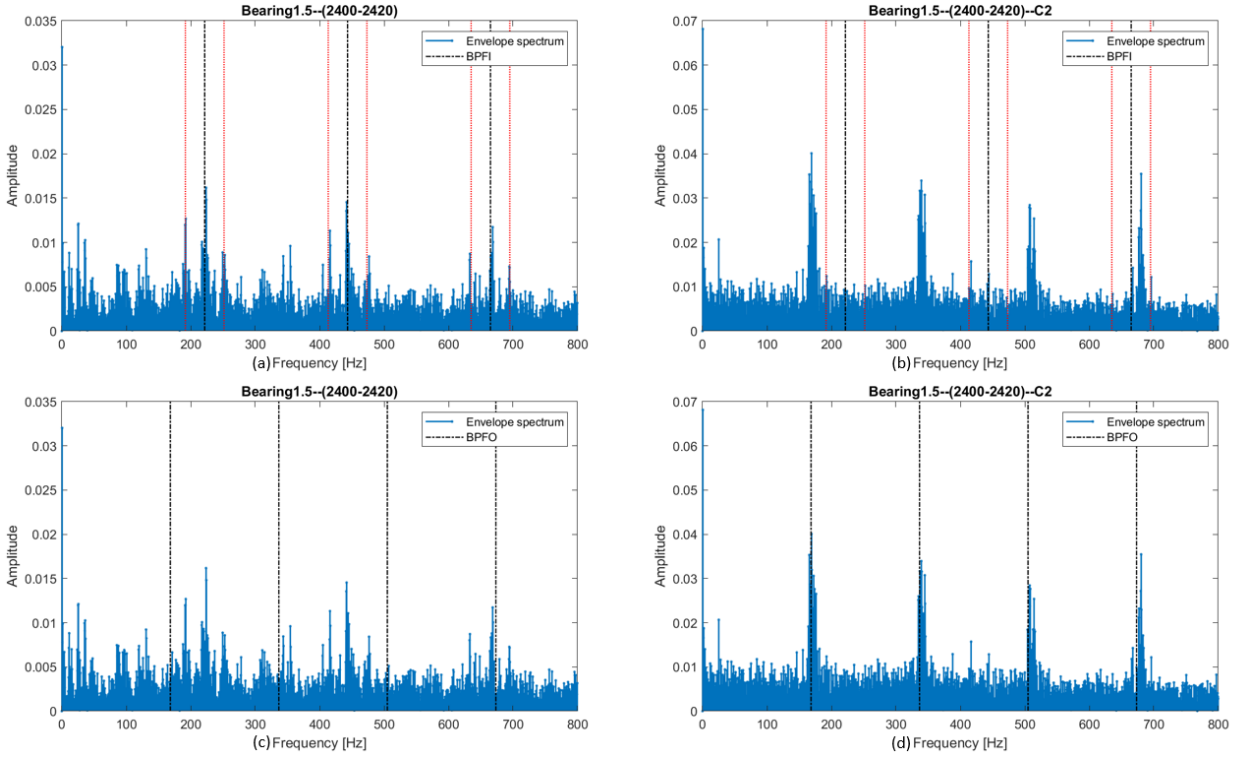


Figure 37 Bearing 1_5 diagnosis results: (a)C1 BPFI (b)C2 BPFI (c)C1 BPFO (d) C2 BPFO

On the other hand, bearing 1_3 and bearing 2_1 are identified as having inner race defects for channel 1 data, but their channel 2 data indicates that no defects are observed, as shown in Figure 38. Nonetheless, it is not safe to discard all channel 2 data, as it may also detect features that channel 1 does not. An example can be found in section 4.1.3.

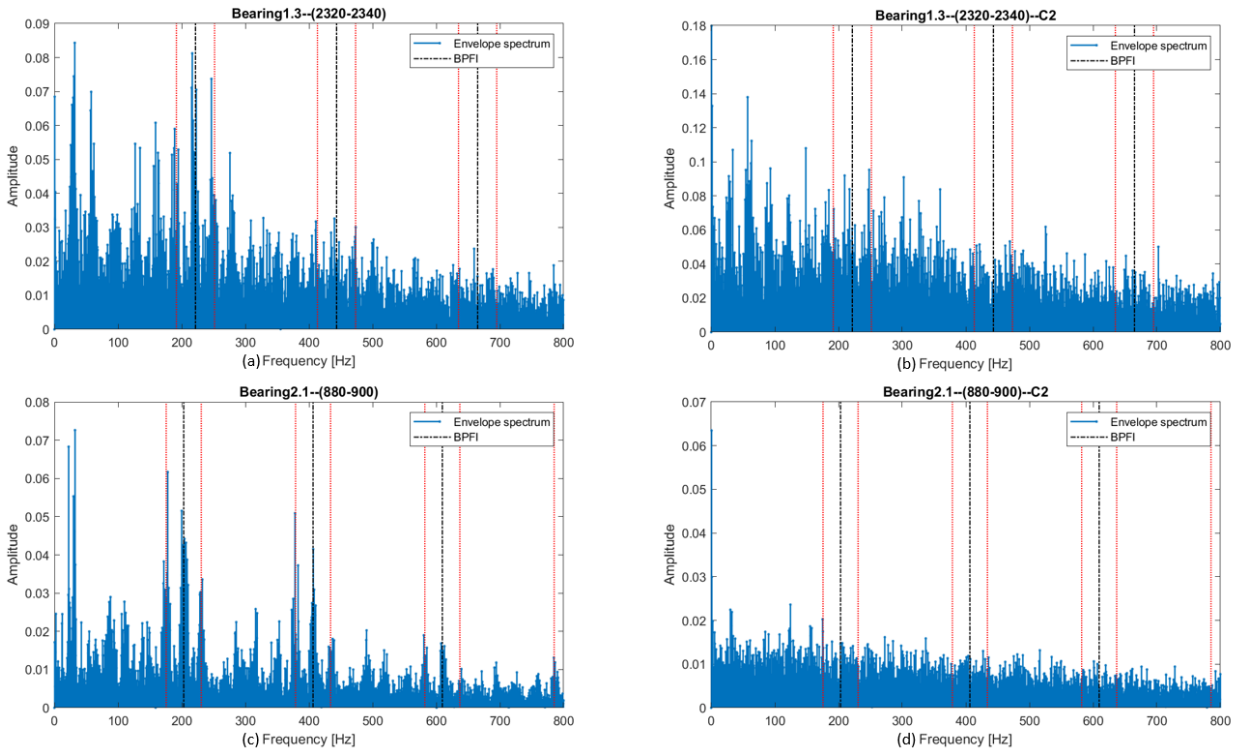


Figure 38 Channel 1 and channel 2 data comparison for: (a) bearing 1_3 channel 1 BPFI data (b) bearing 1_3 channel 2 BPFI data (c) bearing 2_1 channel 1 BPFI data (d) bearing 2_1 channel 2 BPFI data

4.1.3 f_r related behaviors

f_r peaks present a distinct defect behavior. As shown in Figure 39 and Figure 40, bearing 2_5 is identified as having fault frequency peaks at the f_r and its harmonics, especially for channel 2 data. According to [81], a rotating looseness defect can be suspected in this case, as the ground noise level is raised, and almost every single peak lands on the f_r and its harmonics. Since more than 10 peaks align with the f_r, a severe defect is assumed. Bearing 2_3 also indicates similar results in Figure 41. However, only the first few peaks match the f_r and its harmonics, and the remaining

peaks are shifted. It is reasonable to assume that this is still a looseness problem, but combined with an inconstant rotating speed.

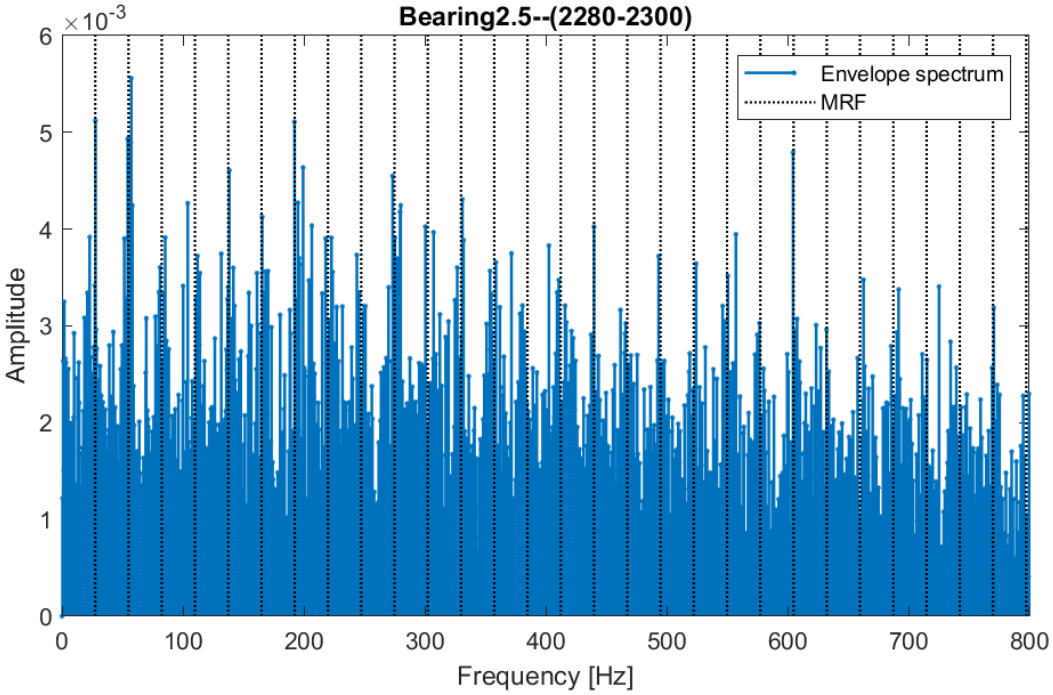


Figure 39 Bearing 2_5 detailed f_r diagnosis result for channel 1

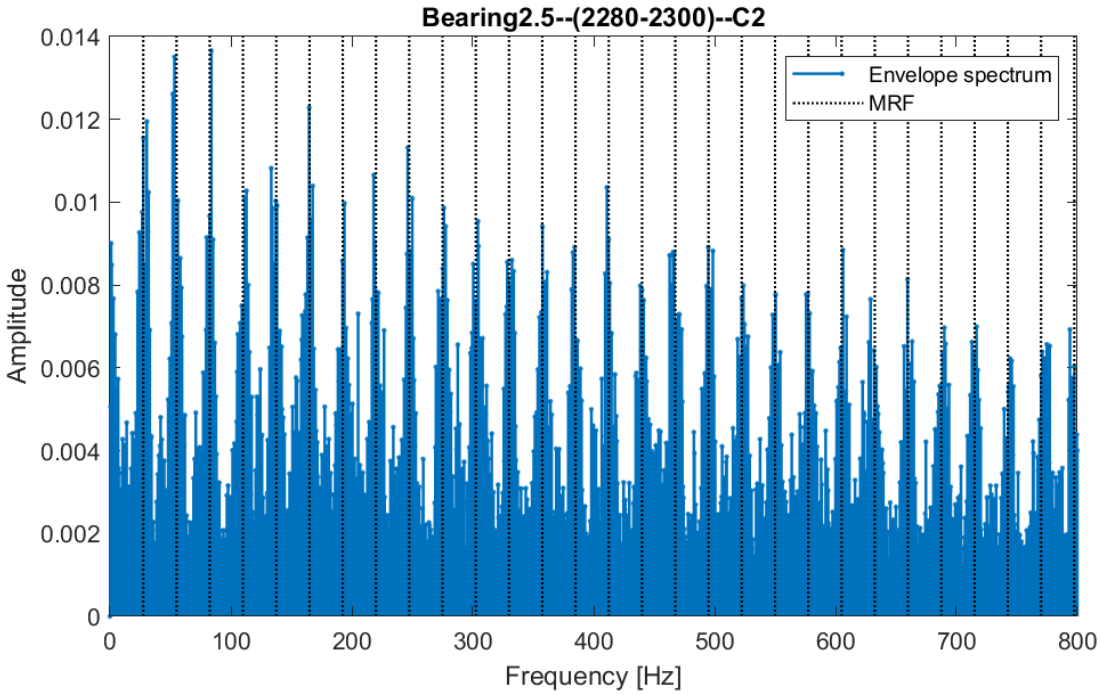


Figure 40 Bearing 2_5 detailed f_r diagnosis result for channel 2

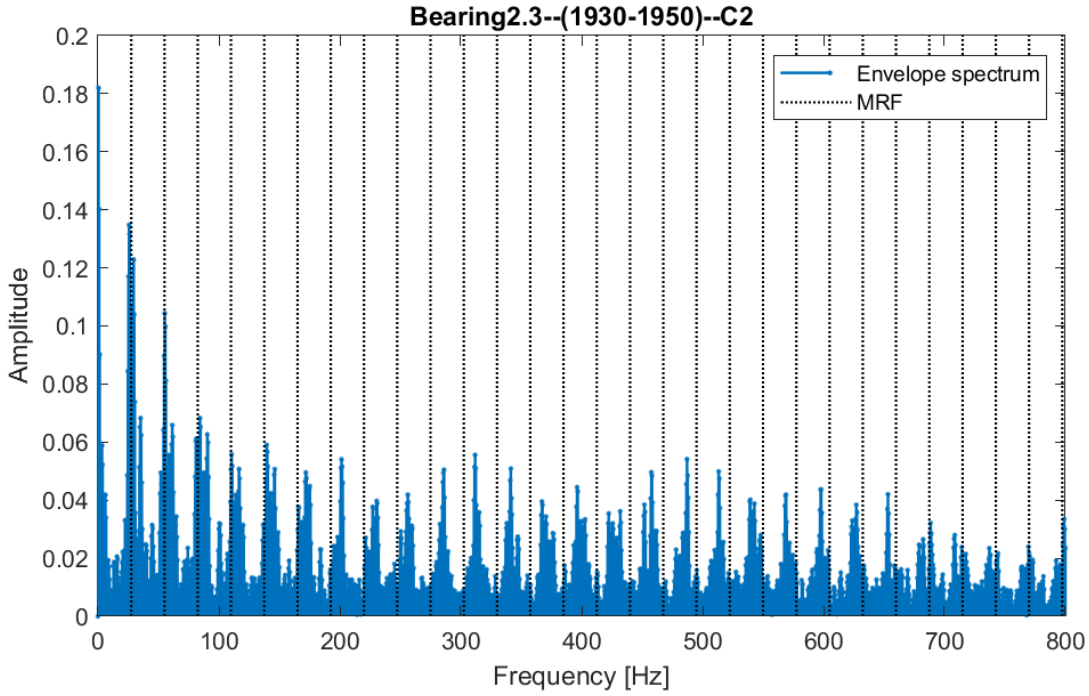


Figure 41 Bearing 2_3 detailed f_r diagnosis result for channel 2

4.1.4 Unidentified defects

A few bearings had assumed defects that were hard to identify. For example, bearing 2_7 appears to have no significant fault frequency, as shown in Figure 42 and Figure 43. According to its total operation time, bearing 2_7 is categorized as a sudden failure early in its life. As such, it is hard to provide a diagnosis for a bearing that failed so quickly.

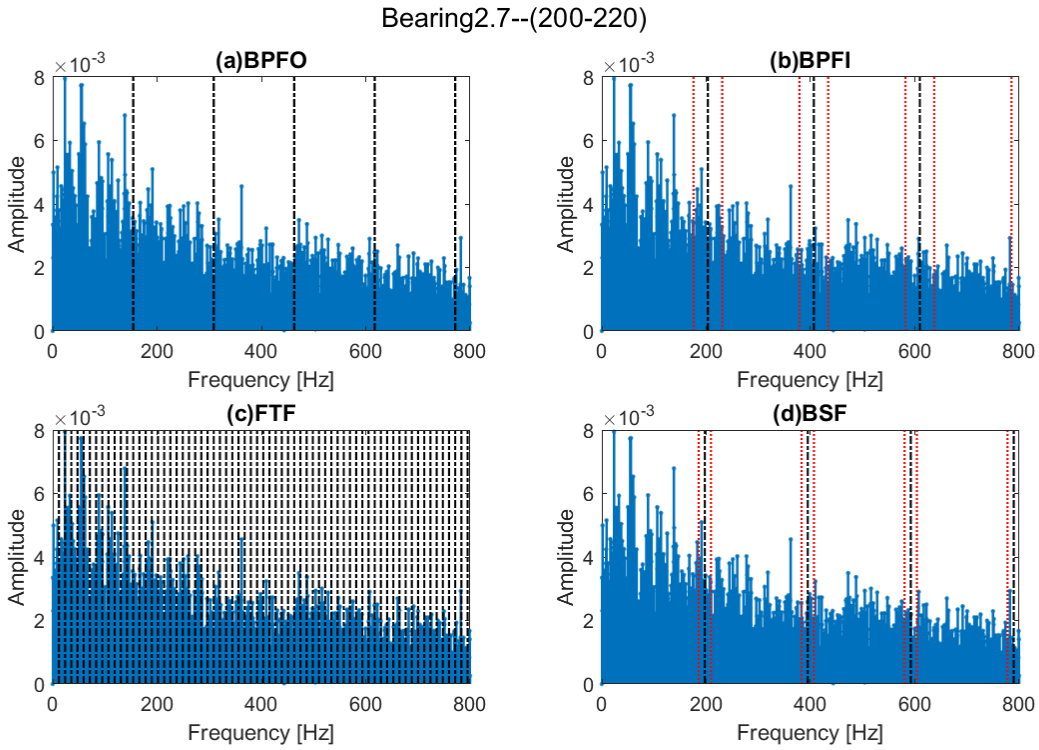


Figure 42 Bearing 2_7 detailed diagnosis results for channel 1

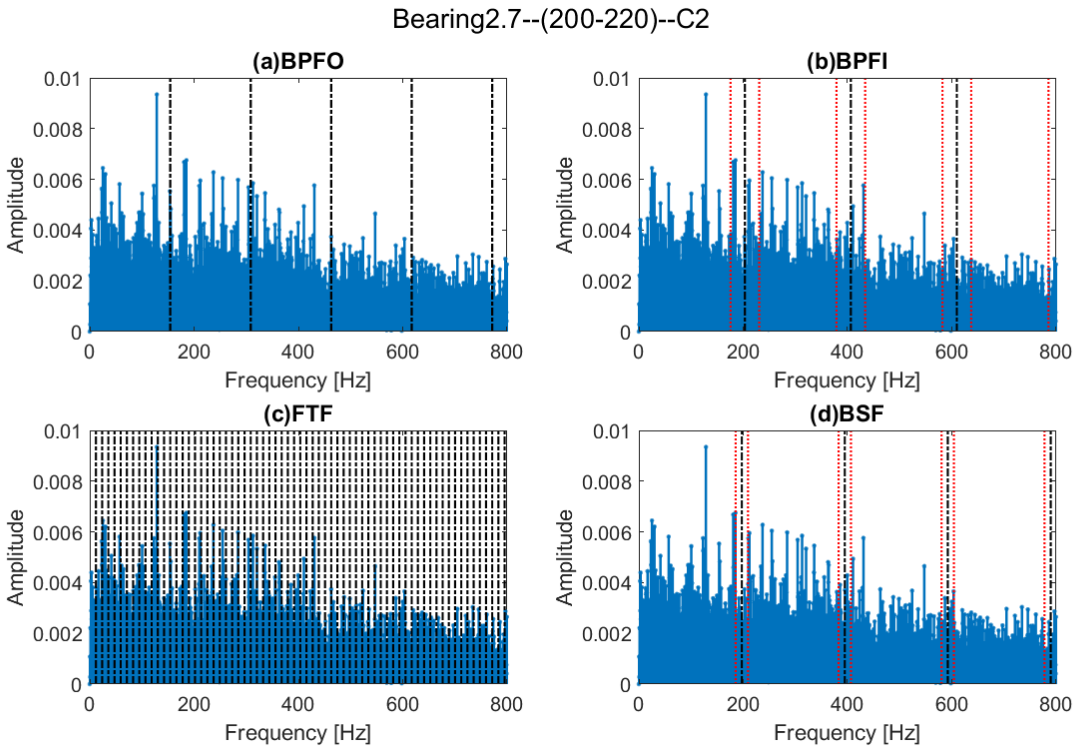


Figure 43 Bearing 2_7 detailed diagnosis results for channel 2

4.2 Prognosis results

This section provides RUL prognosis results for the PRONOSTIA bearing dataset. Results are divided into four major sections, including a spectrogram-based CNN for channel 1 data, a spectrogram-based CNN for channel 2 data, a channel-fusion-based CNN for data from both channels, and a scalogram-based CNN for channel 1 data. The python code used for the channel 1 spectrogram-based CNN results is given Appendix D. Other CNN codes used in this thesis follow the same logic and only required specific modifications to account for the input variations. As shown in Figure 44 and Figure 45, example codes for preprocessing via spectrograms and scalograms

are provided. Changing the code's parameters can achieve output plots with different scales and resolutions. The full list of prognosis plot results is given in Appendix B .

```
def spectrogram_channel_1(sample):
    _, _, ch1_spec = signal.spectrogram(sample[:,0],
                                        fs=1.0,
                                        window=('tukey', 0.25),
                                        nperseg=22,
                                        noverlap=0,
                                        nfft=230,
                                        detrend='constant',
                                        return_onesided=True,
                                        scaling='density',
                                        axis=-1)

    ch1_spec = (ch1_spec - np.min(ch1_spec)) / np.ptp(ch1_spec)
    return ch1_spec
```

Figure 44 Spectrogram code used for channel 1 data

```
def scalogram_channel_1(data, fs=25600):
    a = data[:, 0]
    wavename = "morl"  # morlet

    totalscale = 129
    fc = pywt.central_frequency(wavename, precision=8)  # center frequency
    cparam = 2 * fc * totalscale
    scales = cparam / np.arange(totalscale, 1, -1)
    [cwtmatr, frequencies] = pywt.cwt(a, scales, wavename, 1.0/fs)

    sampled_idxs = np.array(list(range(0, 128))) * 20
    cwtmatr = cwtmatr[:, sampled_idxs]

    return cwtmatr
```

Figure 45 Scalogram code used for channel 1 data

4.2.1 Spectrogram-based CNN for channel 1 data results

The prognosis procedure is developed for the spectrogram-based CNN for channel 1 data as an example of all prognosis results. The whole prognosis procedure follows

the steps outlined in 3.3. Specifically, the raw vibration signal is preprocessed and the HI is established based on processed plots. The CNN is used for training, and then prediction. The ARIMA model is used for forecasting RUL based on CNN results. The data preprocessing and use of the CNN are stage one of the procedure, while the ARIMA model and final evaluation are stage two.

(a) Stage 1:

Raw vibration data for channel 1 is processed into spectrograms, shown in Appendix A , and a CNN is defined, as shown in Figure 25. The final training accuracy for the CNNs is shown in Figure 46. The training process for the CNNs is found to be successful, since in all cases, results align well with the HIs assumed earlier. In addition, the test loss function used to evaluate the testing process is displayed in Figure 47. It is seen that the total loss for the testing process is decreasing, meaning that as more trials are performed, the more accurate the CNN can predict the final results. The final loss is near zero, where the small spikes at the end indicate that since the CNN is well trained, whenever an "outlier" is detected in that epoch, the loss function will assign a big number to it. But since only a few spikes are detected, and the loss function converges to zero, it is safe to say that the CNN is well trained and tested to perform further prediction tasks.

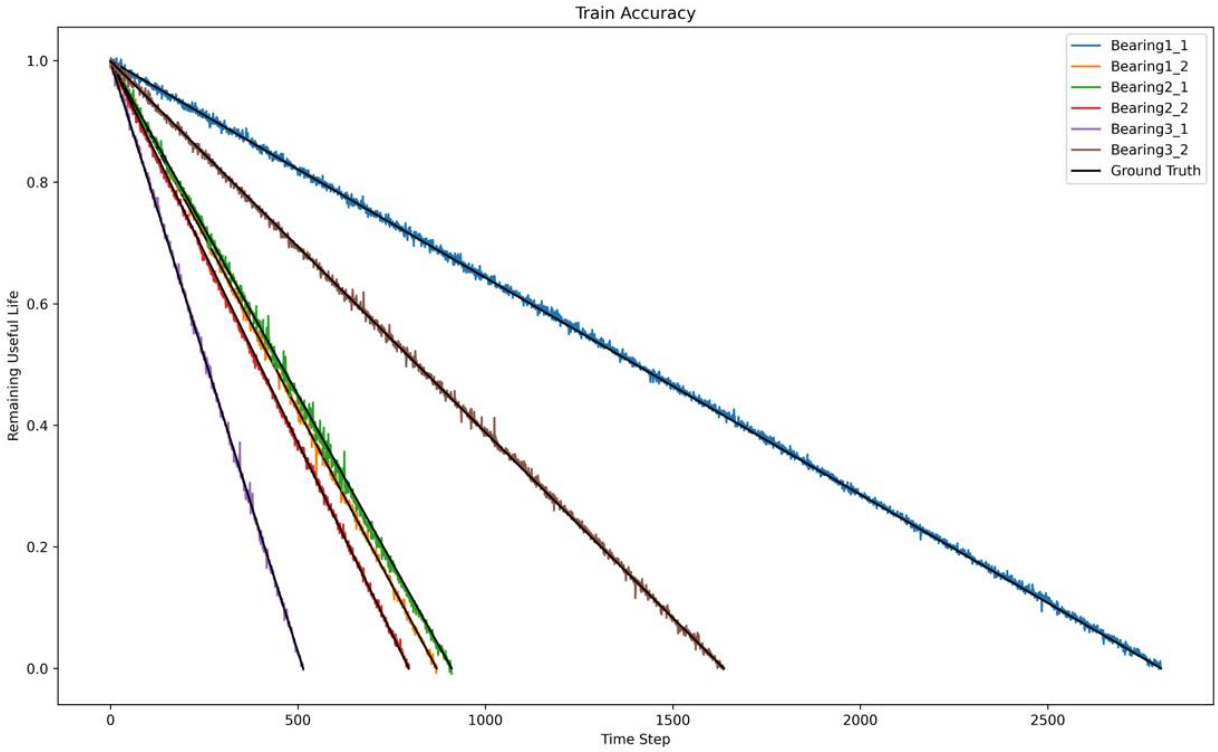


Figure 46 Training accuracy for the spectrogram-based CNN for channel 1 data

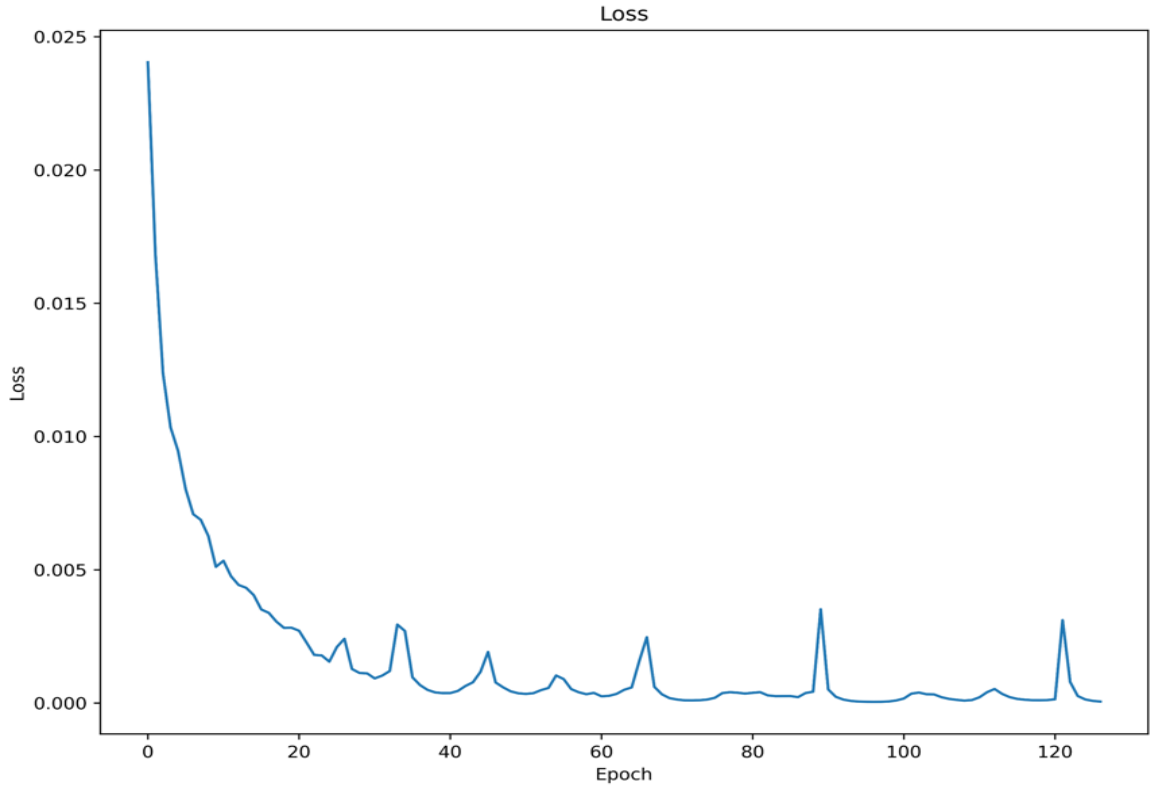


Figure 47 Test loss for the spectrogram-based CNN for channel 1 data

In this thesis, the data from the test set is used (i.e. from healthy to the instant where the PRONOSTIA testers pause the recording) to perform final predictions. Conversely, some studies have used the same technique, but applied the full test data (i.e. from healthy to the ultimate failure point) and claim high prediction accuracies [82], [83]. However, it is believed that this technique is only viable to show the overall trend of a bearing's behavior, and as a comparison to the actual test set, which is the partial data for bearings used to perform RUL predictions. It cannot be used as the final result for the RUL predictions. Nonetheless, since it is still useful for comparison, a shallow attempt is conducted.

Taking test bearing 1_3 as an example, the final prediction results from the CNN align with the course of ground truth (HI), as shown in Figure 48. However, the first 50% of the prediction contains too much noise and brings confusion to the final results. Therefore, a 3rd order polynomial regression was used to help eliminate the noise, as shown in Figure 49. The noise of the final prediction is suppressed compared to Figure 48, but the polynomial regression also eliminates details of the behavior. For example, in the first 50% of the data, bearing 1_3 shows a hint of failure due to a low HI number, but the polynomial regression result indicates the opposite.

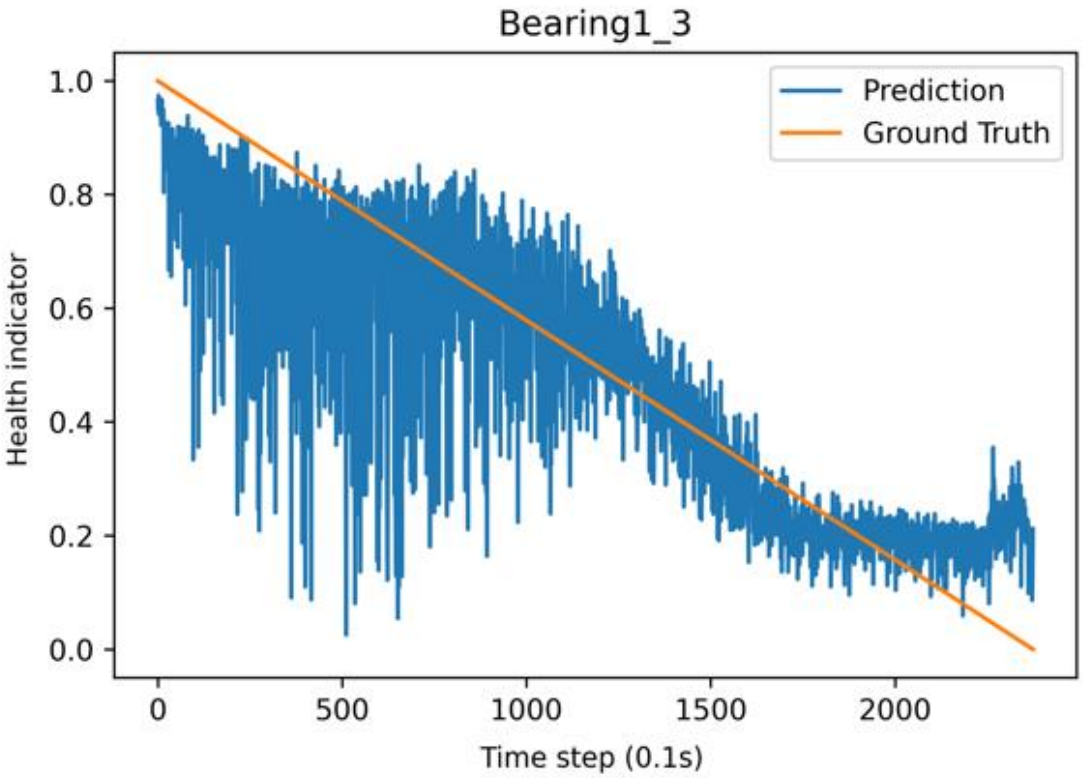


Figure 48 Raw CNN prediction for bearing 1_3

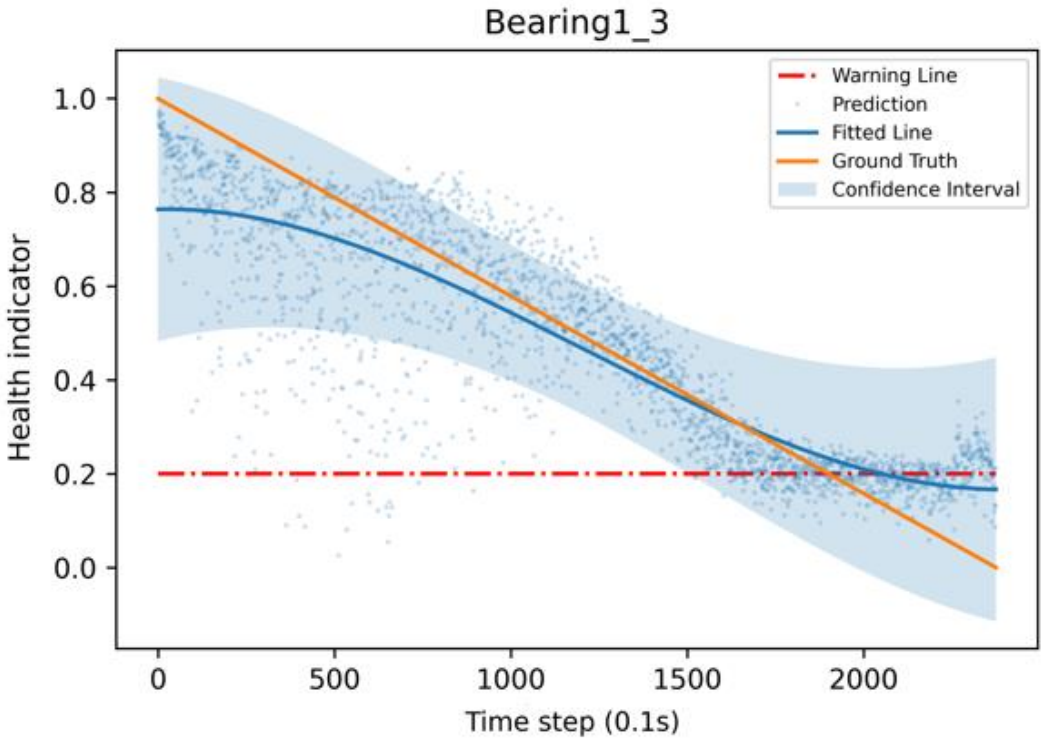


Figure 49 3rd order polynomial regression result for bearing 1_3's prediction

Therefore, a moving average method is used to perform denoising and smoothing, and the proper amount of data is applied to make a final RUL prediction. As shown in Figure 50, bearing 1_3's test data is shown based on the HI values to represent its life. One thing to notice is that the blue dots shown in Figure 49 are not shown in Figure 50, providing a clearer image overall. Moreover the dots themselves would not be the same. This is because each time a CNN is trained and tested differently, the test set inputs are different for Figure 49 and Figure 50, where one uses full life readings and one uses provided readings respectively. Nevertheless, it is still clear that by using the moving average method, more detail and a reduction of noise is given to help understand a bearing's behavior.

As can be seen in Figure 50, before 1200-time steps, bearing 1_3 has an unstable working experience. The HIs for this range vary dramatically, but still show a decreasing trend and never go below zero, which means the bearing is still functioning but not degrading. After this stage, the HIs start to decrease and fewer outliers are expected, meaning that the bearing goes into a stage where the spectrograms start to display features indicating this degradation process. On the other hand, spectrograms for 100% and 70% life stages for bearing 1_3 indicate noise and unstable patterns, as shown in Figure 51. But this noisy situation starts to settle at the 40% and 25% life stages, and a very light converging trend can be seen. By observing the plot, this trend might be omitted, but using a CNN shows that it can

pick up these ambiguous features and come to the conclusion that they represent a bearing's life.

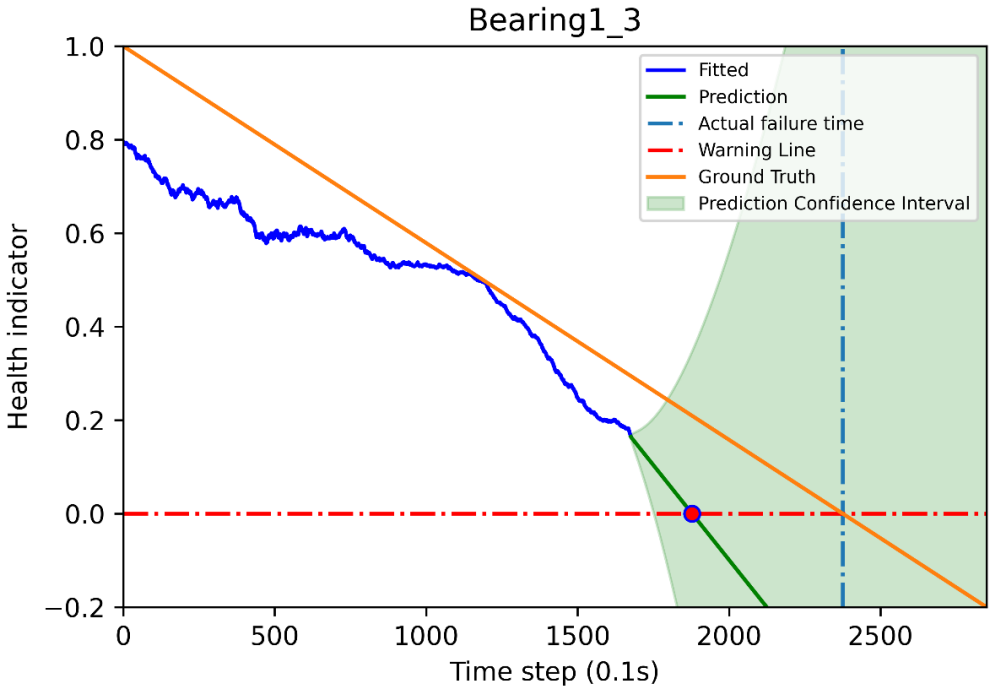


Figure 50 Moving average result for bearing 1_3 with its actual test data prediction

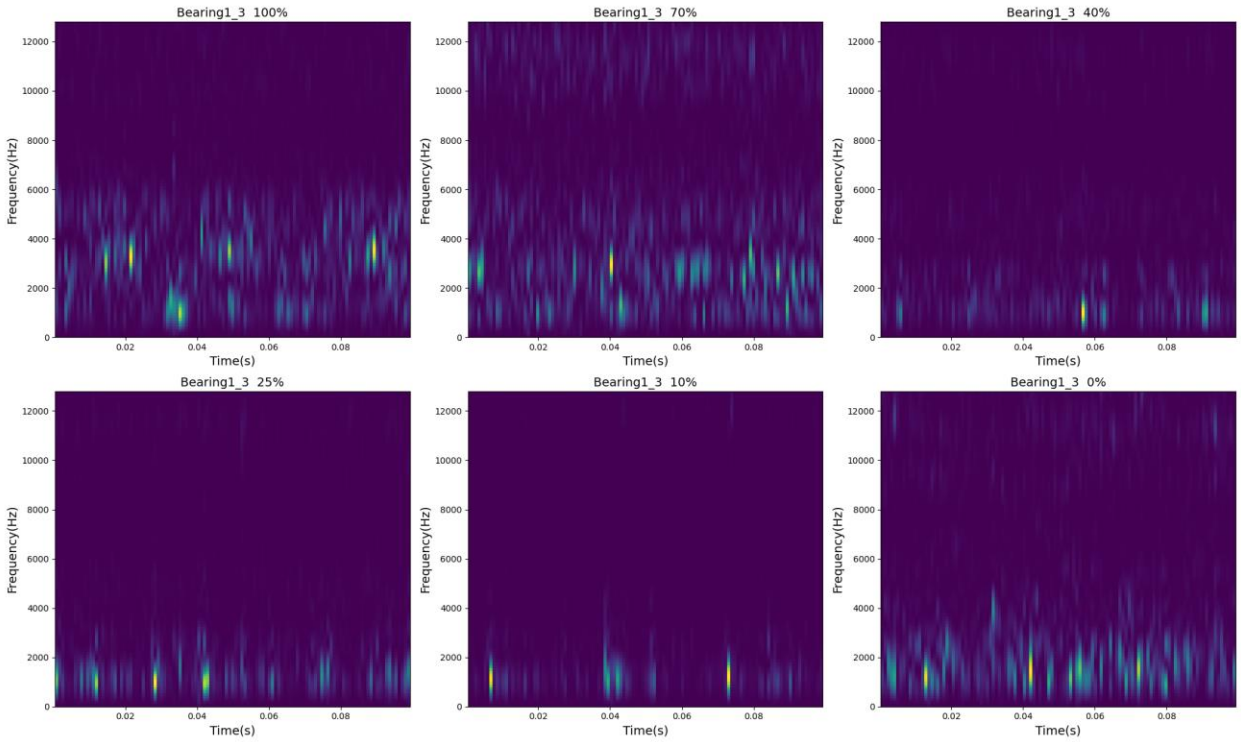


Figure 51 Spectrograms for bearing 1_3 channel 1 data at each life stage

(b) Stage 2:

As the provided test data is labelled with HIs by the CNN, it is possible to use the forecasting method to predict how the rest of a given bearing would perform and how long it could keep working. As shown in Figure 50, the green line represents the ARIMA model results and indicates how the bearing RUL is expected to perform. The blue dashed line indicates the real bearing failure time, and the red dashed line indicates the assumed failure time, which is represented by a zero HI value. When the solid green line intersects with the red dashed line, the RUL is predicted as the time steps from the prediction time to the intersection point. As shown in Figure 50, the predicted bearing failure time is earlier than the actual bearing failure time. The predicted RUL for the bearing 1_3 spectrogram-based CNN for channel 1 data is

calculated to be 498-time steps earlier than the actual failure time, where the actual RUL is 574-time steps. Therefore, based on equations (2)-(4), the individual error percentage for bearing 1_3 is calculated to be 86.7%, and the individual accuracy score is calculated to be 0.049. This result is very promising, as it predicts the bearing's RUL with only a small percentage error, and the final accuracy is satisfactory.

More bearing 1_3 data was then fed into the CNN to explore the relationship between the amount of test data and the final accuracy. As shown in Figure 52, in order to simulate the scenario where more data is fed into the CNN as time goes by, 95% of bearing 1_3's data is fed into the CNN, and the final RUL prediction is better than the one shown in Figure 50, where only ~60% of data was used. The percentage error is calculated to be 91.6%, and the final score is calculated to be 0.042. The percentage and prediction score here seem to have decreased compared to using 60% of the available data, but the confidence interval converges more. Moreover, the increased percentage error is only due to a smaller denominator in equation (2), which is the actual RUL prediction as more data is fed into the CNN. Therefore, it can be assumed that the algorithm can predict the RUL better with more data. Other bearings indicate similar trends, and their prediction plots can be found in Appendix B .

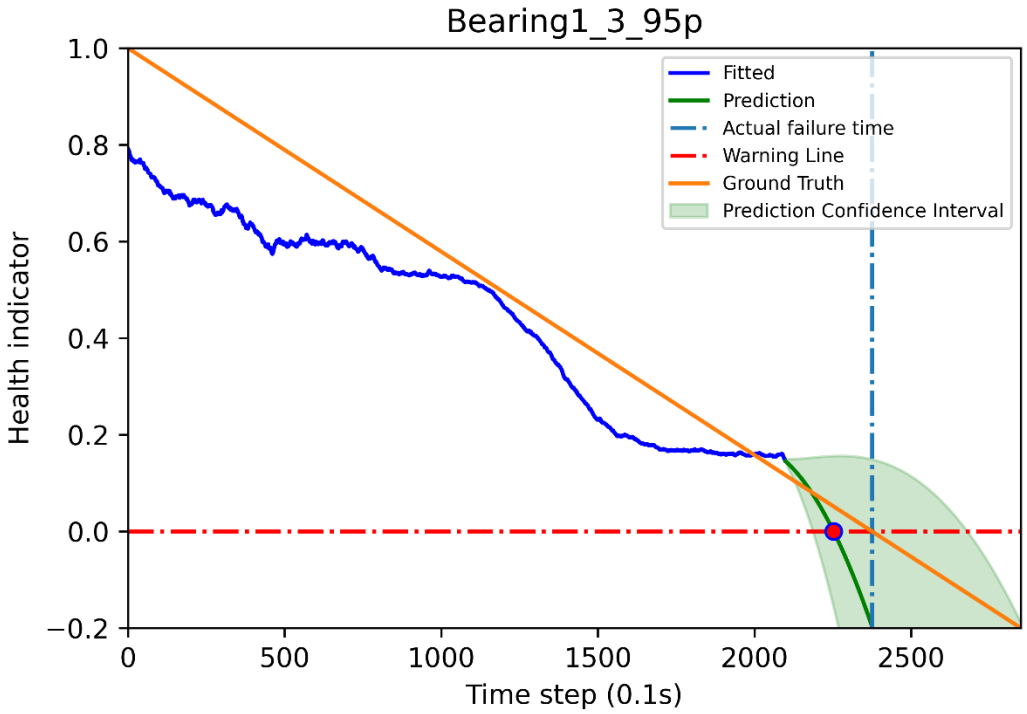


Figure 52 Results of the RUL prediction for bearing 1_3 after 95% of bearing data is fed into the CNN

Bearing 2_7 was found to not give a good estimation, as shown in Figure 53 and Figure 54. The CNN predicted that bearing 2_7 had a high HI at the beginning, and that the HI readings decrease slowly over time. However, the failure happened in less than 20-time steps and with that quick of a failure, the methods introduced don't hold up well. Even with 95% of its total life fed into the CNN, it was still incapable of making predictions for its RUL. The main reason is assumed to be due to the very small data size, which means the bearing fails extremely fast. A better way of making a better prediction is to work on the preprocessing methods. Using more advanced methods than spectrograms may hint at earlier degradation patterns. The preprocess

technologies used in this thesis cannot determine failure signals within only 200-time steps.

One more prediction behavior that can be observed is how the confidence interval changes for different bearings. As stated earlier, bearing 1_3 has a bad confidence interval when only 60% of its data is fed into the algorithm. Still, with more data (95%) fed into the algorithm, the confidence interval converges dramatically, giving a better prediction result. A similar trend is not observed for bearing 2_7, as shown in Figure 53 and Figure 54. This means that the confidence interval depends not only on the amount of data fed into the algorithm, but also on the historical data's behaviour. A similar wide-ranged confidence interval can also be found for bearing 2_3 and bearing 2_5, as shown in Figure 55 and Figure 56. Even though the prediction (green line) gives a good prediction for bearing RUL, the confidence interval incites worry. The historical data for bearing 2_3 and bearing 2_5 is less linear compared to other bearings, and the ARIMA model used to train may be confused by such behavior. Therefore, a possible solution is to identify those bearings before building the ARIMA model and establish ARIMA models based on different bearings with different historical behaviors, which can be done if more sensitive diagnosis methods are used.

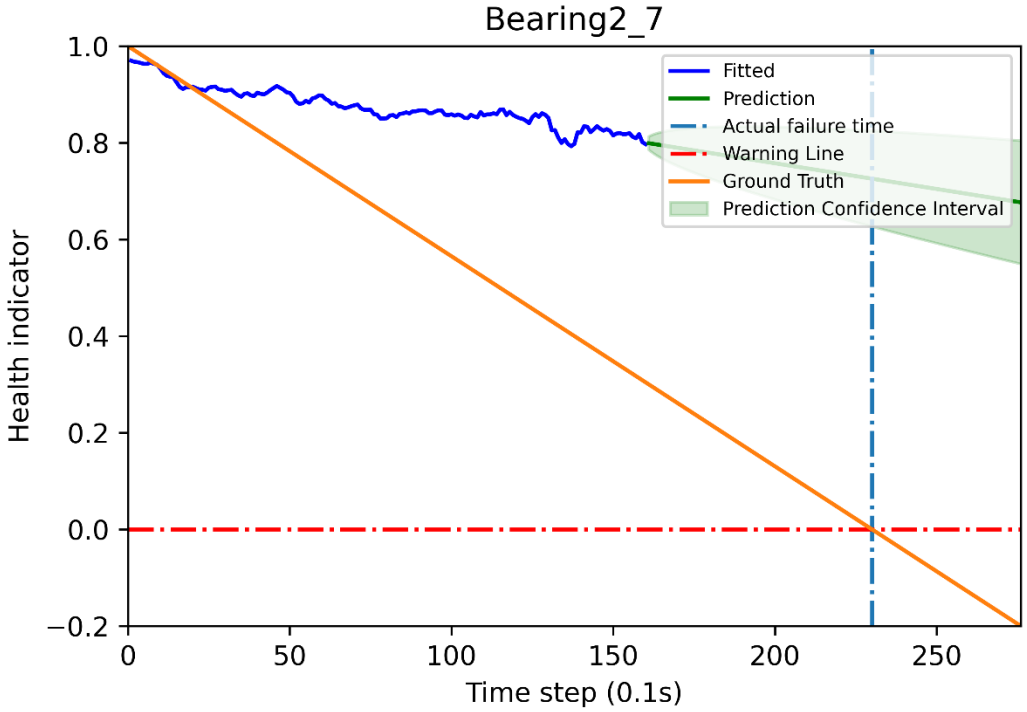


Figure 53 Moving average result for bearing 2_7 with its actual test data prediction

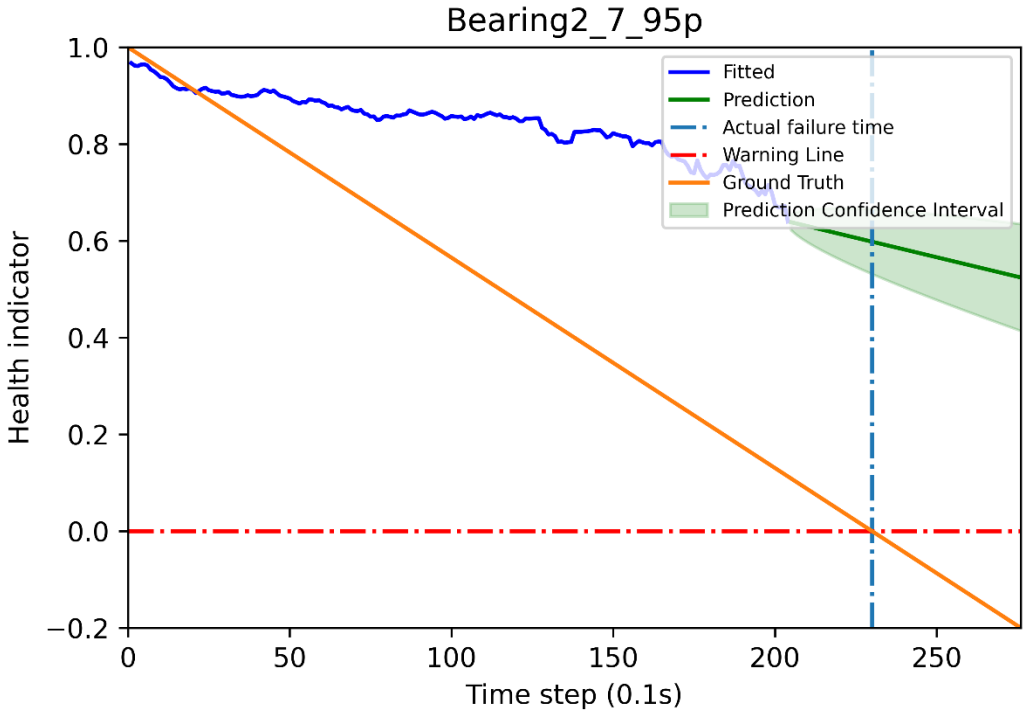


Figure 54 Results of the RUL prediction for bearing 2_7 after 95% of bearing data fed into CNN

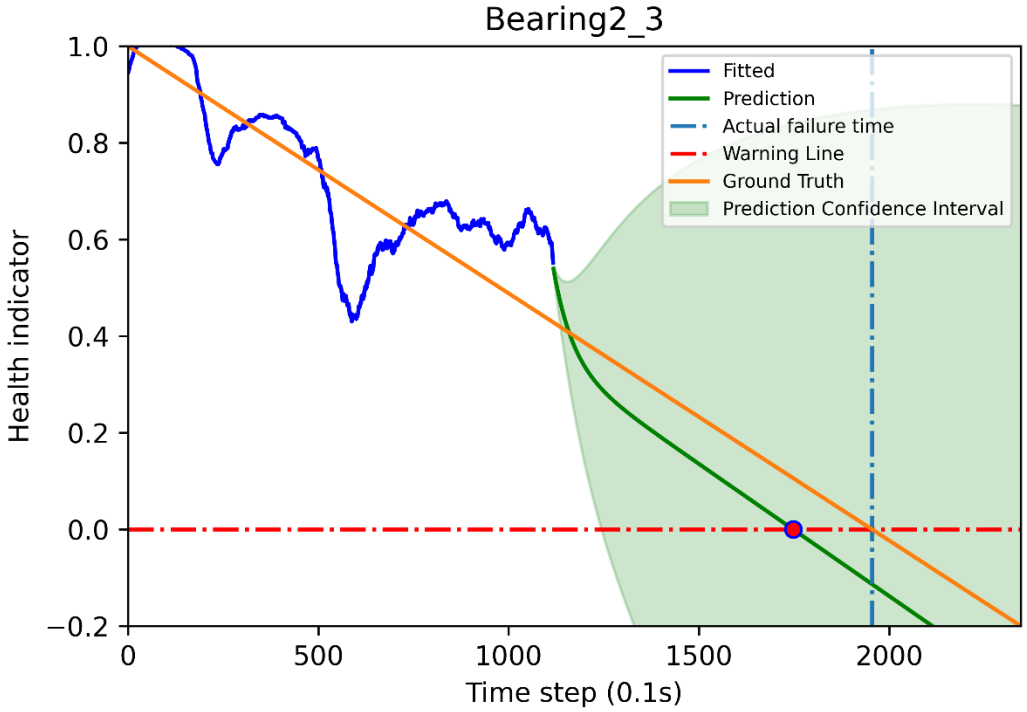


Figure 55 Moving average result for bearing 2_3 with its actual test data prediction

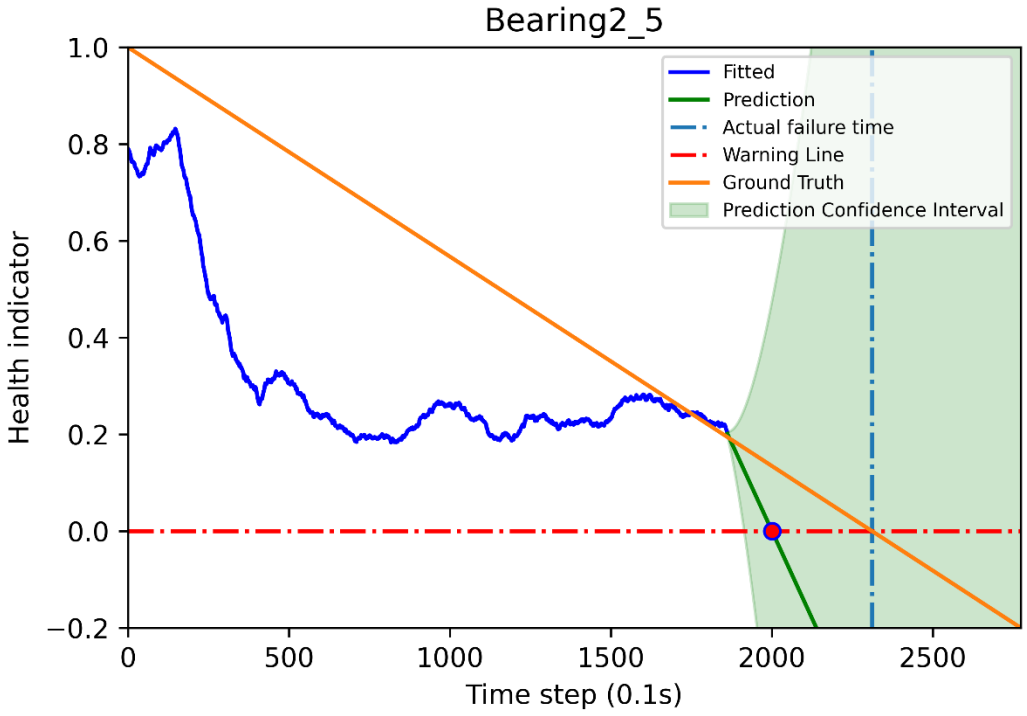


Figure 56 Moving average result for bearing 2_5 with its actual test data prediction

Final bearing RUL predictions are shown in Table 6. The percentage errors indicate how the predicted bearing RUL differs from the actual bearing RUL. Depending on different bearings, these percentage errors can indicate different levels of accuracy. For example, as shown in Figure 57 and Figure 58, the predicted failure time is closer to the actual failure time for bearing 1_6 compared to bearing 1_7, but the percentage error for bearing 1_6 is 124.6%, which is higher than bearing 1_7 (70.5%). The reason is that when the percentage error is calculated using equation (2) in section 3.1, the denominators are different for each bearing. For bearing 1_6 the denominator is small compared to the numerator, and it results in a higher percentage error even though the plots indicate that bearing 1_6 might have a better prediction than bearing 1_7.

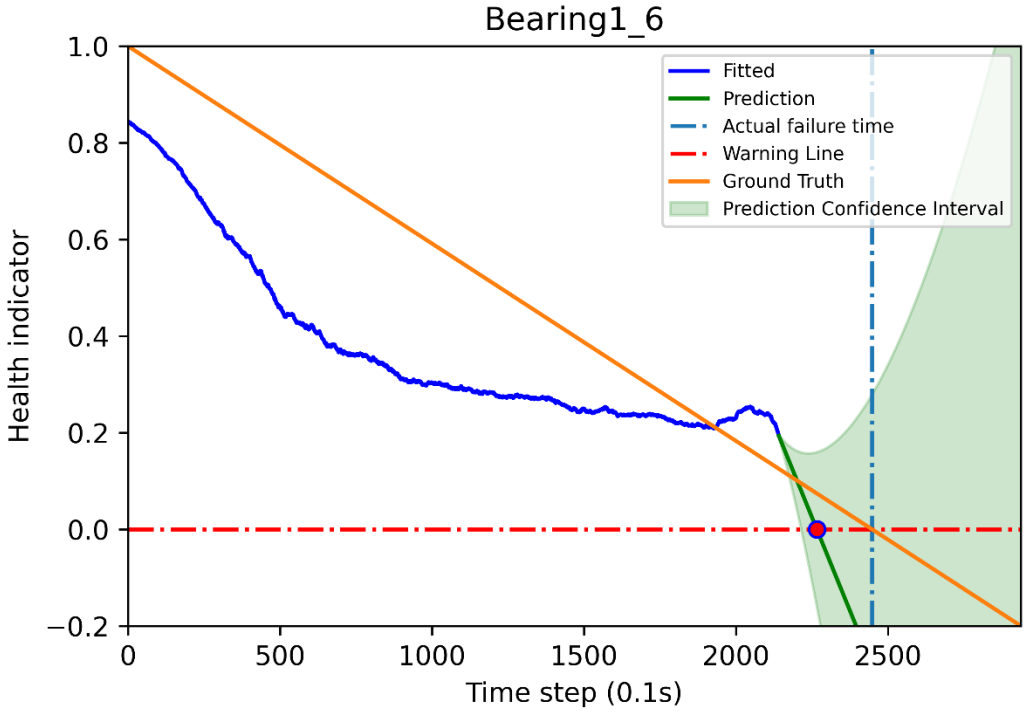


Figure 57 Prediction result plot for bearing 1_6 for channel 1 data using spectrogram

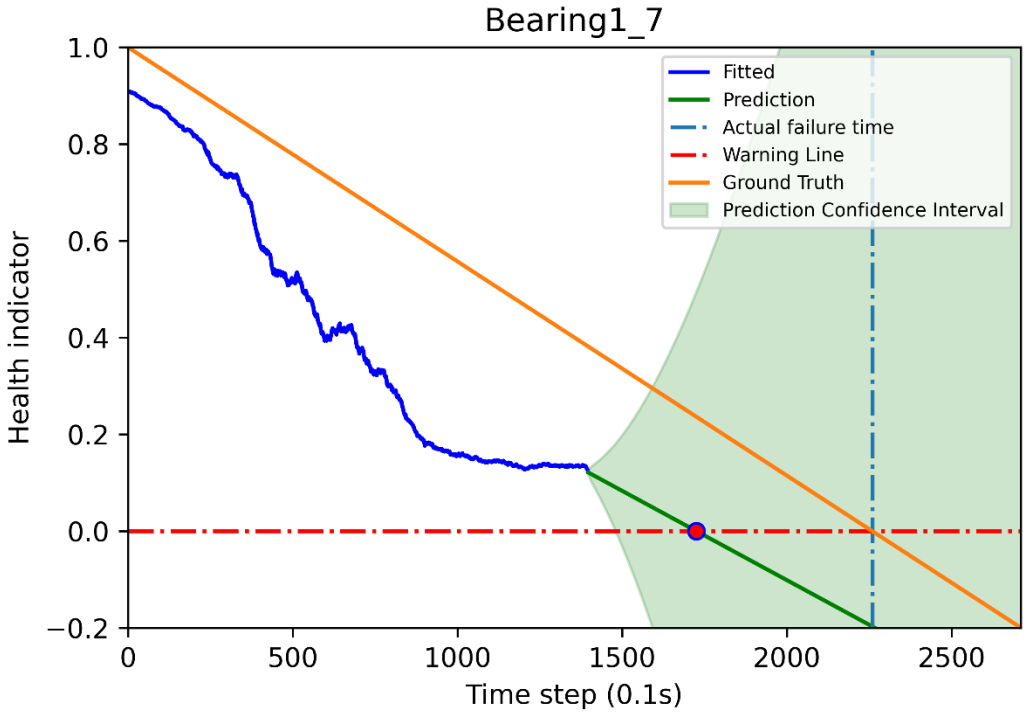


Figure 58 Prediction result plot for bearing 1_7 for channel 1 data using spectrogram

As the table indicates, most predictions are accurate, with only some outliers. For example, bearing 2_7 described above and bearing 2_6, as shown in Figure 59, predicted that the RUL of a bearing last longer than its actual RUL. In this case, the general degradation trend is successfully predicted, but with some noises shown in the 400 to 500 time-step range. In this range, the ARIMA model was found to be confused and failed to give a good prediction. Therefore, if a better preprocessing method, or a better CNN is used, noise could be reduced. However, the method should still be able to identify a reasonable RUL for bearing 2_6. Also, after eliminating bearing 2_7's prediction, which the network does not seem to predict well due to limited original data, the general prediction accuracy was calculated to be 0.188, and the overall percentage error was calculated to be 49.2%.

Table 6 Final spectrogram-based CNN prognosis scores for all bearings using channel 1 data

Spectrogram_C1	Percentage error	Prediction score
Bearing1_3	86.7%	0.049
Bearing1_3_95p	91.5%	0.042
Bearing1_5	24.8%	0.423
Bearing1_6	124.6%	0.013
Bearing1_7	70.5%	0.086
Bearing2_3	27.6%	0.384
Bearing2_4	7.1%	0.779
Bearing2_5	100.6%	0.031
Bearing2_6	-115.5%	0.0001
Bearing2_7	N/A	N/A
Bearing2_7_95p	N/A	N/A
Bearing3_3	74.3%	0.076
Total	49.2%	0.188

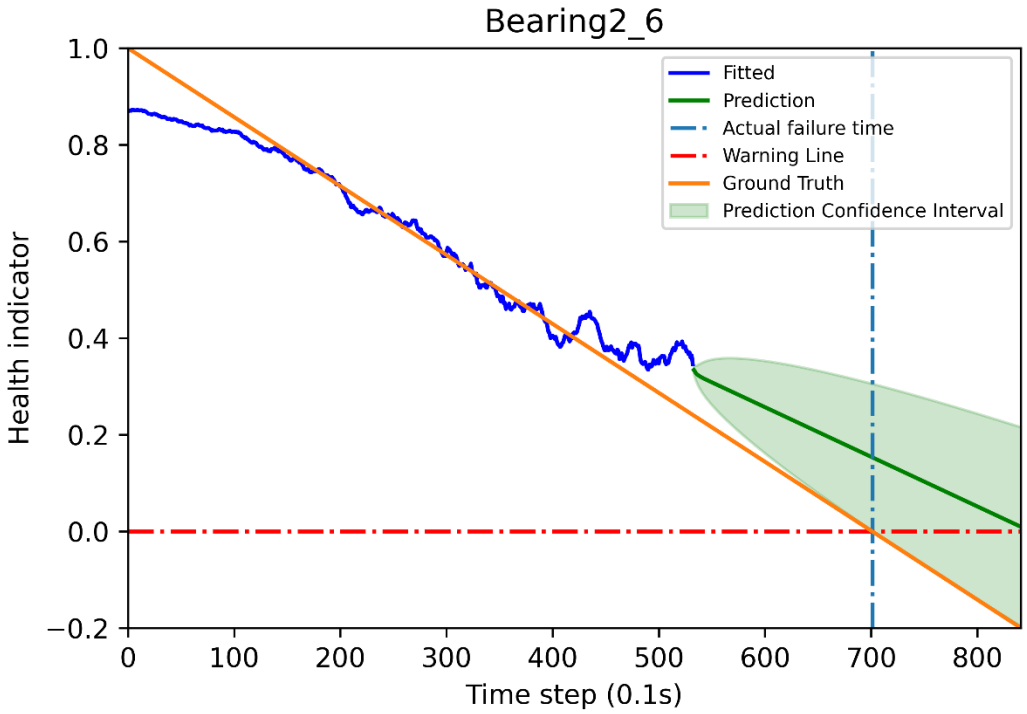


Figure 59 Prediction result for bearing 2_6

Despite how unsatisfying the scores and percentage errors may look; they are designed to land in those low values. The percentage errors are calculated by looking at the difference between actual failure time and the predicted failure time divided by the actual failure time. Therefore, the percentage errors only represent how much the predicted failure time is off from the actual failure time based on the bearing's RUL instead of the bearing's entire life. That is the main reason why the percentage errors are sometimes bigger than what appears on the prognosis plots. The scoring function plot is shown in Figure 60. The scoring function highly penalizes late predictions. Therefore, late predictions can result in much worse score when compared to scores from early predictions that have the same absolute percentage

error. Also, the scoring function only delivers high scores when the percentage error is very small, for example under 2%. For an early prediction with a percentage error of 7.1%, the score is still an "unsatisfying" 0.779, where in other fields, a 7.1% error can mean a very good prediction. Therefore, evaluating performance based on only percentage errors and scores can be confusing. Therefore, considering the prognosis plots themselves remains important.

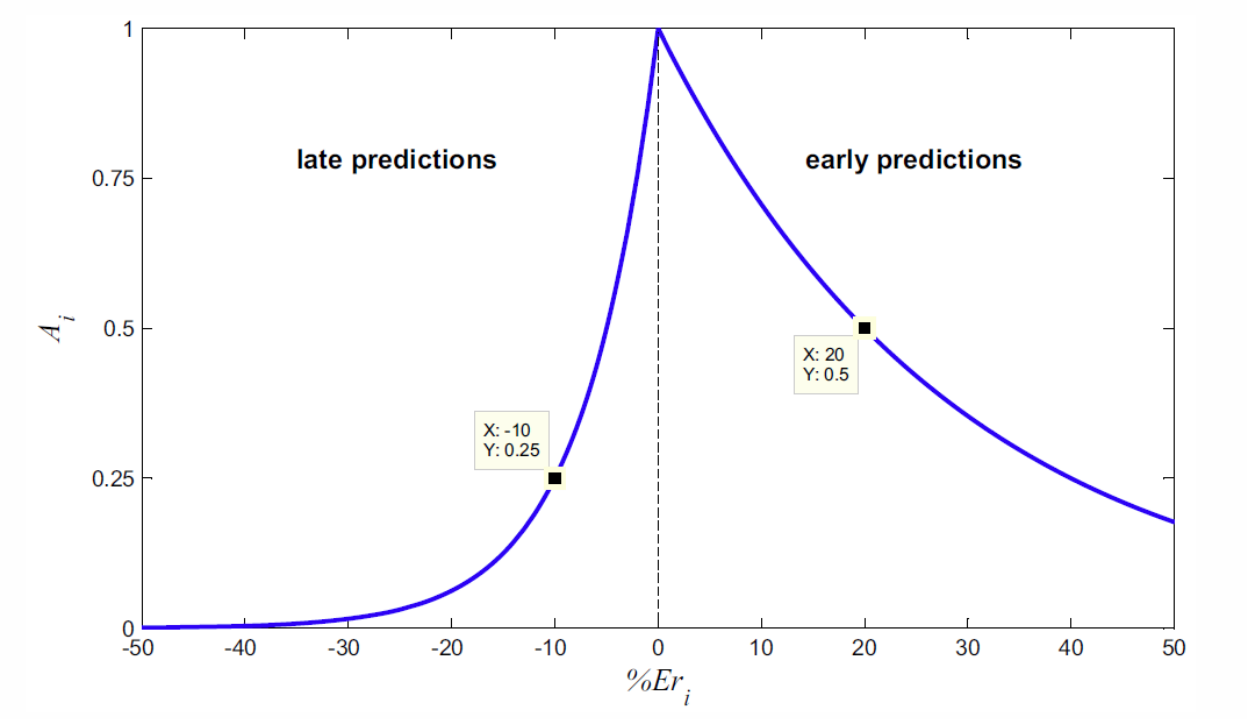


Figure 60 PRONOSTIA scoring function plots [6]

To summarize, combining a CNN with the ARIMA model provides trustworthy prediction results, and it can give reasonable RUL predictions. It is safe to assume that with more data provided, the network can produce better prediction results.

4.2.2 Spectrogram-based CNN for channel 2 data results

The same procedure is used for channel 2 data, but the CNN and ARIMA models are trained with different hyperparameters. However, as shown in Table 5 and Figure A - 4, channel 2 appears to contain more noise than channel 1, possibly confusing the prognosis process and the algorithm. Therefore, final prediction results for channel 2 are found to not be as good. Detailed prediction results are provided in Table 7.

Table 7 Final spectrogram-based CNN prognosis scores for all bearings using channel 2 data

Spectrogram_C2	Percentage error	Prediction score
Bearing1_3	N/A	N/A
Bearing1_3_95p	N/A	N/A
Bearing1_5	136.0%	0.009
Bearing1_6	130.1%	0.011
Bearing1_7	73.4%	0.078
Bearing2_3	27.6%	0.384
Bearing2_4	N/A	N/A
Bearing2_5	-333.9%	Round to 0
Bearing2_6	-164.3%	Round to 0
Bearing2_7	N/A	N/A
Bearing2_7_95p	N/A	N/A
Bearing3_3	N/A	N/A
Total	-21.7%	0.080

As the table indicates, bearing 2_7 remains hard to predict due to its small dataset size. However, this time, bearing 1_3 does not deliver a good prediction, as shown in Figure 61 and Figure 62. The main reason for this is that the early-stage HI values are not adequately predicted by the CNN. As shown in Figure 63, the training dataset

for channel 1 has less noise than the one for channel 2. This noisier dataset for channel 2 causes a worse training result, and eventually leads to a CNN having worse prediction results. Therefore, the HI values for bearing 1_3 for channel 2 data start lower than 0.4, which should start around 1, and remain around 0.2 throughout the whole timeline. This behavior confuses the ARIMA model and finally delivers an inaccurate prediction. With more data fed into the network, the algorithm can more easily find a degradation pattern.

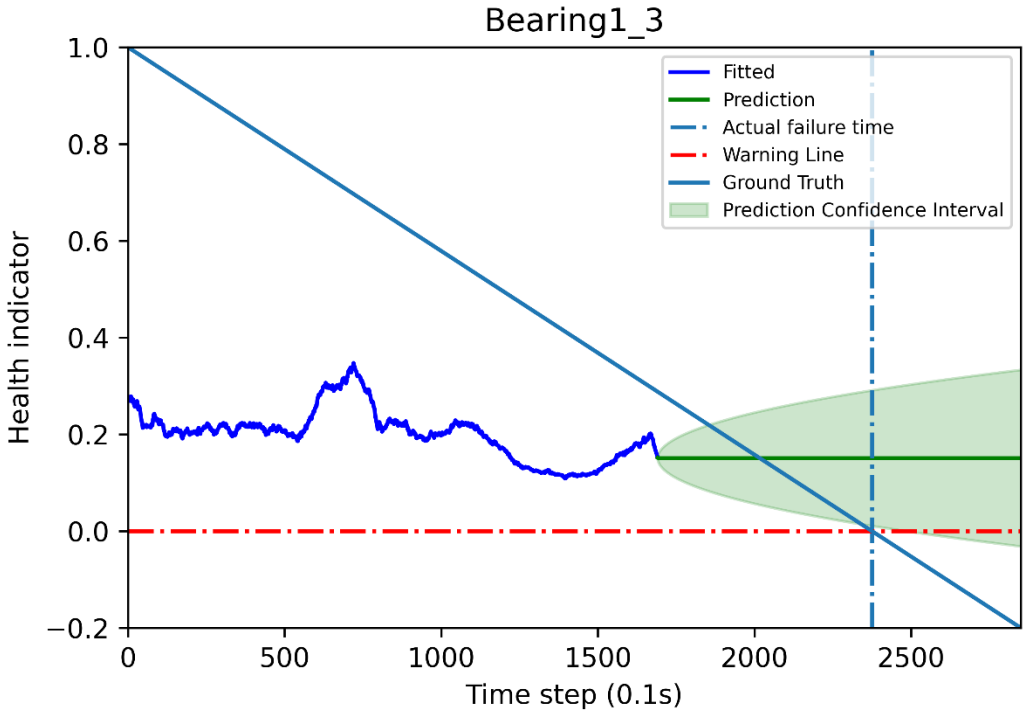


Figure 61 Prediction result for bearing 1_3

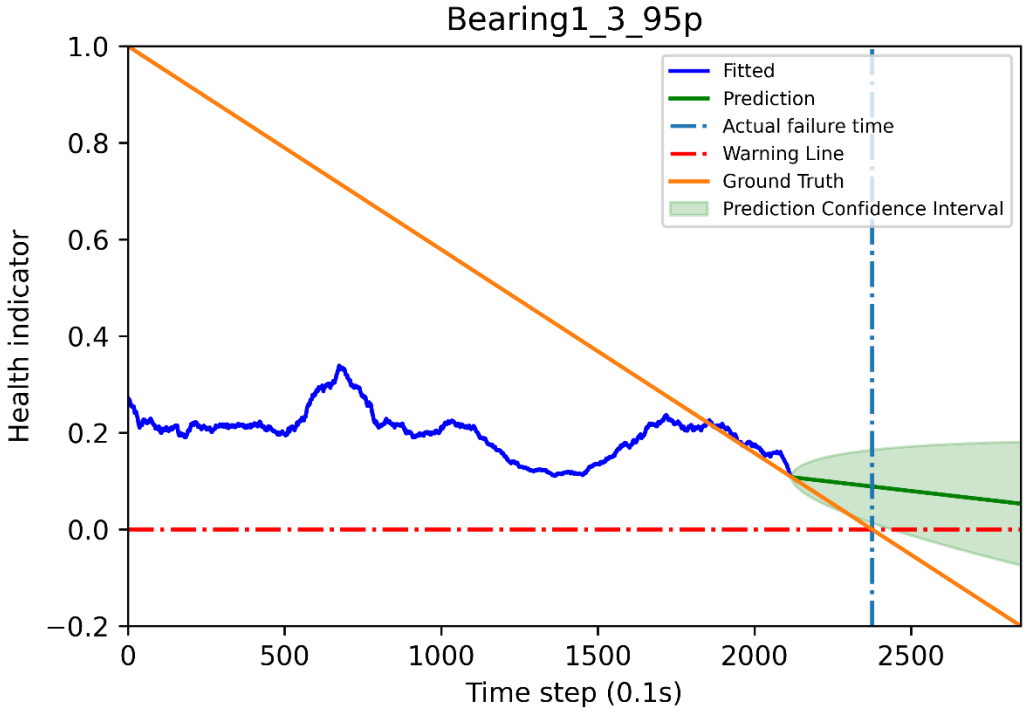


Figure 62 Prediction result for bearing 1_3 with 95% data fed in

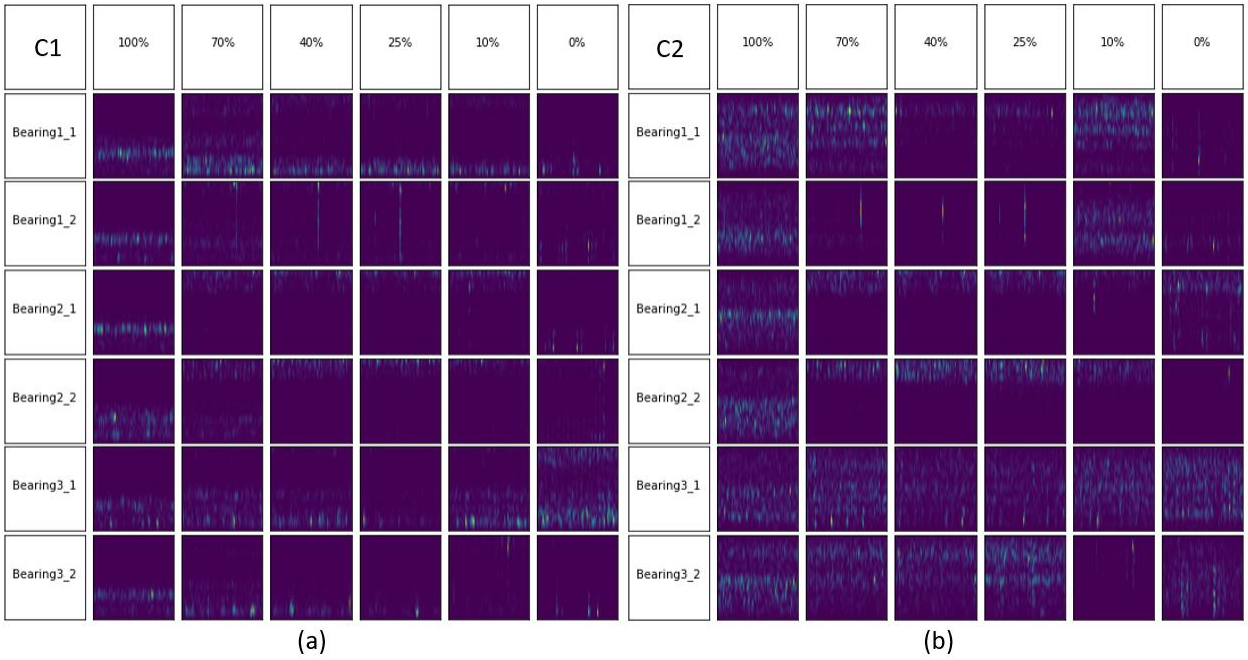


Figure 63 The train dataset comparison for channel 1 and channel 2 data (a)training dataset for channel 1 data (b)training dataset for channel 2 data

Comparing channel 1 and channel 2-based spectrograms in Figure 64, bearing 1_3 delivers a series of valid spectrograms that represent the degradation of the bearing over its life for channel 1 data. However, the spectrogram for channel 2 gives a weaker degradation pattern than channel 1 for the first 50% of the bearing's life. Training and test datasets from channel 2 output spectrograms with less valuable information, which leads to worse prediction results when compared to channel 1 data.

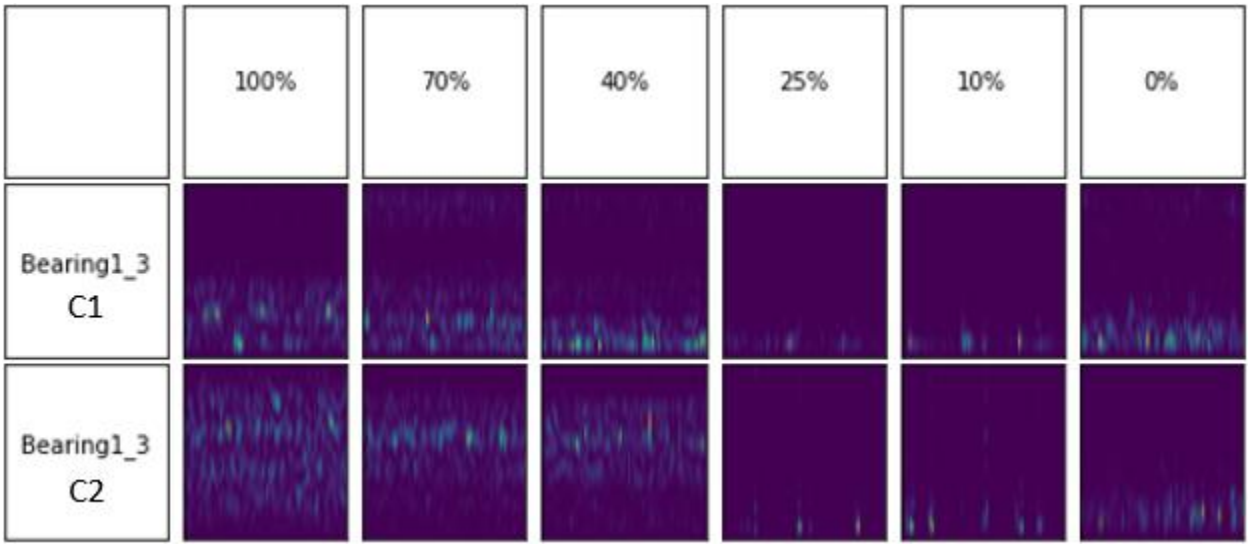


Figure 64 Spectrograms based on two channels of data for bearing 1_3

To summarize, by looking at the final prediction results, using channel 1 data as an input for the spectrogram-based CNN is shown to be more efficient than using channel 2 data. A hypothesis for the vertically placed accelerometer's (channel 1) better performance when compared to the horizontally placed accelerometer (channel 2) is its closer proximity to the bearing's loading zone. In this case, the horizontally placed accelerometer is completely opposite to the loading zone and may see less vibration than its vertical counterpart. However, it is not safe to say that

channel 2 data is always worse than channel 1 data, as stated in section 4.1.3, channel 2 data can pick up looseness signals for diagnosis, whereas channel 1 data cannot. Nonetheless, further investigation would be required to determine the actual reason for the difference in signals provided by each channel.

4.2.3 Spectrogram-based CNN using channel fusion results

Instead of using channel 1 or channel 2 separately as a CNN input, a channel fusion method can be used to combine both trained CNNs to train a new CNN. The best hyperparameter combinations remain the same, and both datasets from each channel can be processed by the CNN simultaneously. The results from each branch are then used to generate a new prediction for HI values, as shown in Table 8. From the table, it can be seen that more bearing RULs can be predicted, and a better overall prediction result is obtained when compared to channel 2 data alone. But bearing 2_4 and bearing 2_7 still cannot be predicted. Bearing 2_4 is well predicted by a channel 1-based CNN, but since channel 2 is also involved in this channel fusion prediction process, bearing 2_4 predictions suffer. As shown in Figure 65, channel 2-based spectrograms for bearing 2_4 are heavily contaminated by noises compared to channel 1, and combined with the reason provided in section 4.2.2, the dataset deteriorates the final prediction.

Table 8 Spectrogram-based CNN prediction results using the channel fusion method

Spectrogram_CF	Percentage error	Prediction score
Bearing1_3	39.4%	0.255
Bearing1_3_95p	-66.4%	0.0001
Bearing1_5	88.2%	0.047
Bearing1_6	43.2%	0.244
Bearing1_7	96.0%	0.035
Bearing2_3	N/A	N/A
Bearing2_4	N/A	N/A
Bearing2_5	101.9%	0.029
Bearing2_6	86.8%	0.049
Bearing2_7	N/A	N/A
Bearing2_7_95p	N/A	N/A
Bearing3_3	79.2%	0.064
Total	58.5%	0.091

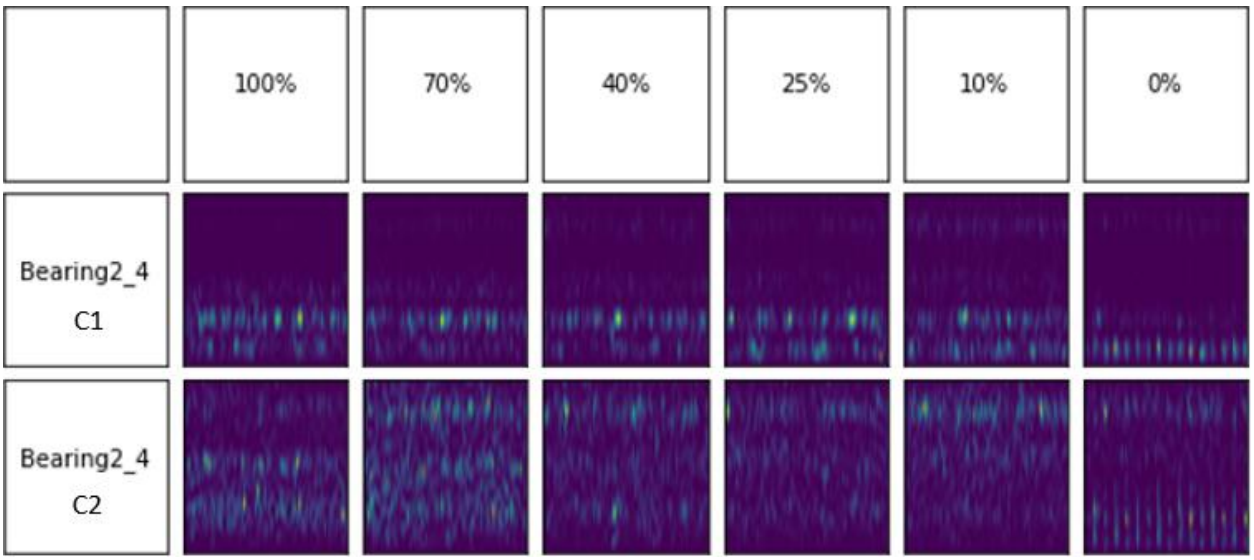


Figure 65 Spectrograms based on two channels of data for bearing 2_4

To summarize, using channel 1 and channel 2 data simultaneously can help fix the confusion caused by channel 2's noise. But it also limits the prediction results compared to the results obtained using channel 1 data alone.

4.2.4 Scalogram-based CNN for channel 1 data results

In this section, scalograms are used as the preprocessing method for the CNN input. The goal is to determine if scalograms can offer a superior preprocessing method for the proposed algorithm. The general procedure remains similar to the one presented in section 4.2.1, and the final prediction results are provided in Table 9.

Table 9 Scalogram-based CNN prediction results using channel 1

Scalogram_C1	Percentage error	Prediction score
Bearing1_3	-85.2%	0.0007
Bearing1_3_95p	116.8%	0.002
Bearing1_5	N/A	N/A
Bearing1_6	85.6%	0.052
Bearing1_7	79.9%	0.063
Bearing2_3	53.7%	0.155
Bearing2_4	-6.5%	0.408
Bearing2_5	N/A	N/A
Bearing2_6	35.6%	0.291
Bearing2_7	N/A	N/A
Bearing2_7_95p	N/A	N/A
Bearing3_3	-96.3%	0.0001
Total	-30.6%	0.098

As shown in Figure 66, scalograms perform well at providing fault diagnosis during the final stages of a bearing's life. They indicate the failure frequencies and time stamps when failures appear. In addition, the final failure plots at the end are well distinguished from healthy ones. However, failure degradation patterns for scalograms in (b) do not show as strongly as spectrograms in (a). For scalograms, when compared with the final failure plots, the healthy plots are too similar, and it creates a problem for CNNs to identify the fault features. Therefore, the overall prediction performance of scalogram-based CNNs is worse than expected.

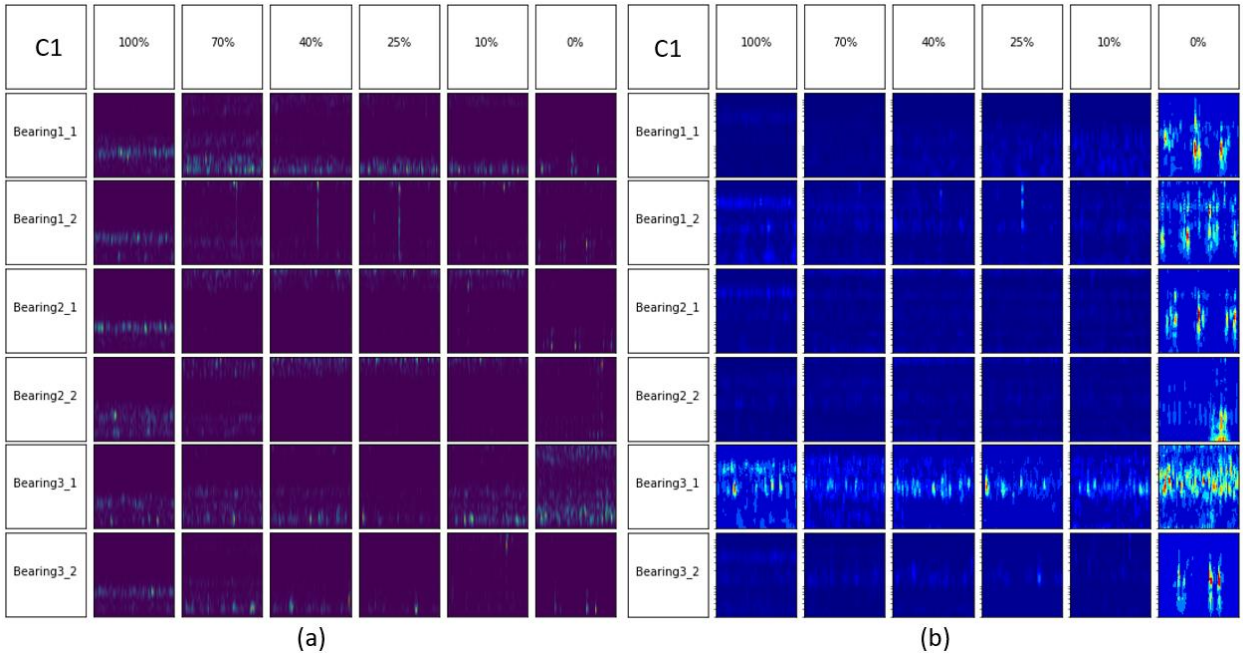


Figure 66 Comparison of spectrograms and scalograms for channel 1 data (a) spectrograms (b) scalograms

Even though scalogram-based CNNs do not deliver convincing results, the method still provides valuable information for future work. For example, when using scalograms, a similar behavior to spectrograms is observed, and when more data is

available, the algorithm gives more accurate predictions. Similarly, the small data size for bearing 2_7 still performs poorly for prediction. This means that the general idea of the proposed algorithm works, and the algorithm is expected to work well when a better and richer dataset is available. In addition, the scalogram-based CNN for channel 1 data outperforms spectrogram-based CNNs for channel 2 data. This may suggest that compared to using different preprocessing methods, the quality of the dataset plays a more critical role in delivering accurate results.

4.3 The relationship between diagnosis and prognosis

In this section, prognosis results for spectrogram-based CNNs for channel 1 data from section 4.2.1, which had the most accurate prediction results, are compared with the diagnosis results shown in section 4.1. The reconstructed diagnosis results for the test bearings are provided in Table 10. As stated in Table 10, three major fault types are obtained: inner race faults, outer race faults, and unidentified faults. One thing to notice is that every diagnosis result is obtained using the last few recording samples from each bearing. Therefore, it is possible that earlier analysis of the data may not result in the same diagnosis of fault types. And from the prognosis result plots, shown in Appendix B (a) and provided in Table 10, every test bearing's RUL prediction is evaluated empirically as one of three levels: good, bad and medium. "Good" indicates the predicted failure point is close to the actual failure point and the confidence interval is narrow compared to others; "Bad" is used for predicted failure

points that are either too far from the actual failure time, or the predicted failure time occurs after the actual failure time; "Medium" is for results that are in between.

Table 10 Final diagnosis results for test bearings using channel 1 data

	Channel 1	Percentage error	Prediction by plots (Appendix B (a))
Bearing1_3	IN	86.7%	Medium
Bearing1_5	IN	24.8%	Good
Bearing1_6	IN	124.6%	Good
Bearing1_7	IN	70.5%	Bad
Bearing2_3	N/A	27.6%	Medium
Bearing2_4	OUT	7.1%	Good
Bearing2_5	N/A	100.6%	Good
Bearing2_6	OUT	-115.5%	Bad
Bearing2_7	N/A	74.3%	Bad
Bearing3_3	OUT	49.2%	Medium

For bearings that are not well diagnosed, bearing 2_3, bearing 2_5 and bearing 2_7, their prognosis accuracies are very different. For example, bearing 2_3 has unclear fault types through diagnosis of channel 1 data, but received a decent prediction percentage error and medium accurate prediction. However, recall from Table 5 in section 4.1, bearing 2_3's channel 2-based diagnosis indicates its fault as looseness with unstable rotating speed. Moreover, for bearing 2_5, similar behaviors are found. While channel 1-based diagnosis cannot determine the fault type, the channel 2-based diagnosis determines that bearing 2_5 has a looseness problem. Its prognosis result also provides good prediction plots, as shown in Figure 67. Therefore, it is reasonable to assume that the ML prognosis procedure can pick up failure

features hidden from the diagnosis technic used in this thesis. Alternatively, it can also pick up failure features even when a "misplaced" accelerator provides bad recordings. For bearing 2_7, the problem of having too few data samples is similar for both diagnosis and prognosis, providing a satisfying result.

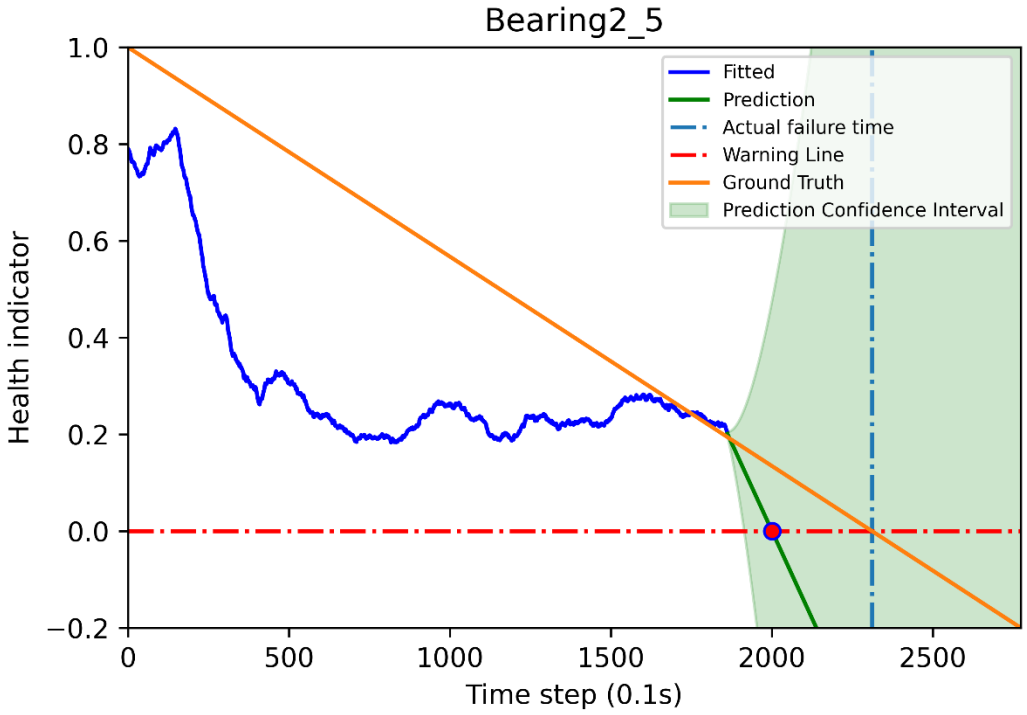


Figure 67 Prediction plot for bearing 2_5 based on a spectrogram-based CNN using channel 1 data

On the other hand, for inner race faulted bearings: bearing 1_3, bearing 1_5, bearing 1_6, and bearing 1_7, their prognosis results also differ from each other. For bearing 1_5 and bearing 1_6, they appear to have similar degradation patterns, as shown in Figure 68, and their prognosis results turn out to be the best among inner race faulted bearings.

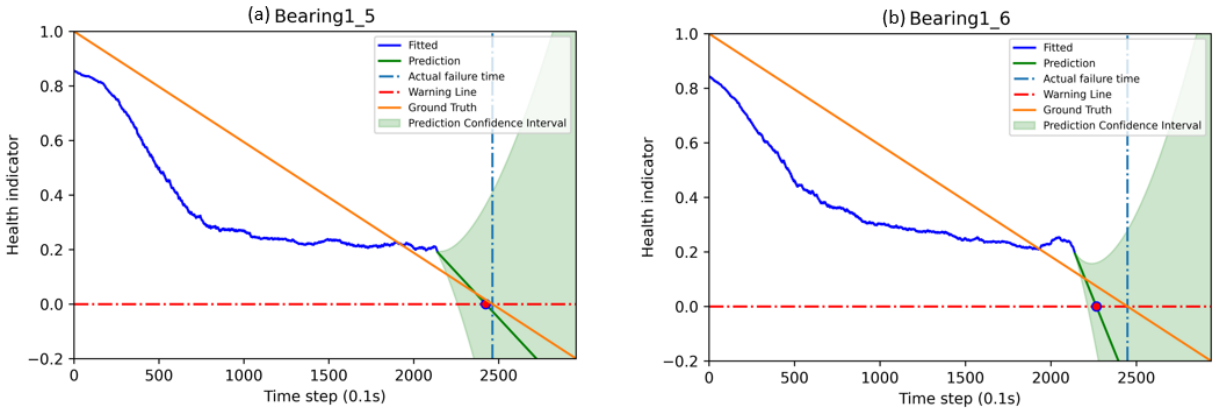


Figure 68 Prediction plots for (a) bearing 1_5 based on a spectrogram-based CNN using channel 1 data, (b) bearing 1_6 based on a spectrogram-based CNN using channel 1 data

Interestingly, diagnosis results for bearing 1_5 suggest an inner race fault for channel 1 data and an outer race fault for channel 2 data, as shown in Figure 69 and Figure 70. Bearing 1_6 does not appear to have an outer race fault for channel 2 data, as shown in Figure 71 and Figure 72. This might suggest that the dominant source of failure for bearing 1_5 is an inner race fault, as it has a similar degradation pattern as bearing 1_6. Moreover, prognosis results seem to ignore a possible outer race fault in bearing 1_5, and gives similar degradation patterns for bearing 1_5 and bearing 1_6. This suggests two possibilities: prognosis results do not depend on the failure type, or prognosis results depend on the dominant failure type for a given bearing.

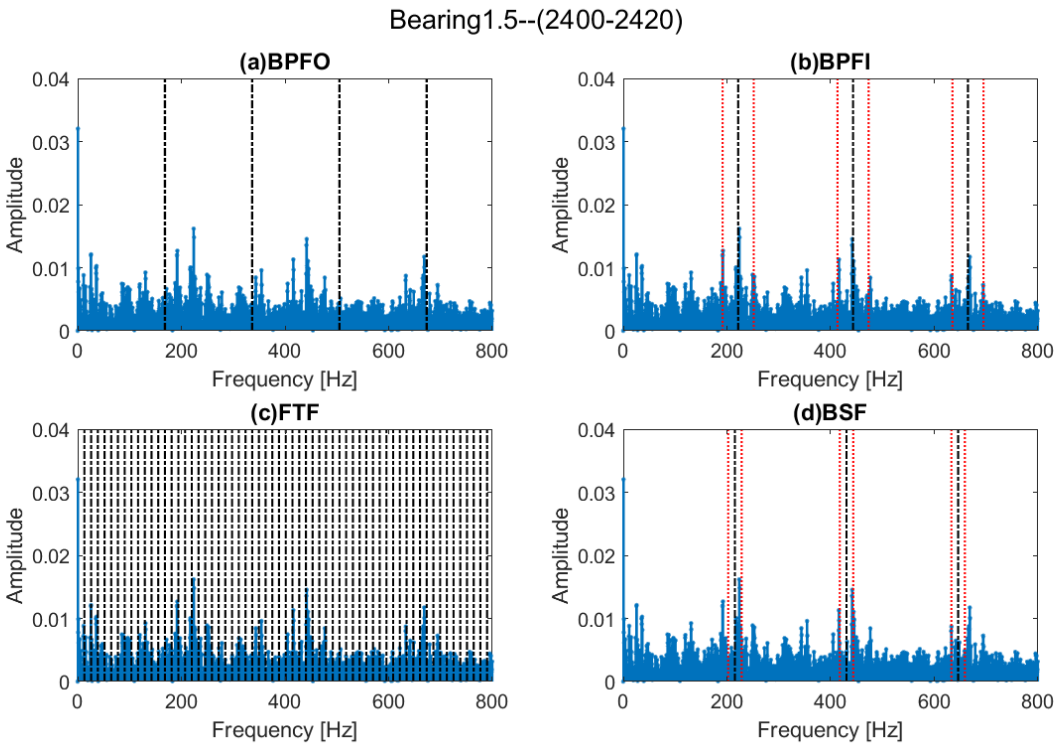


Figure 69 Bearing1_5 channel 1 diagnosis results

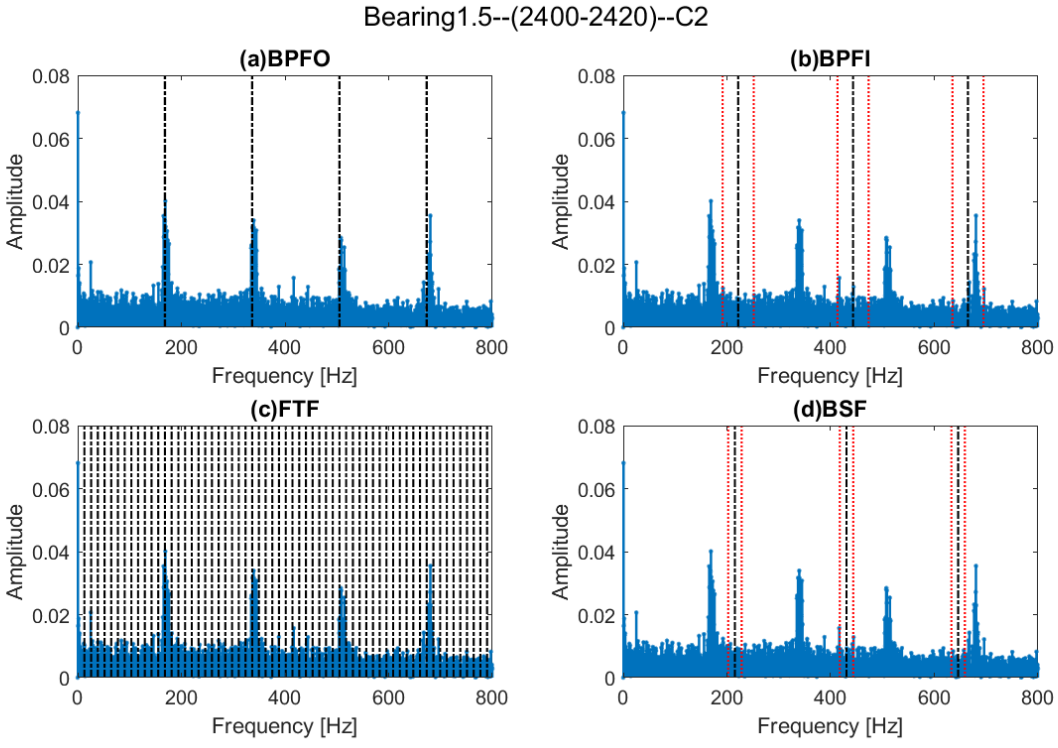


Figure 70 Bearing1_5 channel 2 diagnosis results

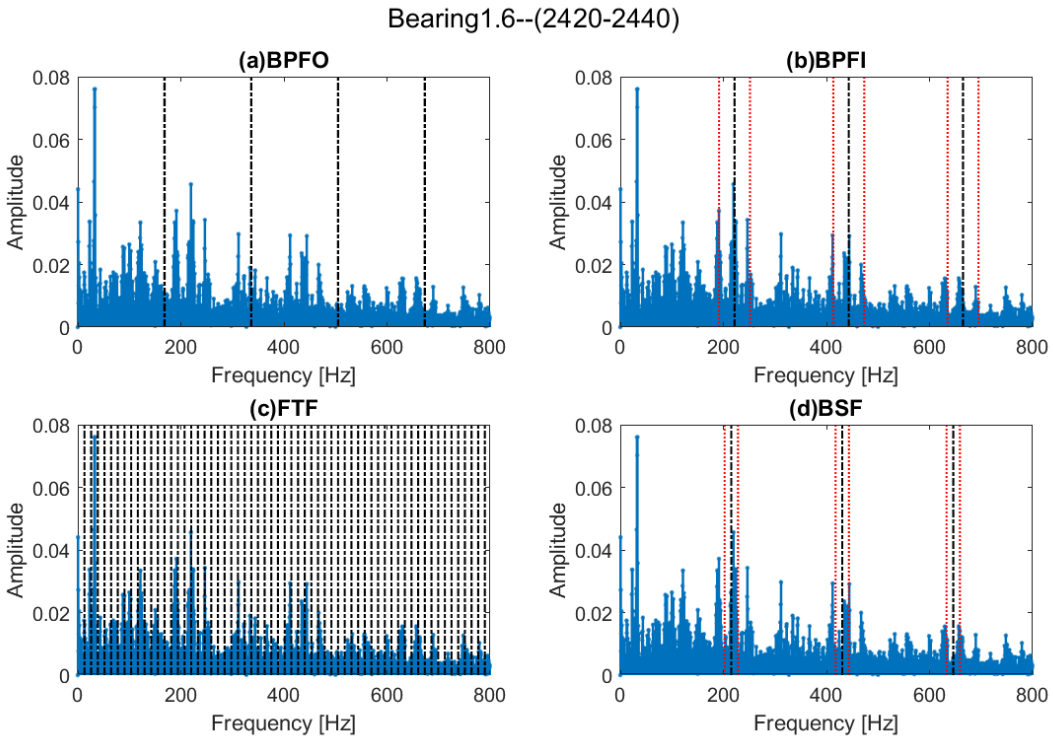


Figure 71 Bearing1_6 channel 1 diagnosis results

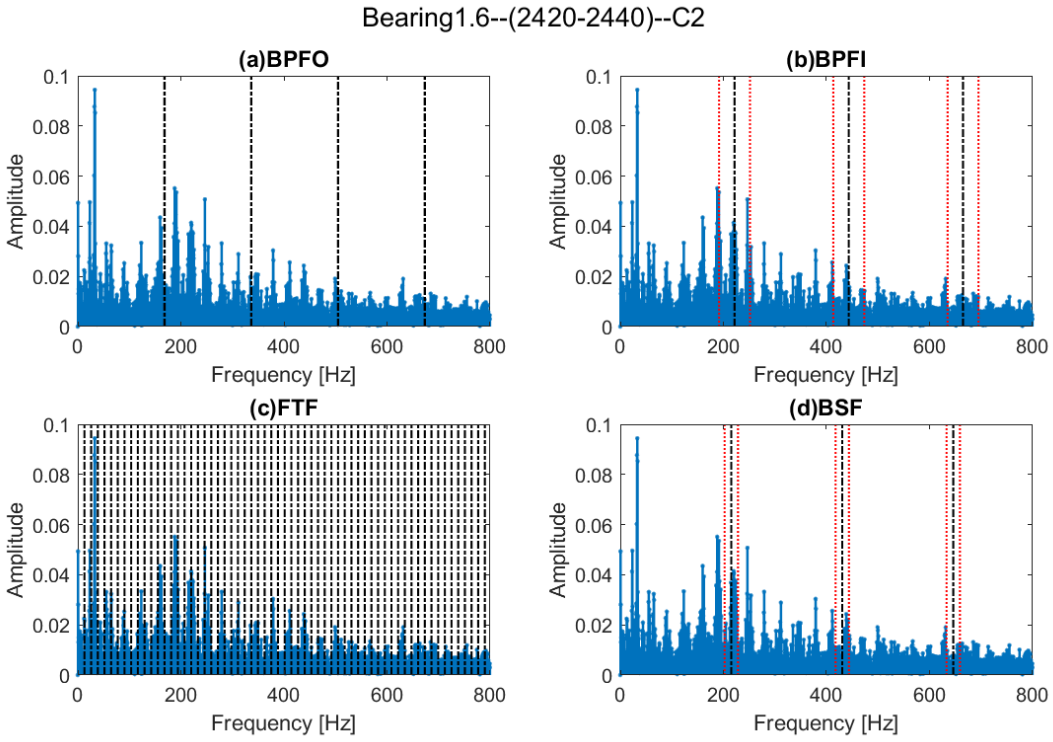


Figure 72 Bearing1_6 channel 2 diagnosis results

As for bearing 1_3 and bearing 1_7, their degradation patterns appear to be the opposite, as shown in the red circles of Figure 73. Bearing 1_3 appears to have a degradation that occurs later than bearing 1_7. Moreover, these failure patterns appear to be hidden features of the diagnosis procedure, as no such behaviors are obtained even when mid-life recording samples are fed into the diagnosis process, as shown in Figure 74 and Figure 75. Both bearings' mid-life recoding-based diagnosis results are not able to identify any faults.

The gentle degradation pattern that bearing 1_3 has provides better prognosis results than bearing 1_7, which suggests that a smoother degradation process can increase the accuracy of prognosis results. When diagnosis is used as a method to determine a bearing's degradation pattern, diagnosis must be performed over and over again at many intervals over the bearing's life. However, prognosis appears to be superior at identifying bearing degradation patterns, and may be further developed in future studies to improve the analysis of bearing faults.

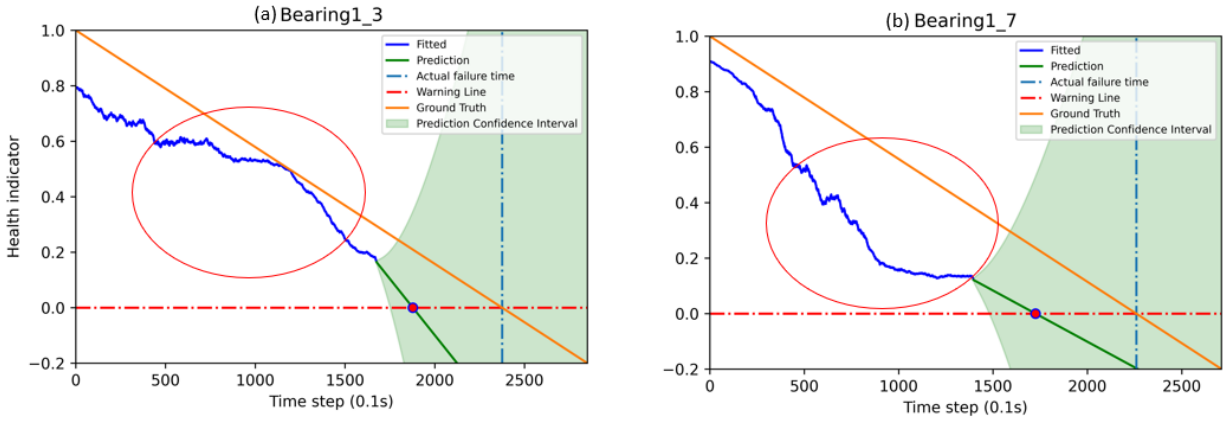


Figure 73 Prediction plots for (a) bearing 1_3 based on a spectrogram-based CNN using channel 1 data, (b) bearing 1_7 based on a spectrogram-based CNN using channel 1 data

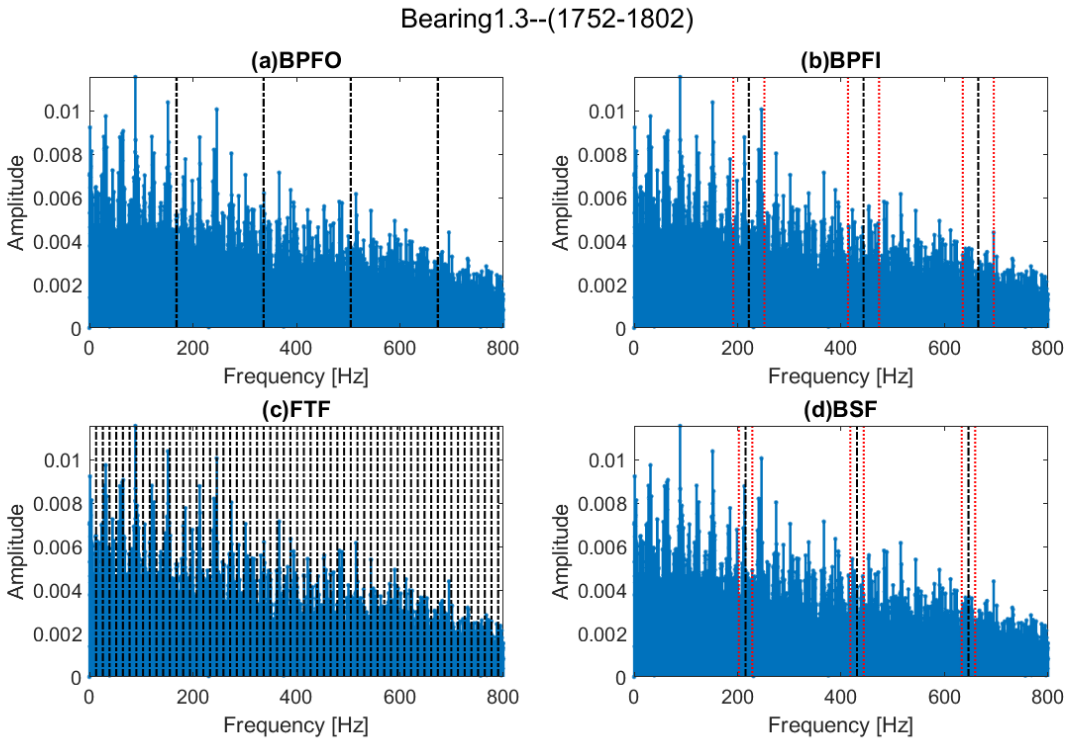


Figure 74 Bearing1_3 channel 1 diagnosis results with mid-life recordings provided

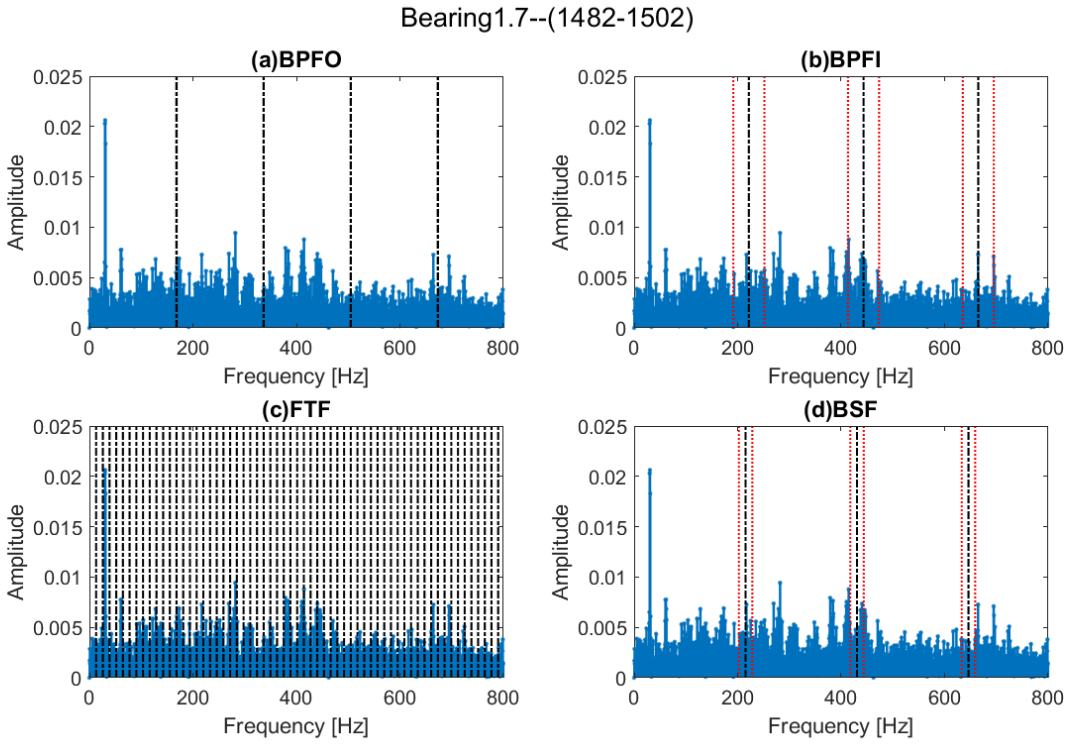


Figure 75 Bearing1_7 channel 1 diagnosis results with mid-life recordings provided

Furthermore, bearing 2_4, bearing 2_6, and bearing 3_3 also provide informative results for outer race faults. As shown in the red circles of Figure 76, bearing 2_4 shows the opposite degradation pattern when compared to bearing 3_3. As mentioned before, a smoother degradation pattern results in a better prognosis accuracy for bearing 2_4 than bearing 3_3.

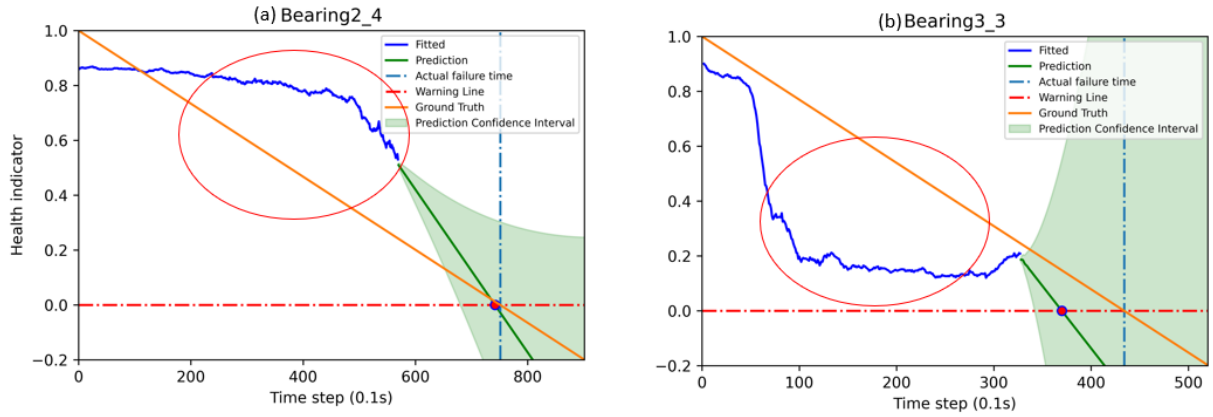


Figure 76 Prediction plots for (a) bearing 2_4 based on a spectrogram-based CNN using channel 1 data, (b) bearing 3_3 based on a spectrogram-based CNN using channel 1 data

Finally, the degradation pattern for bearing 2_6 is smooth at first, but starts to vary as more recording samples are introduced, as shown in Figure 77. This unstable degradation behavior also worsens the prognosis results.

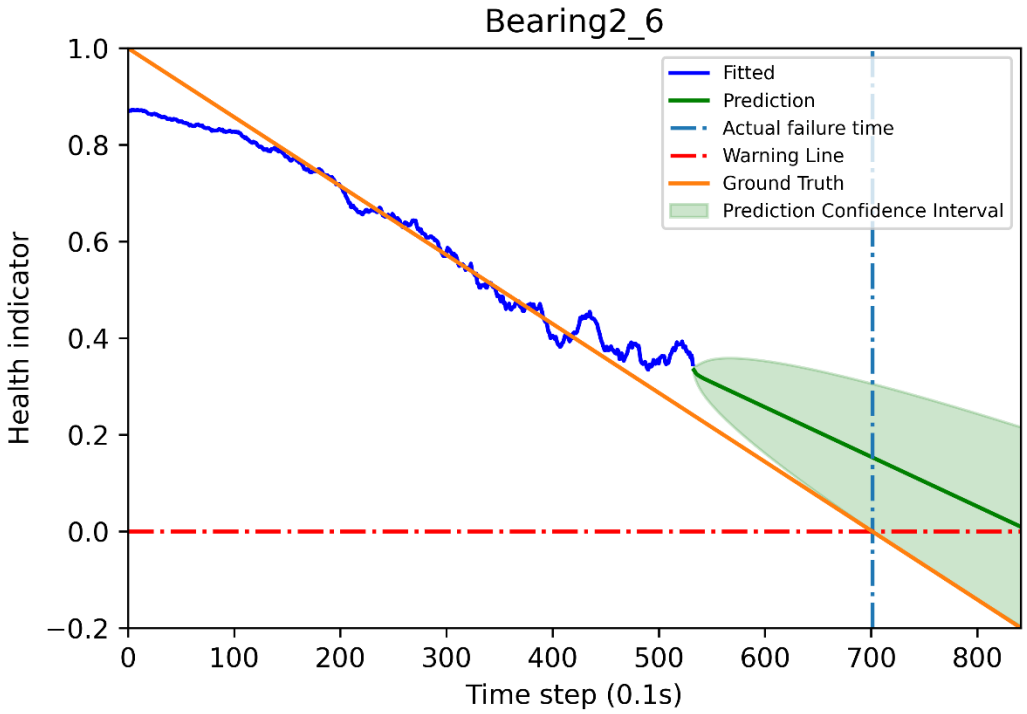


Figure 77 Prediction plots for bearing 2_6 based on a spectrogram-based CNN using channel 1 data

Therefore, several relationships between diagnosis and prognosis can be concluded:

- Prognosis can pick up failure features that sometimes are hidden from the diagnosis procedure, and prognosis can also provide degradation patterns that might be helpful for future bearing fault analysis.
- Smoother degradation patterns tend to result in a more accurate bearing RUL prediction. In which case, diagnosis is not the best choice for providing degradation patterns.
- Prognosis results might depend on the dominant failure type in a bearing when multiple failure types are detected via diagnosis.

- Prognosis results do not appear to be directly related to the failure type, but rather, depend on how the bearing degrades over time.

Chapter 5 Conclusions and Future Work

In this thesis the following objectives were achieved:

(1) Using diagnosis methods to determine the specific failure modes for the PRONOSTIA dataset.

(2) Building an algorithm that consists of spectrogram and scalogram-based CNNs and an ARIMA model to help predict bearing RUL.

(3) Comparing prediction results when certain parameters are introduced, for example, preprocessed methods and two datasets collected by horizontally and vertically placed accelerometers.

(4) Combining prediction results with diagnosis results to further understanding of how to improve the algorithm.

No prior results are available for the bearing fault diagnosis of the PRONOSTIA dataset, with the main reason being the limited data density mentioned in section 3.2. However, by combining multiple short recordings into longer sample sets, diagnosis of the PRONOSTIA dataset was possible. In particular, the results were found to be clear in most cases, and failure modes were determined for most bearings. The major bearing failures in the PRONOSTIA dataset were found to be inner and outer race faults. Two bearings show a potential looseness issue, and it was further confirmed

by introducing channel 2 data. The results also show that channel 2 data contains more noise than channel 1 data. Since the final prognosis results that used channel 1 data are generally more accurate than using channel 2 data, this may indicate that the vertically placed accelerometer is better than the horizontally placed accelerometer for capturing the vibration signal. Furthermore, the forces used to perform accelerated life testing for condition 2 and condition 3 in the PRONOSTIA dataset are 4200N and 5000N, respectively. However, the bearings used in this experiment have a maximum dynamic load of 4000N, which should also be the maximum force applied to perform accelerated life tests for these bearings. Therefore, the effects of overloading may have contributed to the results herein.

The algorithm based on using a CNN and ARIMA model has been shown to make valid predictions. Moreover, the algorithm is also shown to make more accurate predictions when using channel 1 data compared to channel 2 data, and spectrograms outperform scalograms in this dataset. However, the final prediction scores are not as high as some studies, where full test bearing life data are used. Nonetheless, existing studies use either full datasets to determine RUL, which provides predictions when the answer is already given, or use ambiguous thresholding methods based on unclear methodologies [82], [83]. For example, Yoo and Beak use "expert experiences" to determine their thresholds, and Hong et al. set their thresholds without explanation at all [42], [84]. Therefore, the ARIMA model is introduced, as a time series forecasting method, to give RUL predictions based on HI values. And the proposed methods not only deliver more reasonable predictions, but also point out

that spectrogram-based CNNs generally work better than scalogram-based CNNs when using the ARIMA model. Moreover, channel 1 data is once more shown to provide better predictions than channel 2 data. The proposed method is also found to have drawbacks in that a RUL prediction for bearing 2_7 is not satisfying, and the reason is assumed to be the very small data size. This limits the ability to predict the RUL for bearings that fail rapidly, when the algorithms cannot determine failure patterns in time.

The differences between using spectrograms and scalograms as CNN inputs are thoroughly discussed. The advantages and disadvantages of using each preprocessing method are considered, and the most appropriate one for the application can be used in the future when comparing new preprocessing methods. Lastly, the main factor that influence prognosis results appears to be the degradation pattern of the bearing rather than the fault type, as was originally expected.

As a result of conducting experiments on the same bearings under the same loading conditions, no evidence is found to conclude that one bearing can represent all other bearings that fail in a similar time and working environment. This is due to the stochastic nature of fault initiation in bearings. This means that no universal diagnosis, nor prognosis, can be done in advance of obtaining the bearing's experimental data. Therefore, no statistical analysis of the PROGNOSTIA dataset was provided, but a broader study could be conducted in the future that takes into account failure statics.

In conclusion, the proposed methods are proven useful for achieving this thesis's goals and their differences are compared during the process. Recommendations for future work are as follows:

(1) Identifying better preprocessing methods that can pick up failure behaviors in early stages of bearing operation to isolate the degradation patterns buried in heavy noise, as well as discovering new prognosis methods that can use those diagnosis results to improve predictions.

(2) Improving the use of CNNs by introducing combined network structures like long-short term memory (LSTM) networks or autoencoder methods. With combined algorithms, degradation patterns may be picked up more accurately and efficiently.

(3) Trying different forecasting methods can also help to improve RUL predictions of bearings.

(4) Since training and testing deep learning algorithms can be time-consuming, proposing methods that can help reduce the non-characteristic bearing signals would make the whole process more efficient.

(5) A greater number of relationships between diagnosis and prognosis are expected to be found in the future, as being able to categorize bearings according to fault types may be crucial in improving RUL predictions.

References

[1] "Determining types of bearing damage," *Pit & Quarry*, Oct. 30, 2017. https://www.pitandquarry.com/determining-types-of-bearing-damage/

[2] Palmgren, A., "Die Lebensdauer von Kugellagern (The Service Life of Ball Bearings),".

[3] "Bearing damage and failure analysis." Accessed: May 30, 2020. [Online]. Available: https://www.skf.com/binaries/pub12/Images/0901d1968064c148-Bearing-failures---14219_2-EN_tcm_12-297619.pdf

[4] "Bearing faults condition monitoring – A literature survey," *ResearchGate*. https://www.researchgate.net/publication/260037824_Bearing_faults_condition_monitoring_-_A_literature_survey

[5] N. Tandon and A. Choudhury, "A review of vibration and acoustic measurement methods for the detection of defects in rolling element bearings," *Tribol. Int.*, vol. 32, no. 8, pp. 469–480, Aug. 1999, doi: 10.1016/S0301-679X(99)00077-8.

[6] P. Nectoux *et al.*, "PRONOSTIA: An experimental platform for bearings accelerated degradation tests.," p. 9.

[7] F. Elasha, M. Greaves, D. Mba, and A. Addali, "Application of Acoustic Emission in Diagnostic of Bearing Faults within a Helicopter Gearbox," *Procedia CIRP*, vol. 38, pp. 30–36, Jan. 2015, doi: 10.1016/j.procir.2015.08.042.

[8] B. Van Hecke, J. Yoon, and D. He, "Low speed bearing fault diagnosis using acoustic emission sensors," *Appl. Acoust.*, vol. 105, pp. 35–44, Apr. 2016, doi: 10.1016/j.apacoust.2015.10.028.

[9] M. S. Yilmaz and E. Ayaz, "Adaptive neuro-fuzzy inference system for bearing fault detection in induction motors using temperature, current, vibration data," in *IEEE EUROCON 2009*, May 2009, pp. 1140–1145. doi: 10.1109/EURCON.2009.5167779.

[10] A. Choudhary, S. L. Shimi, and A. Akula, "Bearing Fault Diagnosis of Induction Motor Using Thermal Imaging," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, Sep. 2018, pp. 950–955. doi: 10.1109/GUCON.2018.8674889.

[11] T. Wang, M. Liang, J. Li, and W. Cheng, "Rolling element bearing fault diagnosis via fault characteristic order (FCO) analysis," *Mech. Syst. Signal Process.*, vol. 45, no. 1, pp. 139–153, 2014, doi: 10.1016/j.ymssp.2013.11.011.

[12] M. Kang, J. Kim, L. M. Wills, and J.-M. Kim, "Time-Varying and Multiresolution Envelope Analysis and Discriminative Feature Analysis for Bearing Fault Diagnosis," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7749–7761, Dec. 2015, doi: 10.1109/TIE.2015.2460242.

[13] S. Kim, D. An, and J.-H. Choi, "Diagnostics 101: A Tutorial for Fault Diagnostics of Rolling Element Bearing Using Envelope Analysis in MATLAB," *Appl. Sci.*, vol. 10, no. 20, Art. no. 20, Jan. 2020, doi: 10.3390/app10207302.

[14] S. E. Pandarakone, M. Masuko, Y. Mizuno, and H. Nakamura, "Deep Neural Network Based Bearing Fault Diagnosis of Induction Motor Using Fast Fourier

Transform Analysis," in *2018 IEEE Energy Conversion Congress and Exposition (ECCE)*, Sep. 2018, pp. 3214–3221. doi: 10.1109/ECCE.2018.8557651.
[15] X. Lou and K. A. Loparo, "Bearing fault diagnosis based on wavelet transform and fuzzy inference," *Mech. Syst. Signal Process.*, vol. 18, no. 5, pp. 1077–1095, Sep. 2004, doi: 10.1016/S0888-3270(03)00077-3.
[16] M. Kang, J. Kim, L. M. Wills, and J.-M. Kim, "Time-Varying and Multiresolution Envelope Analysis and Discriminative Feature Analysis for Bearing Fault Diagnosis," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7749–7761, Dec. 2015, doi: 10.1109/TIE.2015.2460242.
[17] R. Guo and B. Gong, "Research on remaining useful life of rolling bearings using EWT-DI-ALSTM," *Meas. Sci. Technol.*, vol. 33, no. 9, 2022, doi: 10.1088/1361-6501/ac6ec9.
[18] S. Hanly, "Vibration Analysis: FFT, PSD, and Spectrogram Basics [Free Download]." https://blog.endaq.com/vibration-analysis-fft-psd-and-spectrogram
[19] Y.-H. Byeon, S.-B. Pan, and K.-C. Kwak, "Intelligent Deep Models Based on Scalograms of Electrocardiogram Signals for Biometrics," *Sensors*, vol. 19, no. 4, Art. no. 4, Jan. 2019, doi: 10.3390/s19040935.
[20] J. Li, X. Li, and D. He, "A Directed Acyclic Graph Network Combined With CNN and LSTM for Remaining Useful Life Prediction," *IEEE Access*, vol. 7, pp. 75464–75475, 2019, doi: 10.1109/ACCESS.2019.2919566.
[21] L. Ren, Y. Sun, H. Wang, and L. Zhang, "Prediction of Bearing Remaining Useful Life With Deep Convolution Neural Network," *IEEE Access*, vol. 6, pp. 13041–13049, 2018, doi: 10.1109/ACCESS.2018.2804930.
[22] J. Zhu, N. Chen, and W. Peng, "Estimation of Bearing Remaining Useful Life Based on Multiscale Convolutional Neural Network," *IEEE Trans. Ind. Electron.*, vol. 66, no. 4, pp. 3208–3216, Apr. 2019, doi: 10.1109/TIE.2018.2844856.
[23] C. C. Tan, "Application of Acoustic Emission to the Detection of Bearing Failures," *Int. Tribol. Conf. 1990 Brisb. 2-5 Dec. 1990 Putt. Tribol. Work Reliab. Maintainab. Lubr. Wear Technol. Prepr. Pap.*, p. 110, 1990.
[24] Y. Wang, J. Xiang, R. Markert, and M. Liang, "Spectral kurtosis for fault detection, diagnosis and prognostics of rotating machines: A review with applications," *Mech. Syst. Signal Process.*, vol. 66–67, pp. 679–698, Jan. 2016, doi: 10.1016/j.ymssp.2015.04.039.
[25] M. Kang, J. Kim, and J.-M. Kim, "An FPGA-Based Multicore System for Real-Time Bearing Fault Diagnosis Using Ultrasampling Rate AE Signals," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2319–2329, Apr. 2015, doi: 10.1109/TIE.2014.2361317.
[26] D. T. Hoang and H. J. Kang, "A Motor Current Signal-Based Bearing Fault Diagnosis Using Deep Learning and Information Fusion," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 6, pp. 3325–3333, Jun. 2020, doi: 10.1109/TIM.2019.2933119.
[27] C. Lessmeier, J. K. Kimotho, D. Zimmer, and W. Sextro, "Condition Monitoring of Bearing Damage in Electromechanical Drive Systems by Using Motor Current Signals of Electric Motors: A Benchmark Data Set for Data-Driven Classification," p. 17, 2016.

[28] G. Karatzinis, Y. S. Boutalis, and Y. L. Karnavas, "Motor Fault Detection and Diagnosis Using Fuzzy Cognitive Networks with Functional Weights," in *2018 26th Mediterranean Conference on Control and Automation (MED)*, Jun. 2018, pp. 709–714. doi: 10.1109/MED.2018.8443043.

[29] M. Jason, "Spectrum Analysis The key features of analyzing spectra," SKF USA Inc., May 2002. [Online]. Available: https://www.skf.com/binaries/pub12/Images/0901d1968024acef-CM5118-EN-Spectrum-Analysis_tcm_12-113997.pdf

[30] J. Ben Ali, N. Fnaiech, L. Saidi, B. Chebel-Morello, and F. Fnaiech, "Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals," *Appl. Acoust.*, vol. 89, pp. 16–27, Mar. 2015, doi: 10.1016/j.apacoust.2014.08.016.

[31] B. Li, M.-Y. Chow, Y. Tipsuwan, and J. C. Hung, "Neural-network-based motor rolling bearing fault diagnosis," *IEEE Trans. Ind. Electron.*, vol. 47, no. 5, pp. 1060–1069, Oct. 2000, doi: 10.1109/41.873214.

[32] B. Ronald Newbold and B. Ronald N, *The Fourier transform and its applications.*, vol. Vol. 31999. New York: McGraw-Hill, 1986.

[33] I. Grattan-Guinness and J. B. J. Fourier, *Joseph Fourier, 1768-1830; a survey of his life and work, based on a critical edition of his monograph on the propagation of heat, presented to the Institut de France in 1807*. MIT Press, 1972. Accessed: May 18, 2021. [Online]. Available: https://dspace.mit.edu/handle/1721.1/1724

[34] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*. Society for Industrial and Applied Mathematics, 1992. doi: 10.1137/1.9781611970999.

[35] D. Strömbergsson, P. Marklund, K. Berglund, and P. Larsson, "Bearing monitoring in the wind turbine drivetrain: A comparative study of the FFT and wavelet transforms," *Wind Energy*, vol. 23, Jun. 2020, doi: 10.1002/we.2491.

[36] V. Purushotham, S. Narayanan, and S. Prasad a N, "Multi-fault diagnosis of rolling bearing elements using wavelet analysis and hidden Markov model based fault recognition," *NDT E Int.*, vol. 38, pp. 654–664, Dec. 2005, doi: 10.1016/j.ndteint.2005.04.003.

[37] M. T. Pham, J.-M. Kim, and C. H. Kim, "Accurate Bearing Fault Diagnosis under Variable Shaft Speed using Convolutional Neural Networks and Vibration Spectrogram," *Appl. Sci.*, vol. 10, no. 18, Art. no. 18, Jan. 2020, doi: 10.3390/app10186385.

[38] "STFT Spectrograms VI - NI LabVIEW 8.6 Help." https://zone.ni.com/reference/en-XX/help/371361E-01/lvanls/stft_spectrogram_core/#details

[39] Ahmet Taspinar, "A guide for using the Wavelet Transform in Machine Learning," Dec. 21, 2018. https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

[40] J. C. GARCIA-PRADA, C. Castejón, and O. Lara, "Incipient bearing fault diagnosis using DWT for feature extraction," pp. 18–21, Jan. 2007.

[41] "Continuous and Discrete Wavelet Transforms - MATLAB & Simulink." https://www.mathworks.com/help/wavelet/gs/continuous-and-discrete-wavelet-transforms.html (accessed May 20, 2021).
[42] Y. Yoo and J.-G. Baek, "A Novel Image Feature for the Remaining Useful Lifetime Prediction of Bearings Based on Continuous Wavelet Transform and Convolutional Neural Network," *Appl. Sci.*, vol. 8, no. 7, Art. no. 7, Jul. 2018, doi: 10.3390/app8071102.
[43] D. Gao, Y. Zhu, X. Wang, K. Yan, and J. Hong, "A Fault Diagnosis Method of Rolling Bearing Based on Complex Morlet CWT and CNN," in *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, Oct. 2018, pp. 1101–1105. doi: 10.1109/PHM-Chongqing.2018.00194.
[44] P. Konar and P. Chattopadhyay, "Bearing fault detection of induction motor using wavelet and Support Vector Machines (SVMs)," *Appl. Soft Comput.*, vol. 11, no. 6, pp. 4203–4211, 2011, doi: 10.1016/j.asoc.2011.03.014.
[45] Y. Liu, "Fault diagnosis based on SWPT and Hilbert transform," *Procedia Eng.*, vol. 15, pp. 3881–3885, Jan. 2011, doi: 10.1016/j.proeng.2011.08.726.
[46] F. Immovilli, M. Cocconcelli, A. Bellini, and R. Rubini, "Detection of Generalized-Roughness Bearing Fault by Spectral-Kurtosis Energy of Vibration or Current Signals," *IEEE Trans. Ind. Electron.*, vol. 56, no. 11, pp. 4710–4717, Nov. 2009, doi: 10.1109/TIE.2009.2025288.
[47] J. Antoni, "The spectral kurtosis: a useful tool for characterising non-stationary signals," *Mech. Syst. Signal Process.*, vol. 20, no. 2, pp. 282–307, Feb. 2006, doi: 10.1016/j.ymssp.2004.09.001.
[48] J. Tian, C. Morillo, M. H. Azarian, and M. Pecht, "Motor Bearing Fault Detection Using Spectral Kurtosis-Based Feature Extraction Coupled With K-Nearest Neighbor Distance Analysis," *IEEE Trans. Ind. Electron.*, vol. 63, no. 3, pp. 1793–1803, Mar. 2016, doi: 10.1109/TIE.2015.2509913.
[49] Y. Zhang and R. B. Randall, "Rolling element bearing fault diagnosis based on the combination of genetic algorithms and fast kurtogram," *Mech. Syst. Signal Process.*, vol. 23, no. 5, pp. 1509–1517, Jul. 2009, doi: 10.1016/j.ymssp.2009.02.003.
[50] D. Ciro, "Supervised and Unsupervised Learning," Astronomy Colloquia. USA, 2011 - sites.astro.caltech.edu, Apr. 2011.
[51] Y. Yang, D. Yu, and J. Cheng, "A fault diagnosis approach for roller bearing based on IMF envelope spectrum and SVM," *Measurement*, vol. 40, no. 9, pp. 943–950, Nov. 2007, doi: 10.1016/j.measurement.2006.10.010.
[52] D. Fernández-Francos, D. Martínez-Rego, O. Fontenla-Romero, and A. Alonso-Betanzos, "Automatic bearing fault diagnosis based on one-class v-SVM," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 357–365, Jan. 2013, doi: 10.1016/j.cie.2012.10.013.
[53] L. Auria and R. A. Moro, "Support Vector Machines (SVM) as a Technique for Solvency Analysis," *SSRN Electron. J.*, 2008, doi: 10.2139/ssrn.1424949.
[54] S. Huang, N. Cai, P. P. Pacheco, S. Narrandes, Y. Wang, and W. Xu, "Applications of Support Vector Machine (SVM) Learning in Cancer Genomics," *Cancer Genomics Proteomics*, vol. 15, no. 1, pp. 41–51, Jan. 2018.

[55] "K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint," *www.javatpoint.com*. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning (accessed May 20, 2021).
[56] "LAWRENCE, S., et al. 'Face Recognition : A Convolutional Neural-Network Approach: Artificial Neural Networks and Statistical Pattern Recognition.' IEEE Transactions on Neural Networks, vol. 8, no. 1, Institute of Electrical and Electronics Engineers, 1997, pp. 98–113.".
[57] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017, doi: 10.1109/TIP.2017.2662206.
[58] M. Hussain, J. J. Bird, and D. R. Faria, "A Study on CNN Transfer Learning for Image Classification," in *Advances in Computational Intelligence Systems*, Cham, 2019, pp. 191–202. doi: 10.1007/978-3-319-97982-3_16.
[59] L. Eren, T. Ince, and S. Kiranyaz, "A Generic Intelligent Bearing Fault Diagnosis System Using Compact Adaptive 1D CNN Classifier," *J. Signal Process. Syst.*, vol. 91, no. 2, pp. 179–189, Feb. 2019, doi: 10.1007/s11265-018-1378-3.
[60] M. F. Yaqub, I. Gondal, and J. Kamruzzaman, "Inchoate Fault Detection Framework: Adaptive Selection of Wavelet Nodes and Cumulant Orders," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 3, pp. 685–695, Mar. 2012, doi: 10.1109/TIM.2011.2172112.
[61] Q. Hu, Z. He, Z. Zhang, and Y. Zi, "Fault diagnosis of rotating machinery based on improved wavelet package transform and SVMs ensemble," *Mech. Syst. Signal Process.*, vol. 21, no. 2, pp. 688–705, Feb. 2007, doi: 10.1016/j.ymssp.2006.01.007.
[62] D. Wang, Q. Guo, Y. Song, S. Gao, and Y. Li, "Application of Multiscale Learning Neural Network Based on CNN in Bearing Fault Diagnosis," *J. Signal Process. Syst.*, vol. 91, no. 10, pp. 1205–1217, Oct. 2019, doi: 10.1007/s11265-019-01461-w.
[63] E. Sutrisno, H. Oh, A. S. S. Vasan, and M. Pecht, "Estimation of remaining useful life of ball bearings using data driven methodologies," in *2012 IEEE Conference on Prognostics and Health Management*, Jun. 2012, pp. 1–7. doi: 10.1109/ICPHM.2012.6299548.
[64] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, Mar. 2014, pp. 106–112. doi: 10.1109/UKSim.2014.67.
[65] D. Benvenuto, M. Giovanetti, L. Vassallo, S. Angeletti, and M. Ciccozzi, "Application of the ARIMA model on the COVID-2019 epidemic dataset," *Data Brief*, vol. 29, p. 105340, Apr. 2020, doi: 10.1016/j.dib.2020.105340.
[66] Z. Sun, X. Hu, and K. Dong, "Remaining Useful Life Prediction of Quay Crane Hoist Gearbox Bearing under Dynamic Operating Conditions Based on ARIMA-CAPF Framework," *Shock Vib.*, vol. 2021, p. e9403401, Dec. 2021, doi: 10.1155/2021/9403401.

[67] Z. Luo, X.-B. Wang, and Z.-X. Yang, “An Improved Recursive ARIMA Method with Recurrent Process for Remaining Useful Life Estimation of Bearings,” *Shock Vib.*, vol. 2022, p. e9010419, Feb. 2022, doi: 10.1155/2022/9010419.
[68] Y. Lu, Q. Li, and S. Y. Liang, “Physics-based intelligent prognosis for rolling bearing with fault feature extraction,” *Int. J. Adv. Manuf. Technol.*, vol. 97, no. 1, pp. 611–620, Jul. 2018, doi: 10.1007/s00170-018-1959-0.
[69] O. Barndorff-Nielsen and G. Schou, “On the parametrization of autoregressive models by partial autocorrelations,” *J. Multivar. Anal.*, vol. 3, no. 4, pp. 408–419, Dec. 1973, doi: 10.1016/0047-259X(73)90030-4.
[70] N. Sawalhi, R. B. Randall, and H. Endo, “The enhancement of fault detection and diagnosis in rolling element bearings using minimum entropy deconvolution combined with spectral kurtosis,” *Mech. Syst. Signal Process.*, vol. 21, no. 6, pp. 2616–2633, Aug. 2007, doi: 10.1016/j.ymssp.2006.12.002.
[71] L. Luo, B. Li, Y. Yu, X. Xu, K. Soga, and J. Yan, “Time and Frequency Localized Pulse Shape for Resolution Enhancement in STFT-BOTDR,” *J. Sens.*, vol. 2016, p. e3204130, Jan. 2016, doi: 10.1155/2016/3204130.
[72] E. Sejdić, I. Djurović, and J. Jiang, “Time–frequency feature representation using energy concentration: An overview of recent advances,” *Digit. Signal Process.*, vol. 19, no. 1, pp. 153–183, Jan. 2009, doi: 10.1016/j.dsp.2007.12.004.
[73] H. Jeon, Y. Jung, S. Lee, and Y. Jung, “Area-Efficient Short-Time Fourier Transform Processor for Time–Frequency Analysis of Non-Stationary Signals,” *Appl. Sci.*, vol. 10, p. 7208, Oct. 2020, doi: 10.3390/app10207208.
[74] D. Griffin and J. Lim, “Signal estimation from modified short-time Fourier transform,” *IEEE Trans. Acoust. Speech Signal Process.*, vol. 32, no. 2, pp. 236–243, Apr. 1984, doi: 10.1109/TASSP.1984.1164317.
[75] D. Komorowski and S. Pietraszek, “The Use of Continuous Wavelet Transform Based on the Fast Fourier Transform in the Analysis of Multi-channel Electrogastrography Recordings,” *J. Med. Syst.*, vol. 40, no. 1, p. 10, Oct. 2015, doi: 10.1007/s10916-015-0358-4.
[76] S. Mark J., “The discrete wavelet transform: wedding the a trous and Mallat algorithms.,” *IEEE Trans. Signal Process. 4010*, pp. 2464–2482, 1992.
[77] L. Chun-lin, “A Tutorial of the Wavelet Transform,” NTUEE Taiwan 21, Feb. 2010.
[78] Z. ZiLong, H. Lv, J. Xu, H. Zizhao, and W. Qin, “A Deep Learning Method for Bearing Fault Diagnosis through Stacked Residual Dilated Convolutions,” *Appl. Sci.*, vol. 9, p. 1823, May 2019, doi: 10.3390/app9091823.
[79] M. Yani, S. Irawan, and C. Setianingsih, “Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry’s Nail,” *J. Phys. Conf. Ser.*, vol. 1201, p. 012052, May 2019, doi: 10.1088/1742-6596/1201/1/012052.
[80] P. Newbold, “ARIMA model building and the time series analysis approach to forecasting,” *J. Forecast.*, vol. 2, no. 1, pp. 23–35, 1983, doi: 10.1002/for.3980020104.

[81] "Vibration Trainning," MOBIUS INSTITUTE, 2020. [Online]. Available: www.MobiusInstitute.com

[82] L. B. Cosme, W. M. Caminhas, M. F. S. V. D'Angelo, and R. M. Palhares, "A Novel Fault-Prognostic Approach Based on Interacting Multiple Model Filters and Fuzzy Systems," *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 519–528, Jan. 2019, doi: 10.1109/TIE.2018.2826449.

[83] R. K. Jha and P. D. Swami, "Failure prognosis of rolling bearings using maximum variance wavelet subband selection and support vector regression," *J. Braz. Soc. Mech. Sci. Eng.*, vol. 44, no. 1, p. 49, Jan. 2022, doi: 10.1007/s40430-021-03345-2.

[84] S. Hong, Z. Zhou, E. Zio, and K. Hong, "Condition assessment for the performance degradation of bearing based on a combinatorial feature extraction method," *Digit. Signal Process.*, vol. 27, pp. 159–166, Apr. 2014, doi: 10.1016/j.dsp.2013.12.010.

Appendix A - Diagnosis Results

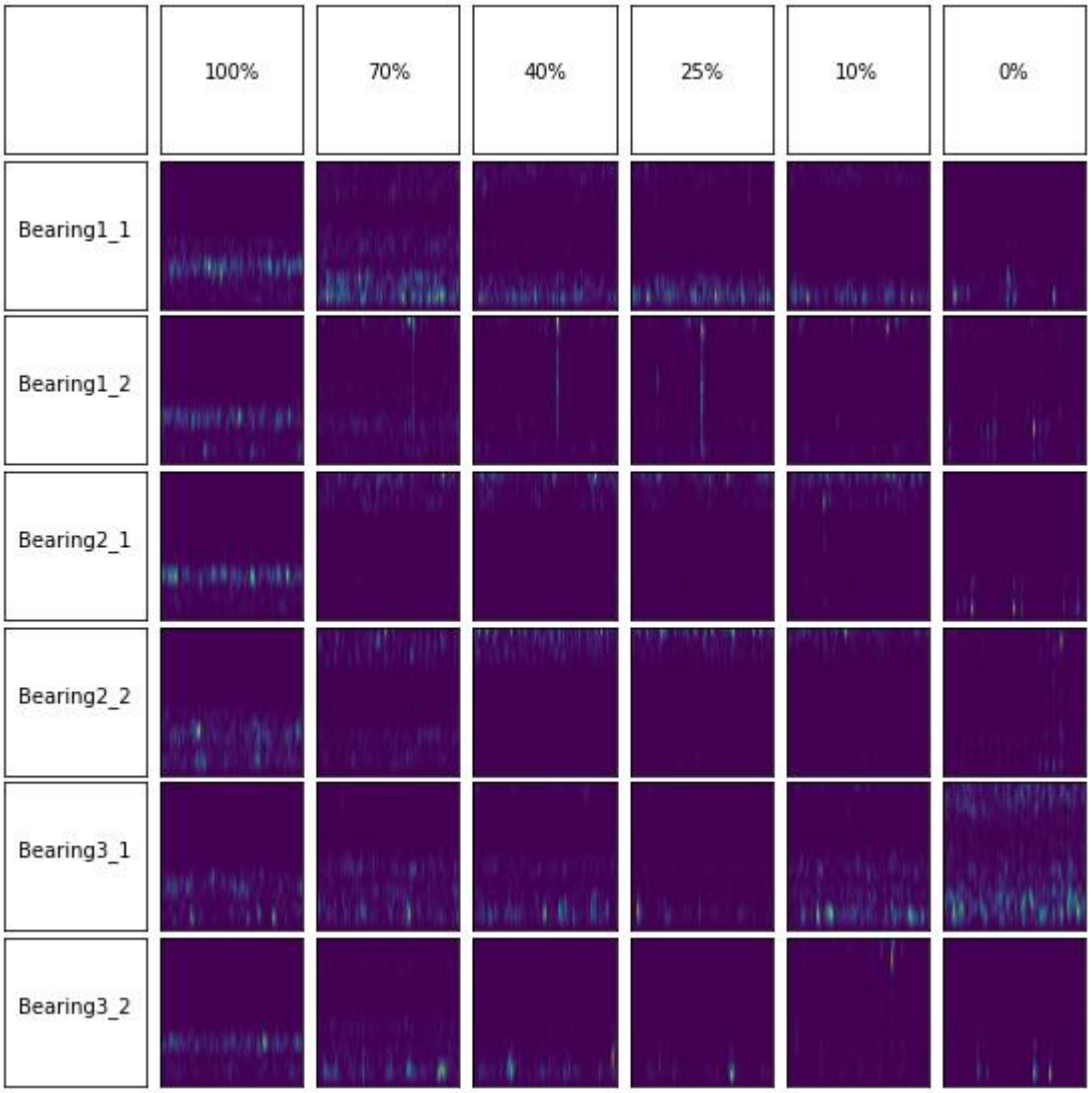


Figure A - 1 Spectrograms for train bearings for Channel 1 data

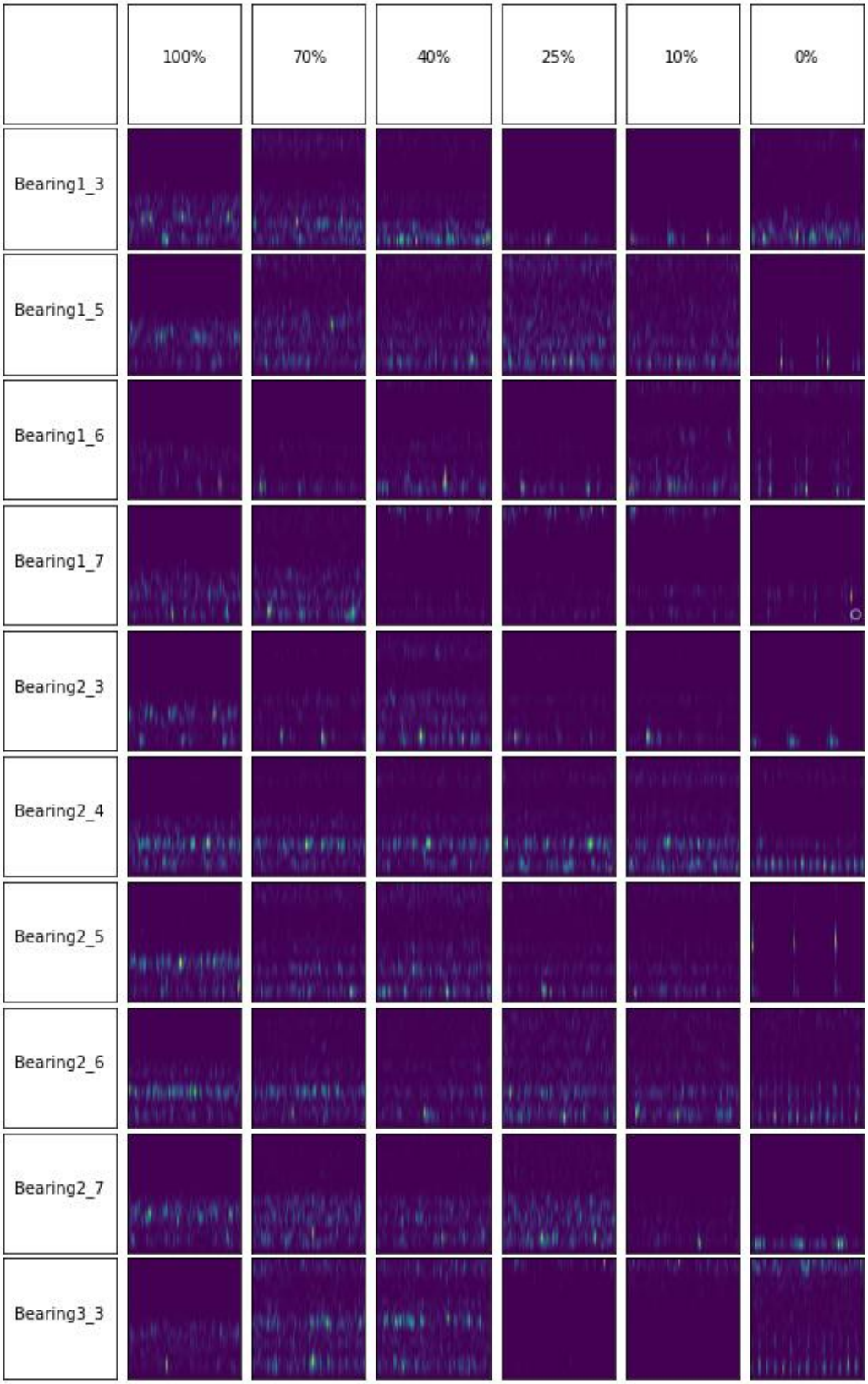


Figure A - 2 Spectrograms for test bearings for Channel 1 data

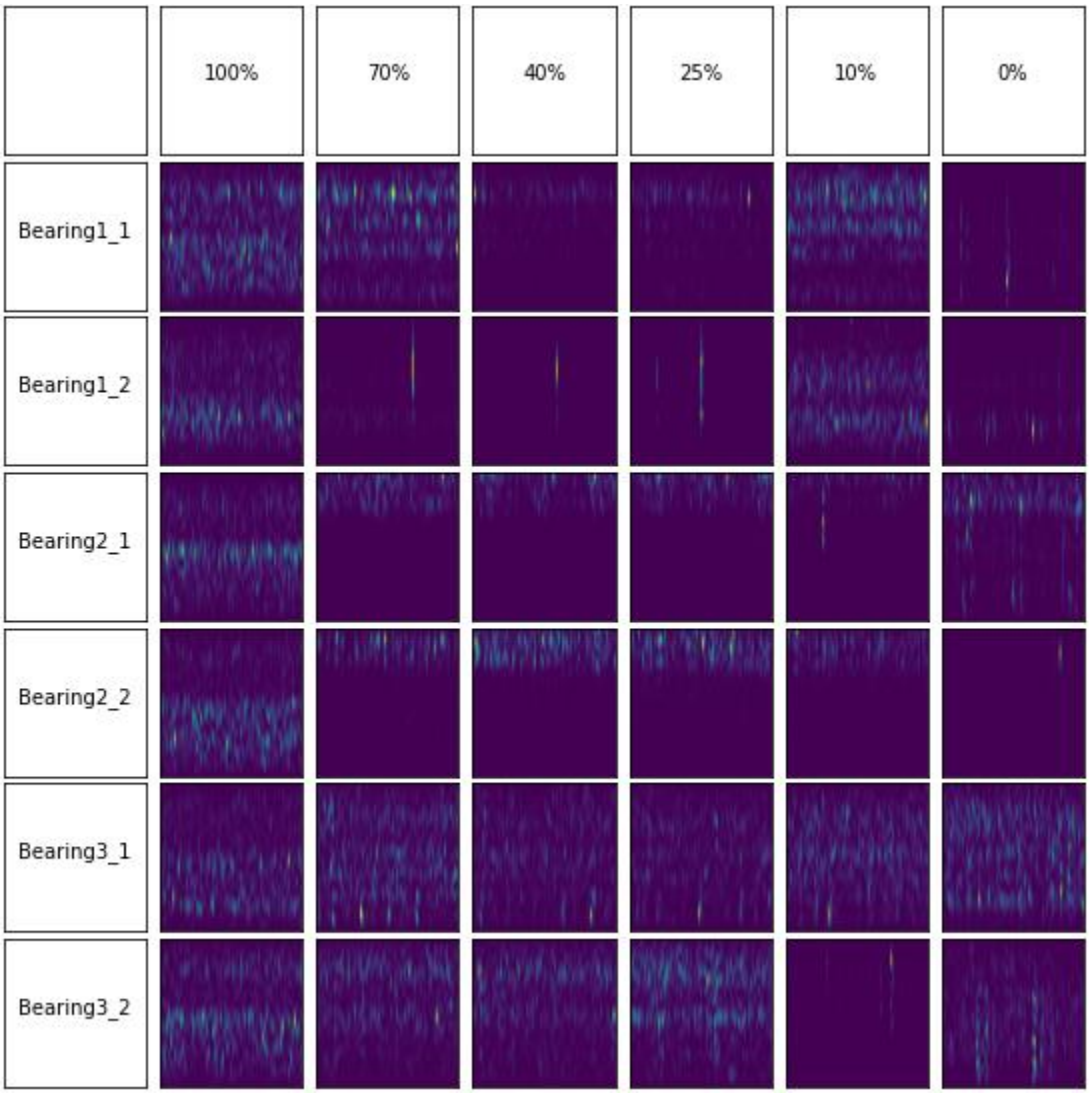


Figure A - 3 Spectrograms for train bearings for Channel 2 data

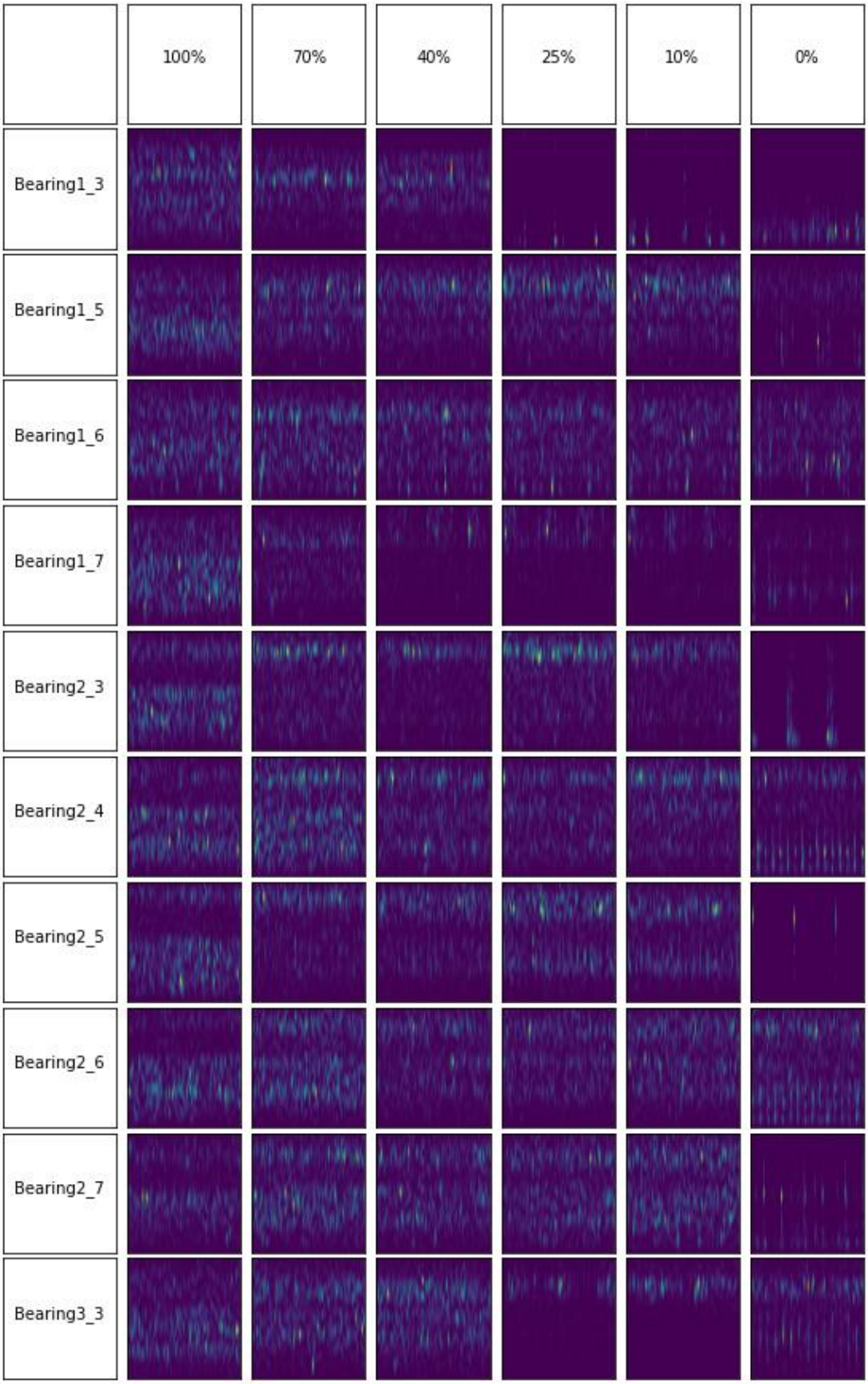


Figure A - 4 Spectrograms for test bearings for Channel 2 data

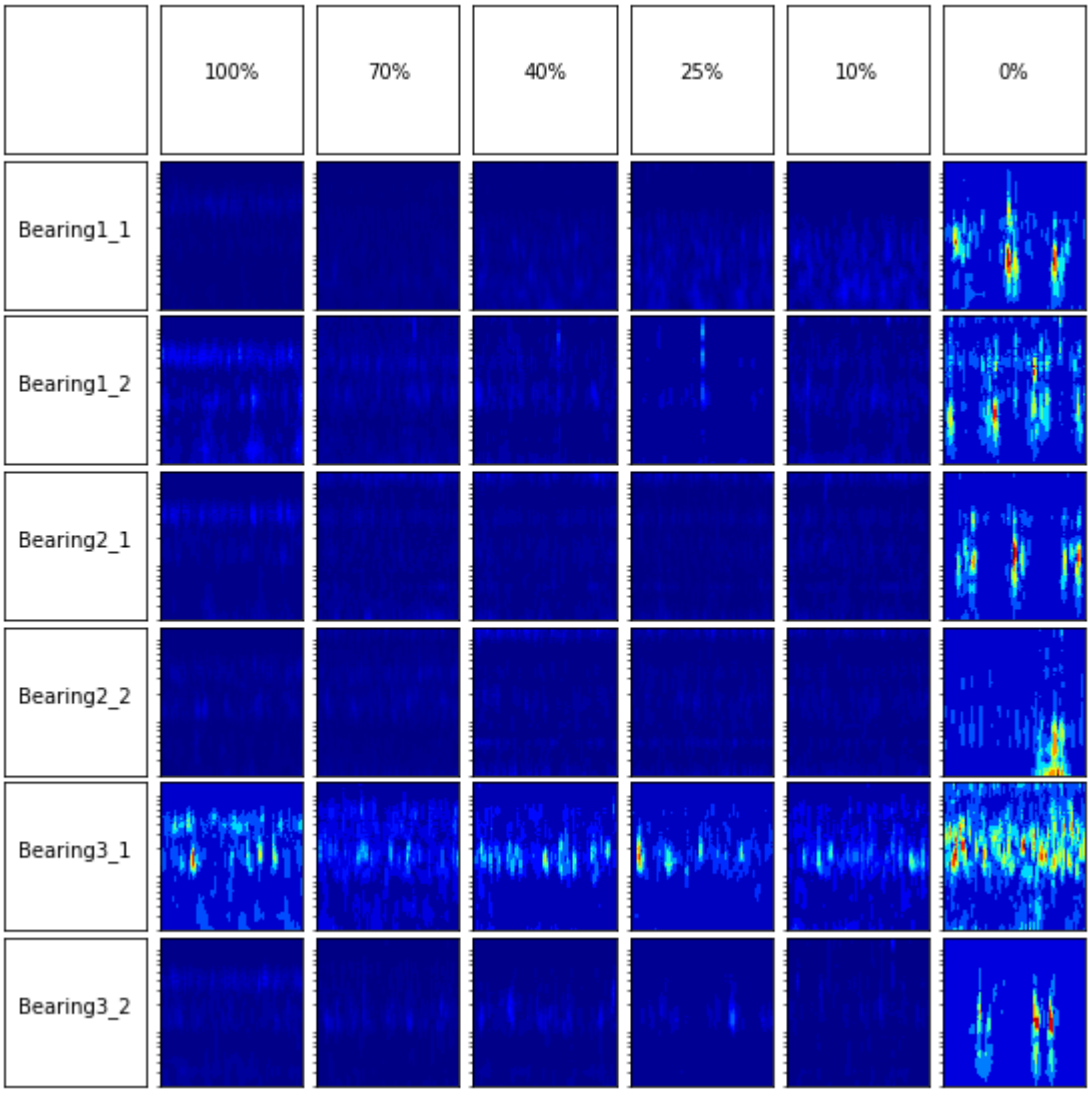


Figure A - 5 Scalograms for train bearings for Channel 1 data

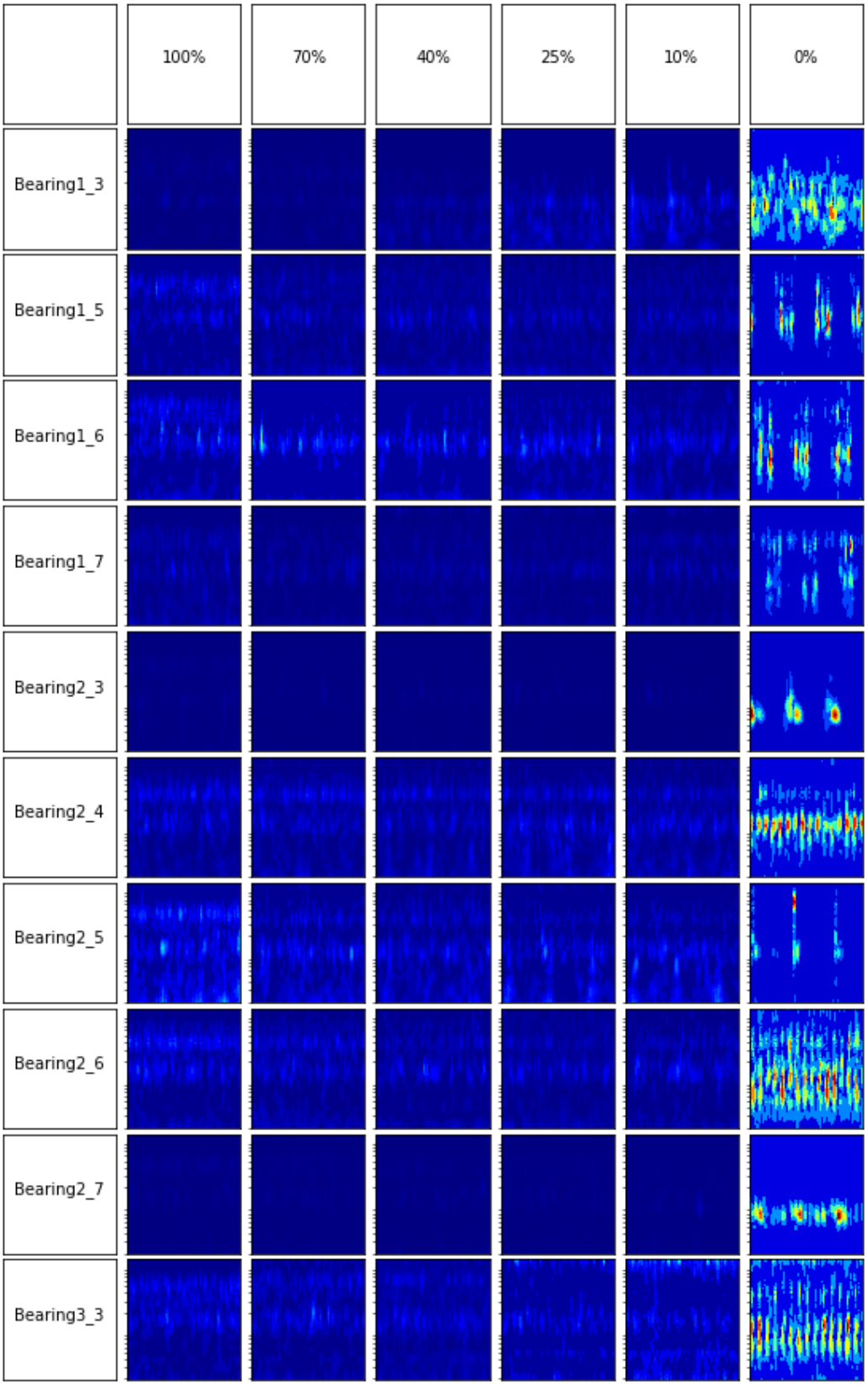


Figure A - 6 Scalograms for test bearings for Channel 1 data

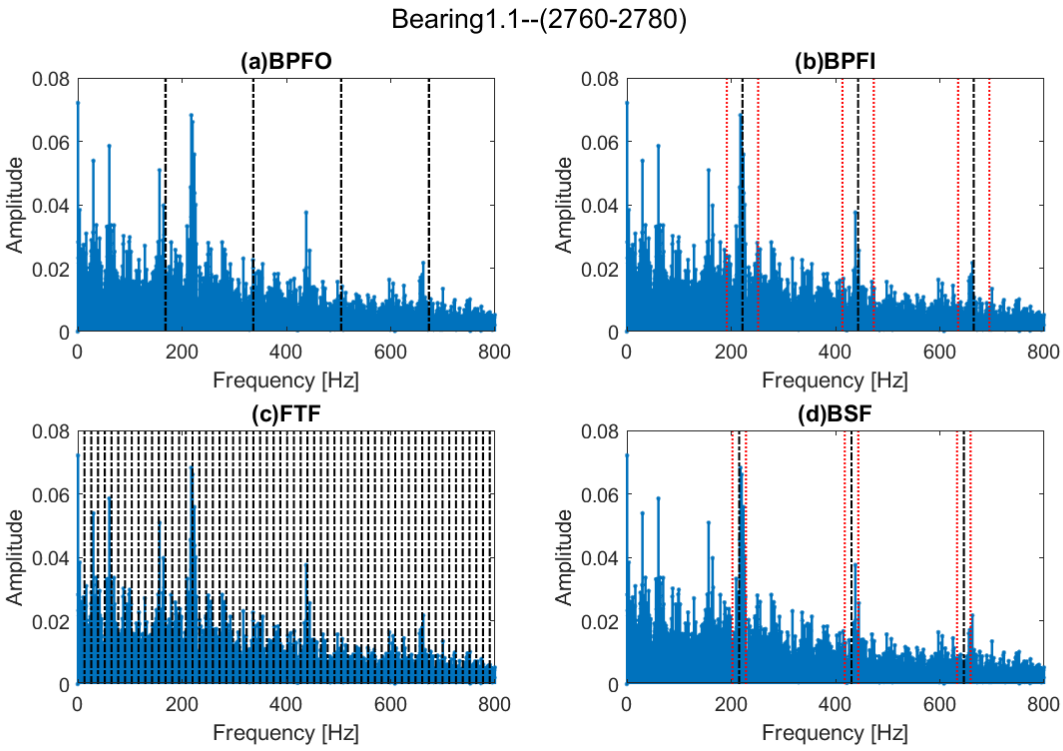


Figure A - 7 Bearing1_1 channel 1 diagnosis result

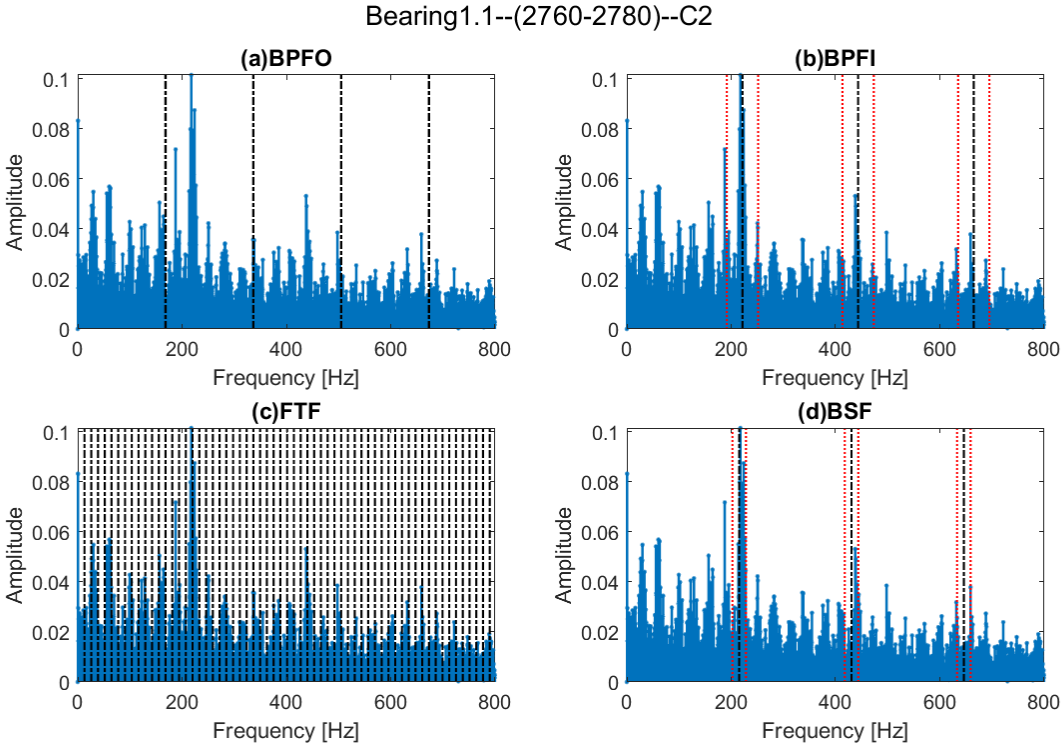


Figure A - 8 Bearing1_1 channel 2 diagnosis result

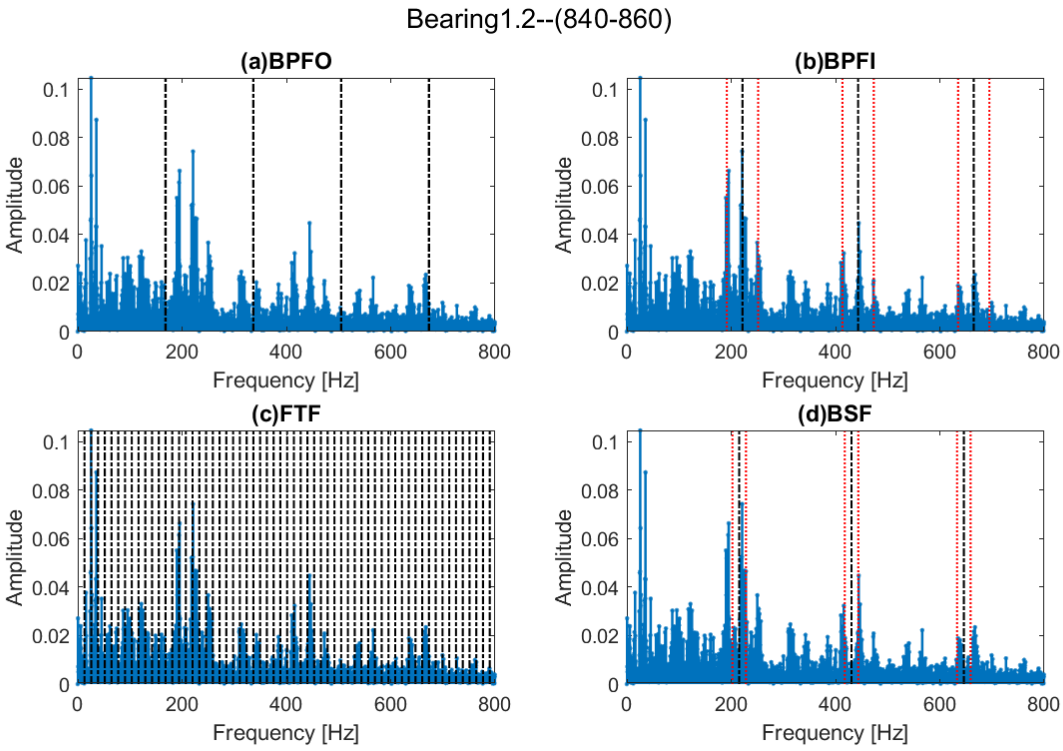


Figure A - 9 Bearing1_2 channel 1 diagnosis result

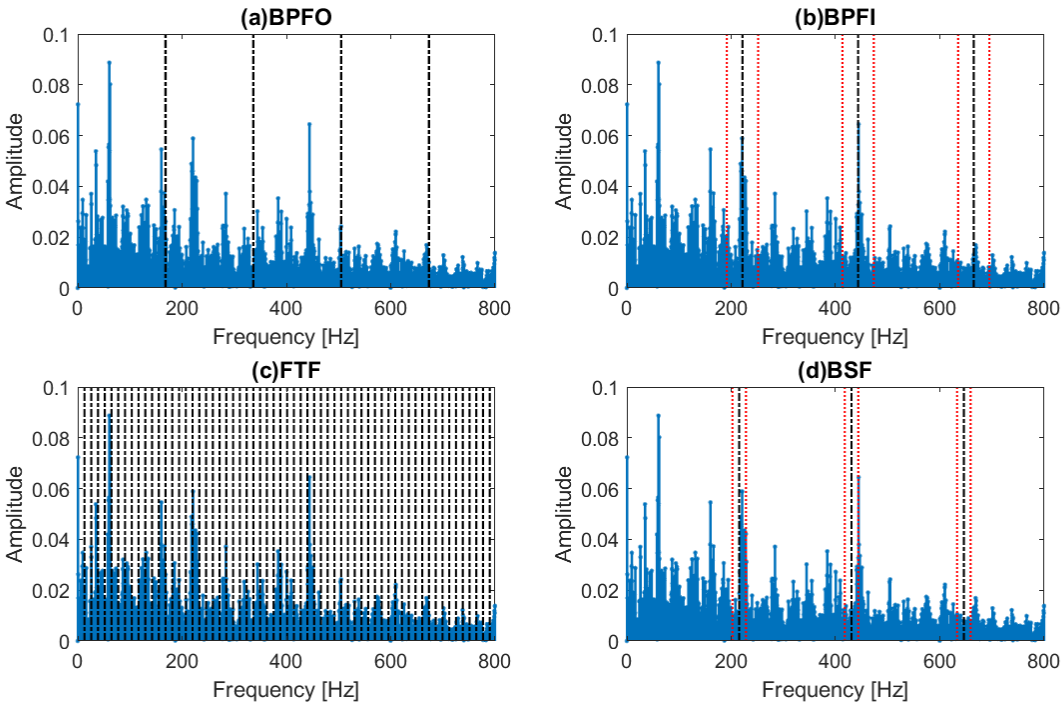


Figure A - 10 Bearing1_2 channel 2 diagnosis result

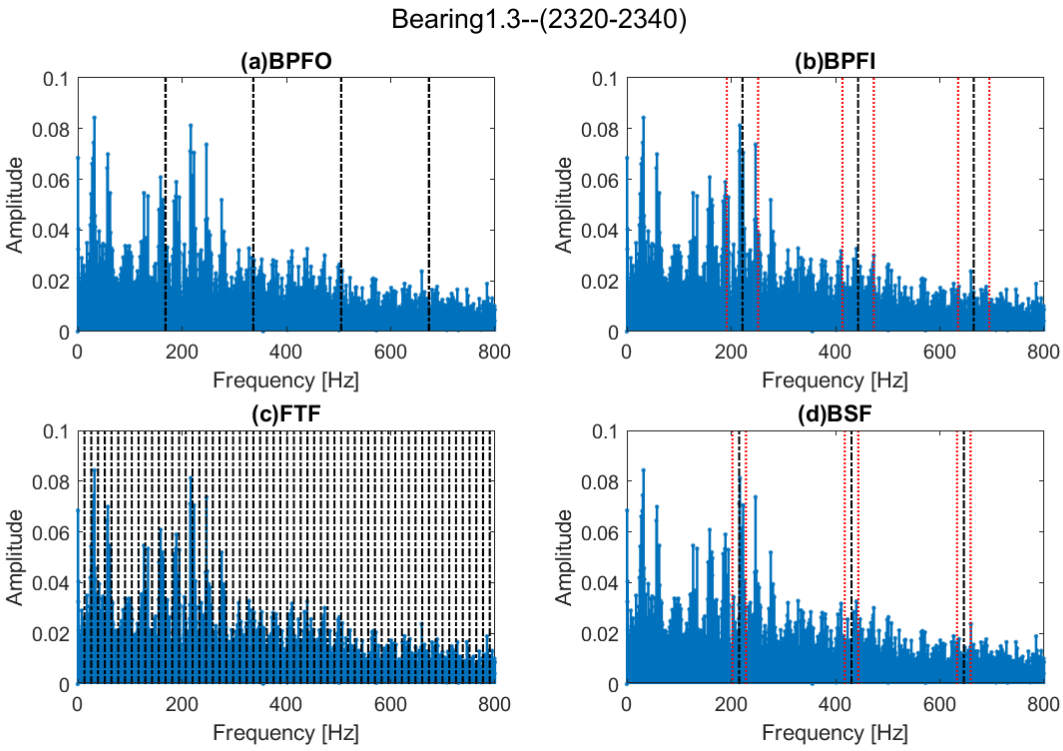


Figure A - 11 Bearing1_3 channel 1 diagnosis result

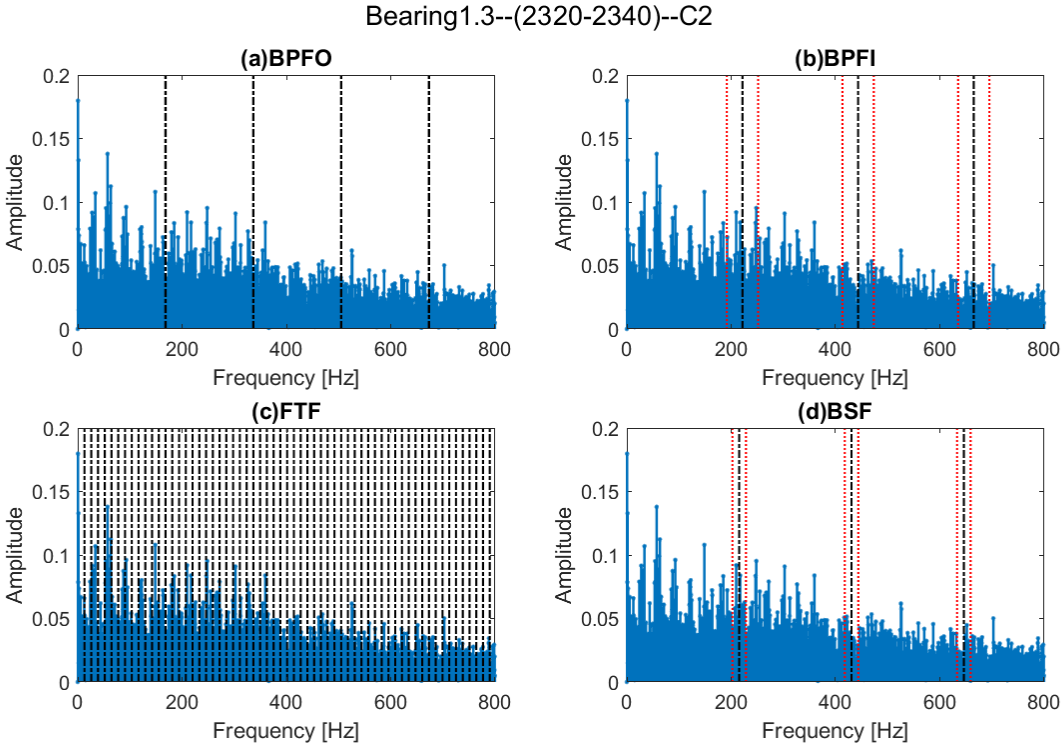


Figure A - 12 Bearing1_3 channel 2 diagnosis result

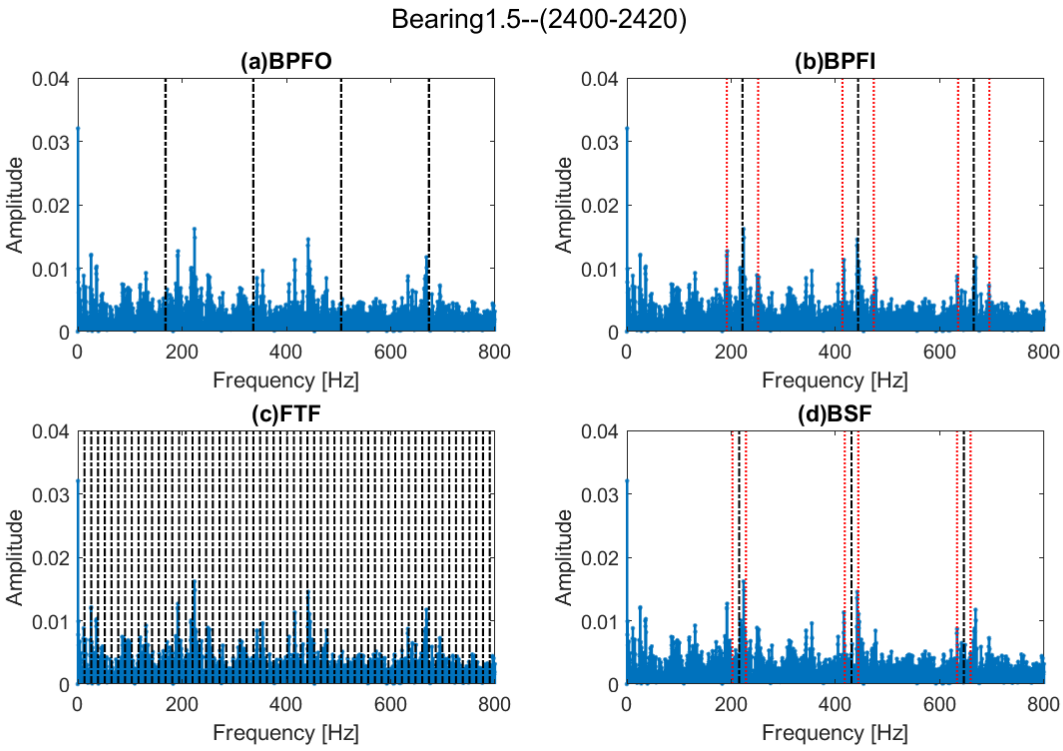


Figure A - 13 Bearing1_5 channel 1 diagnosis result

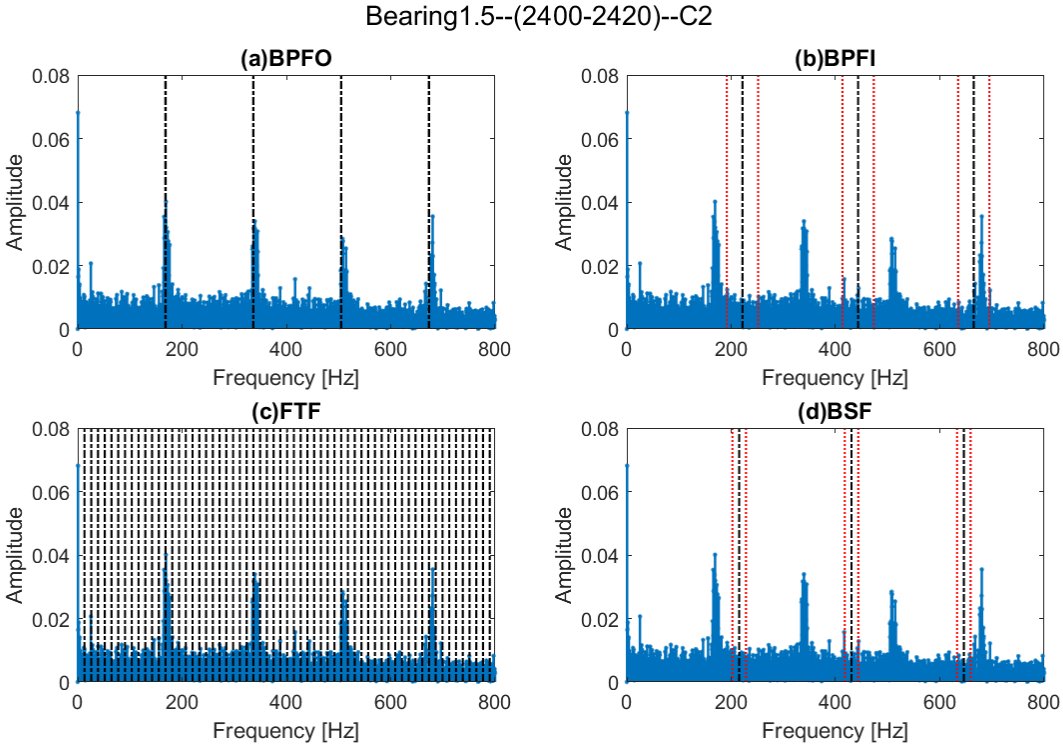


Figure A - 14 Bearing1_5 channel 2 diagnosis result

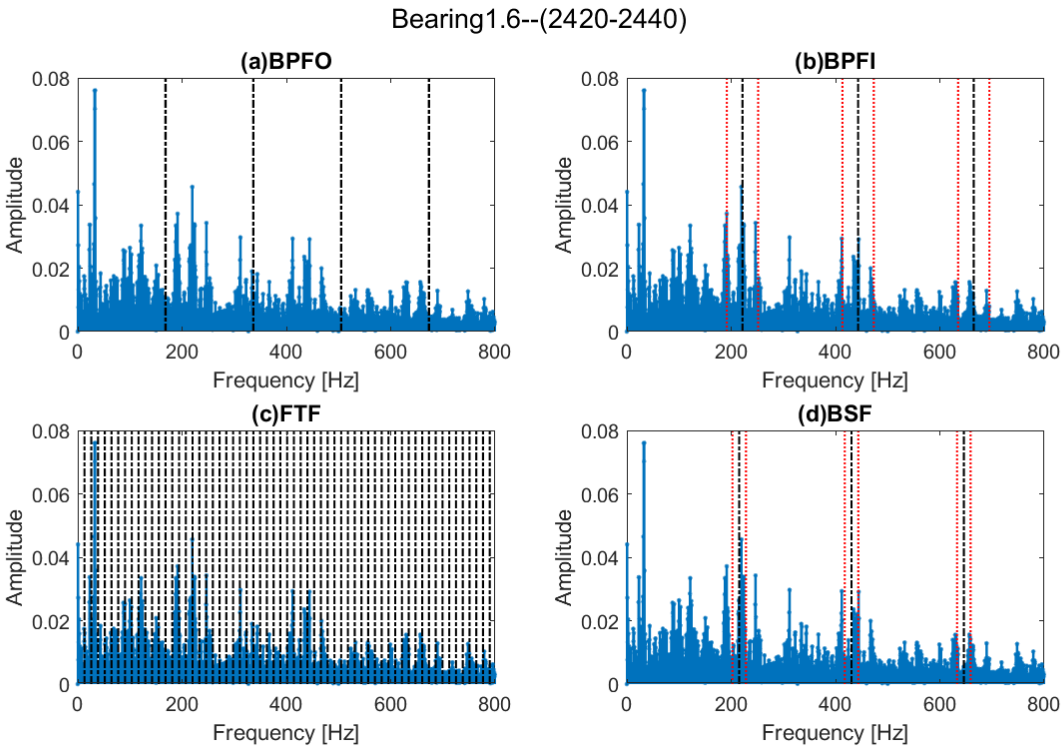


Figure A - 15 Bearing1_6 channel 1 diagnosis result

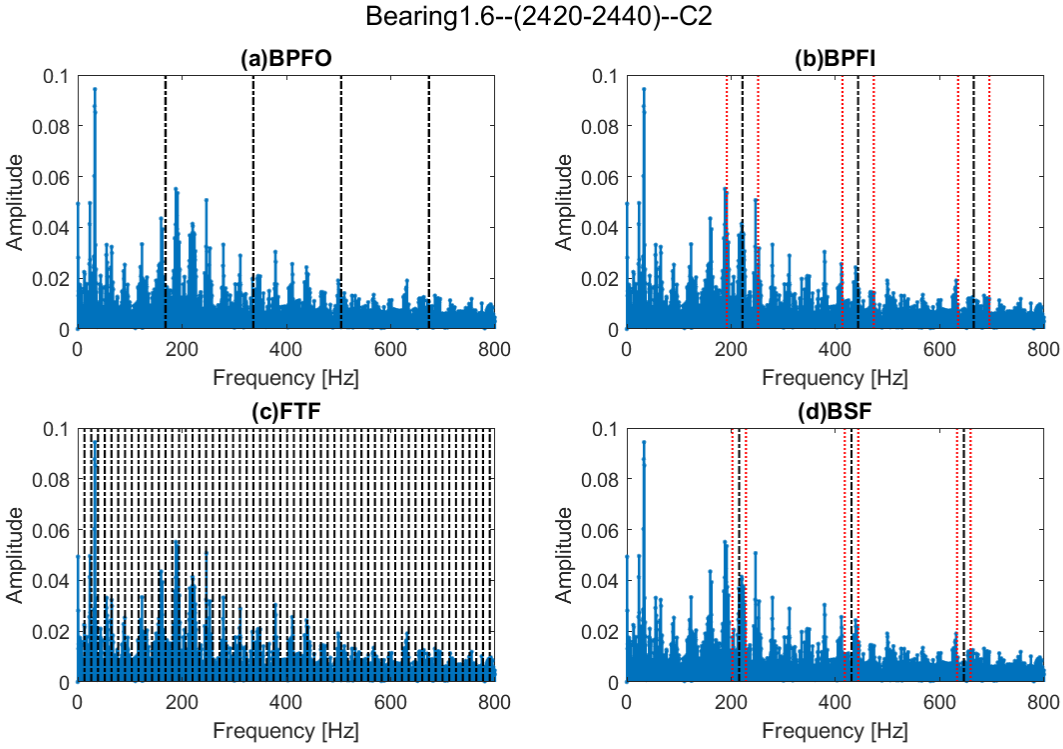


Figure A - 16 Bearing1_6 channel 2 diagnosis result

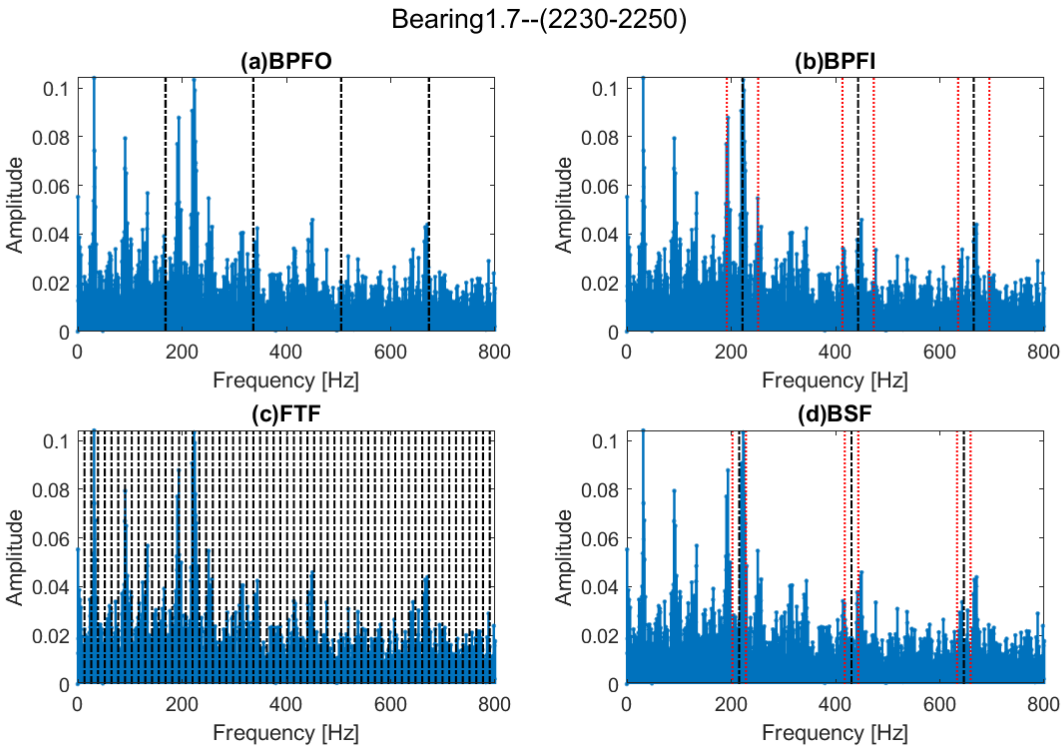


Figure A - 17 Bearing1_7 channel 1 diagnosis result

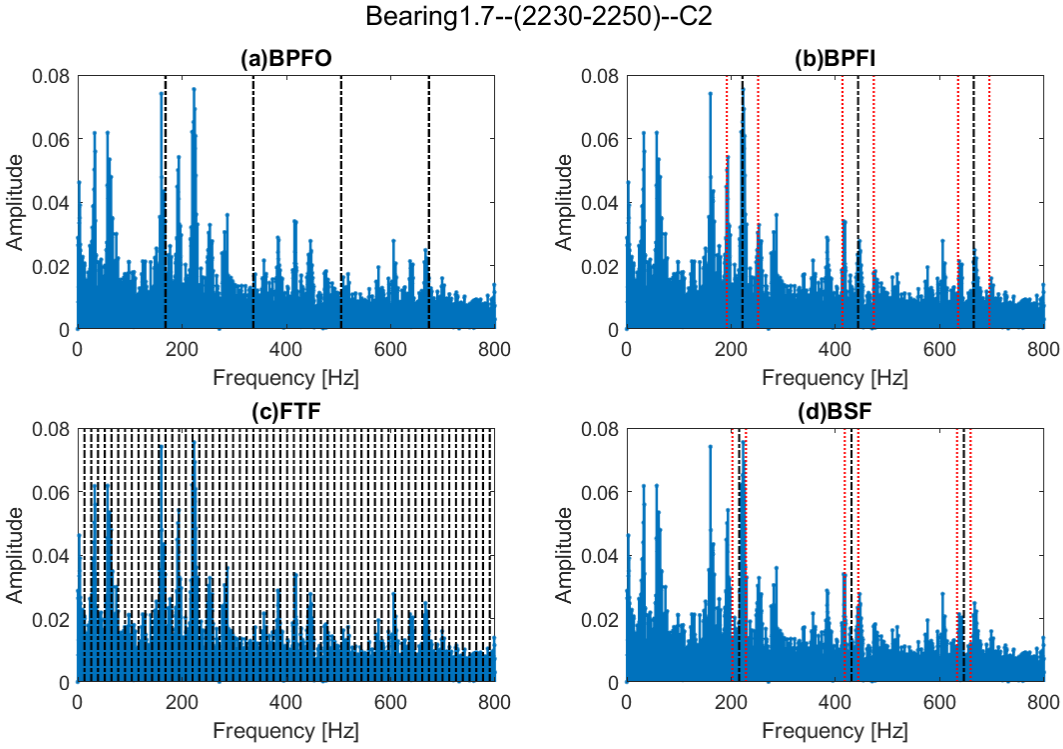


Figure A - 18 Bearing1_7 channel 2 diagnosis result

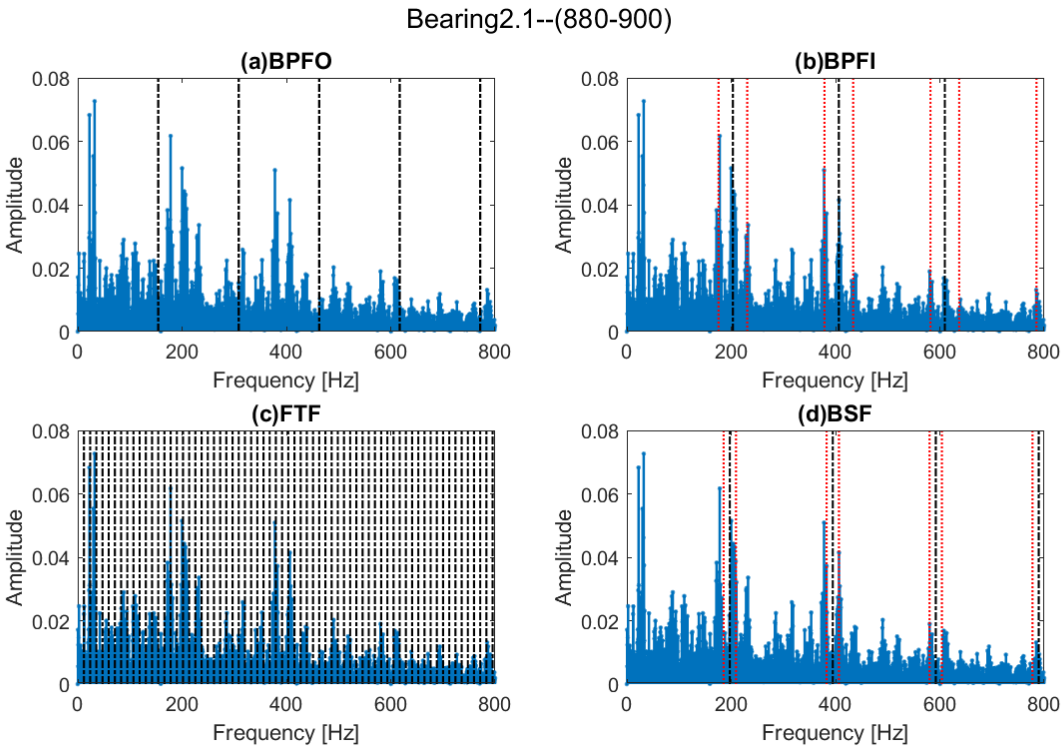


Figure A - 19 Bearing2_1 channel 1 diagnosis result

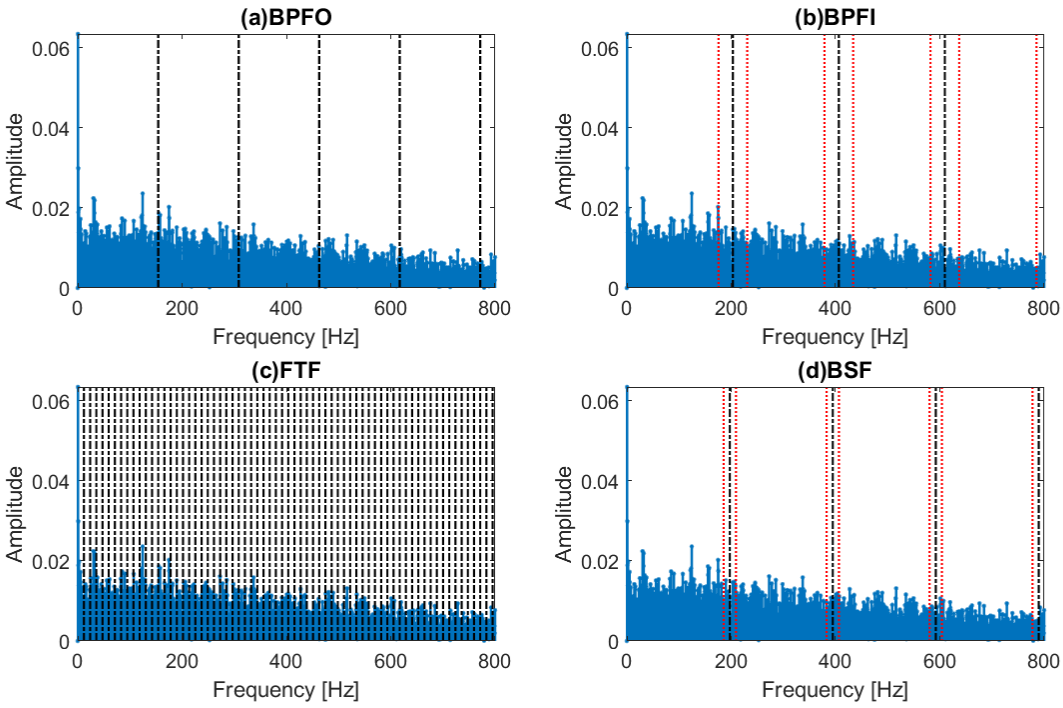


Figure A - 20 Bearing2_1 channel 2 diagnosis result

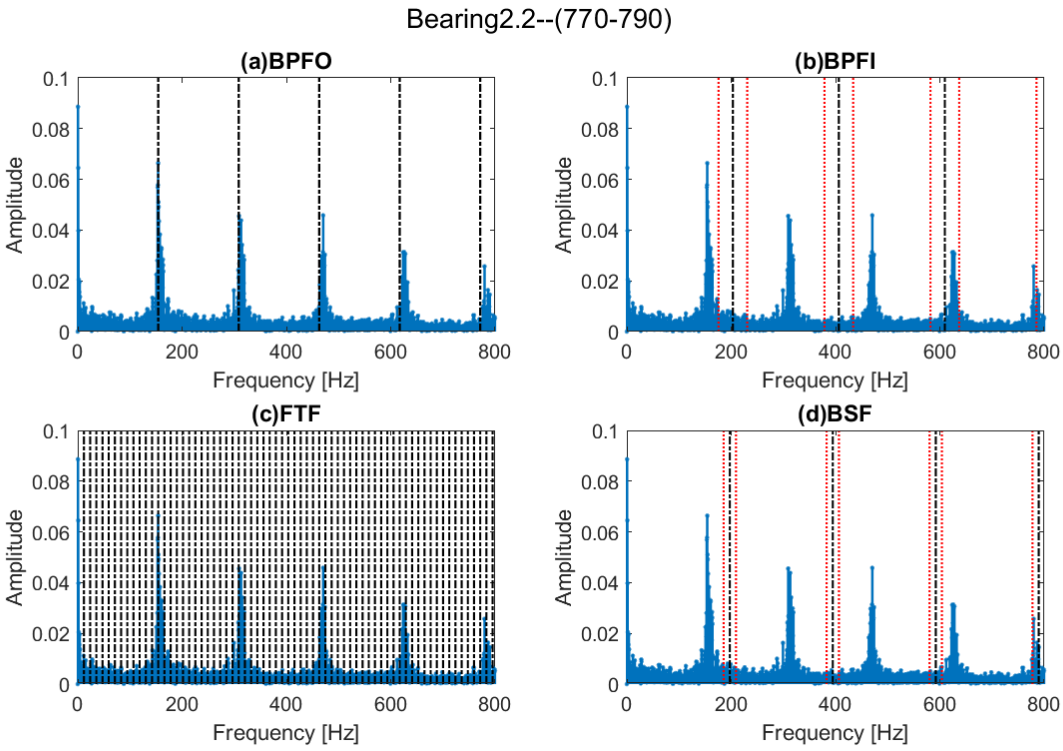


Figure A - 21 Bearing2_2 channel 1 diagnosis result

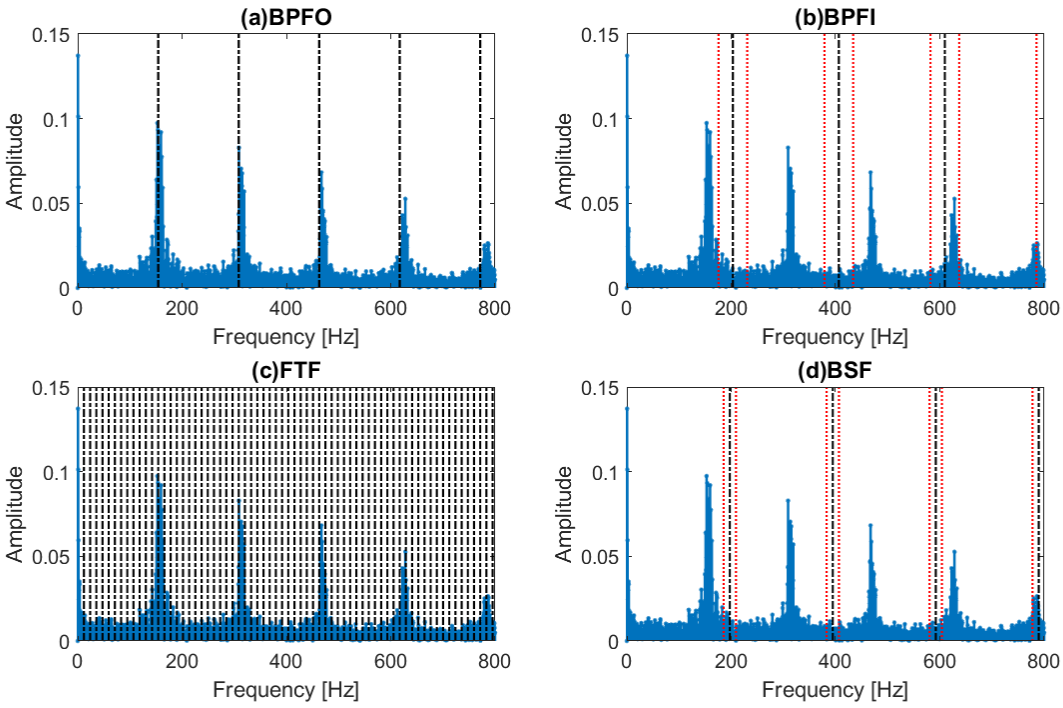


Figure A - 22 Bearing2_2 channel 2 diagnosis result

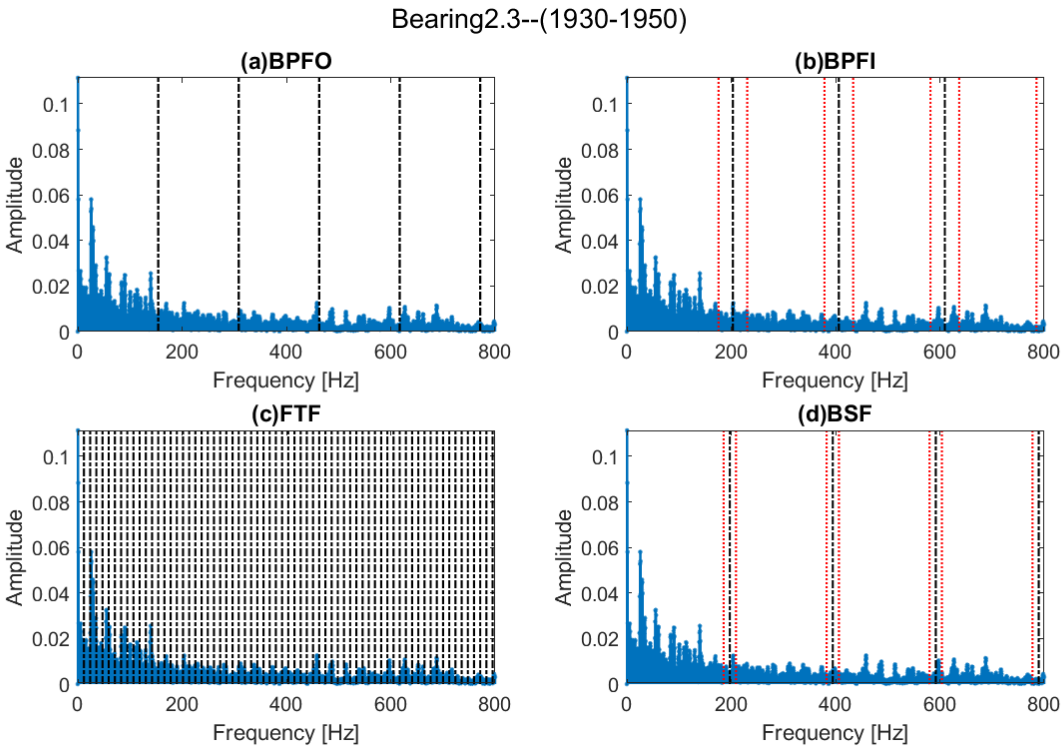


Figure A - 23 Bearing2_3 channel 1 diagnosis result

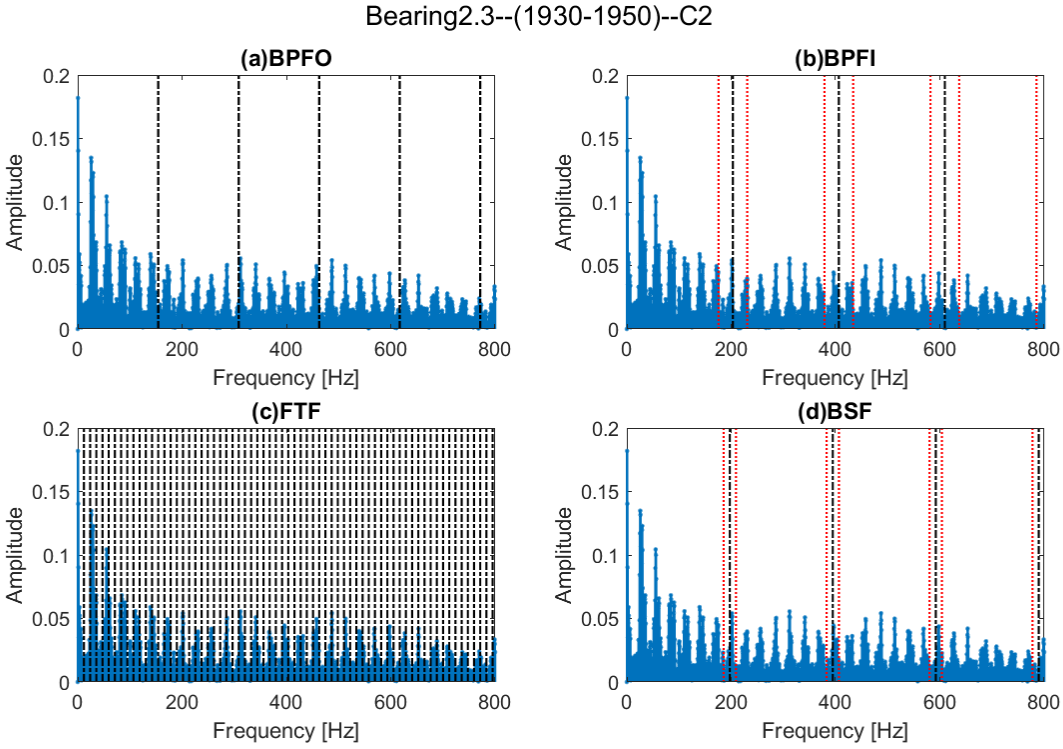


Figure A - 24 Bearing2_3 channel 2 diagnosis result

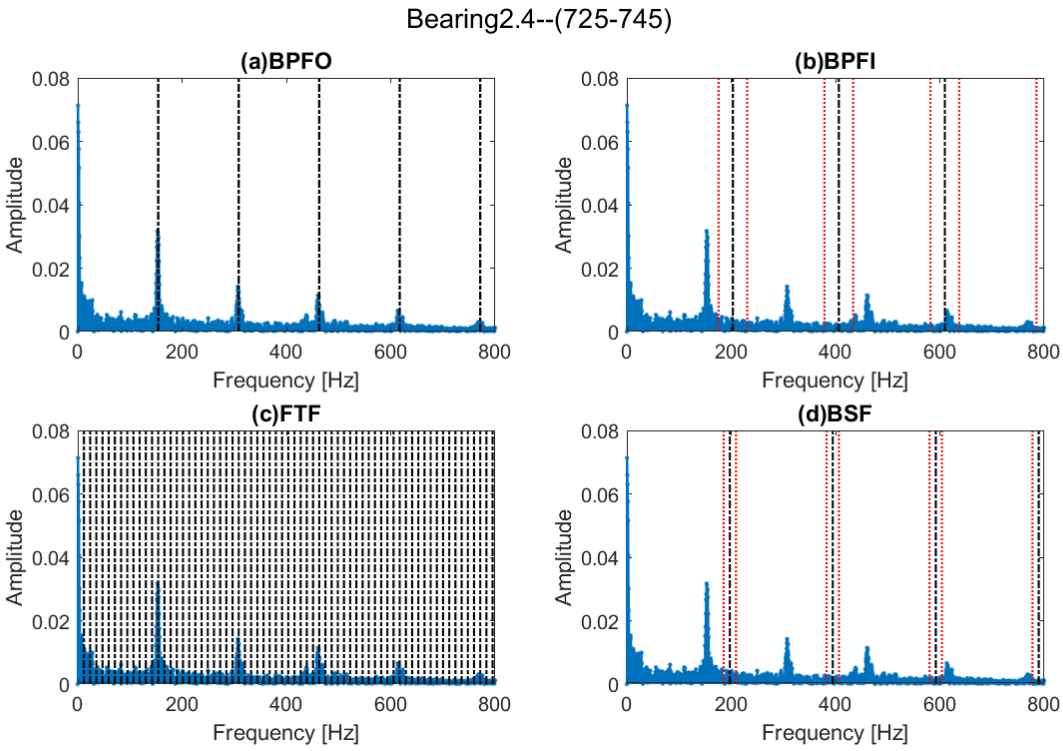


Figure A - 25 Bearing2_4 channel 1 diagnosis result

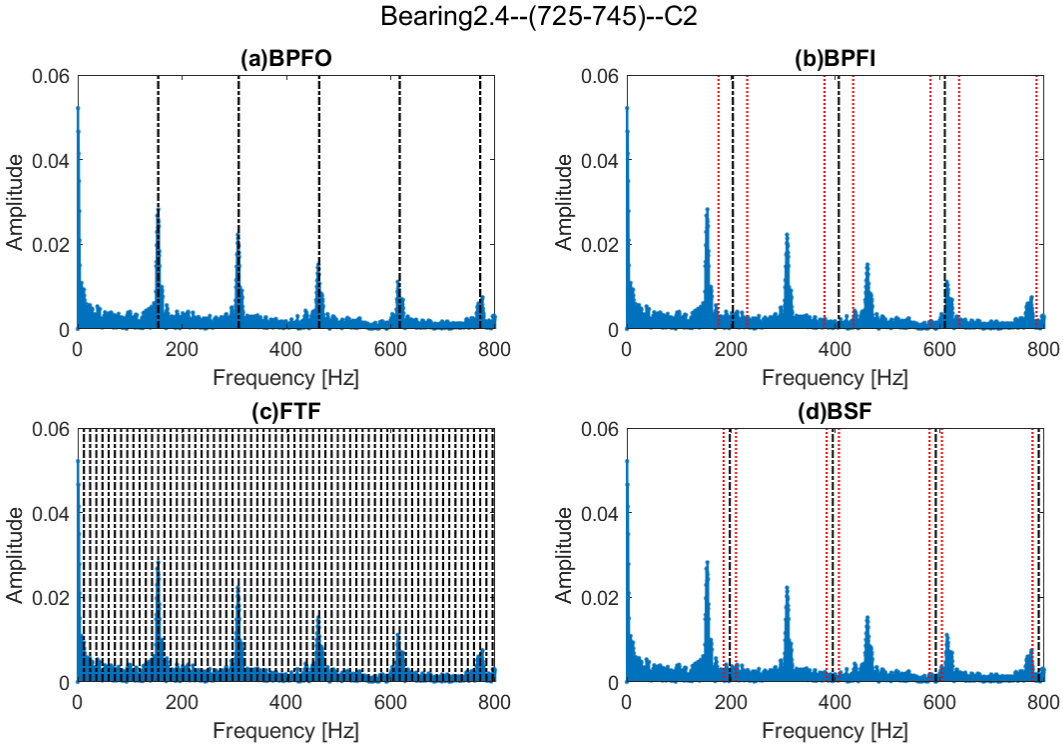


Figure A - 26 Bearing2_4 channel 2 diagnosis result

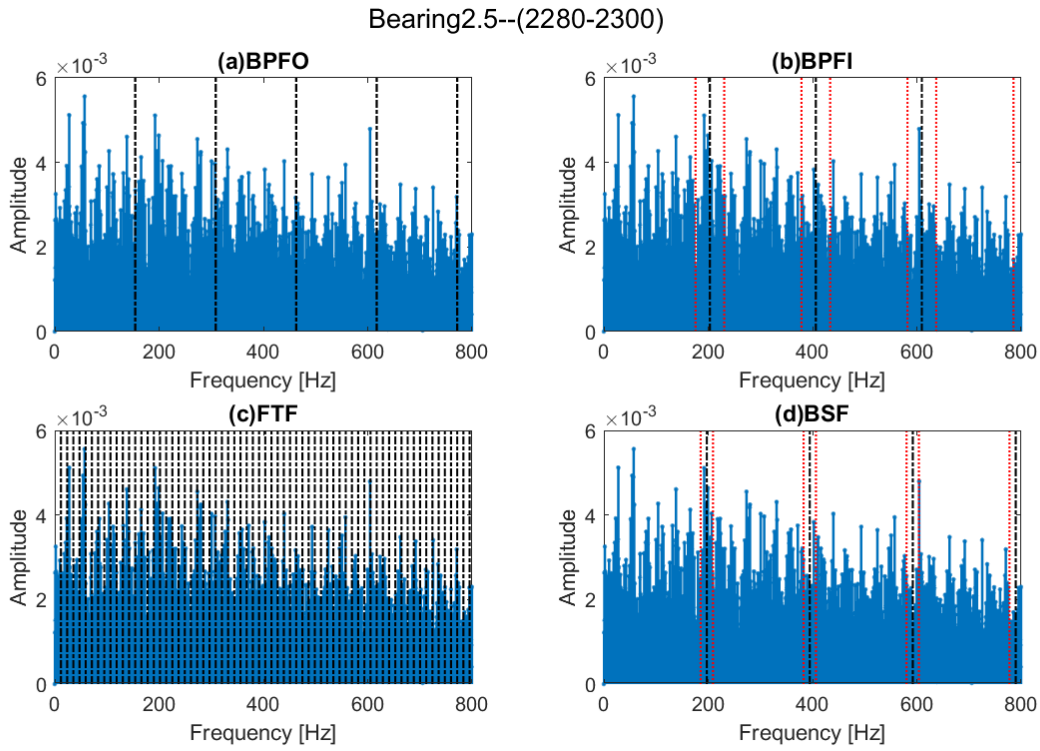


Figure A - 27 Bearing2_5 channel 1 diagnosis result

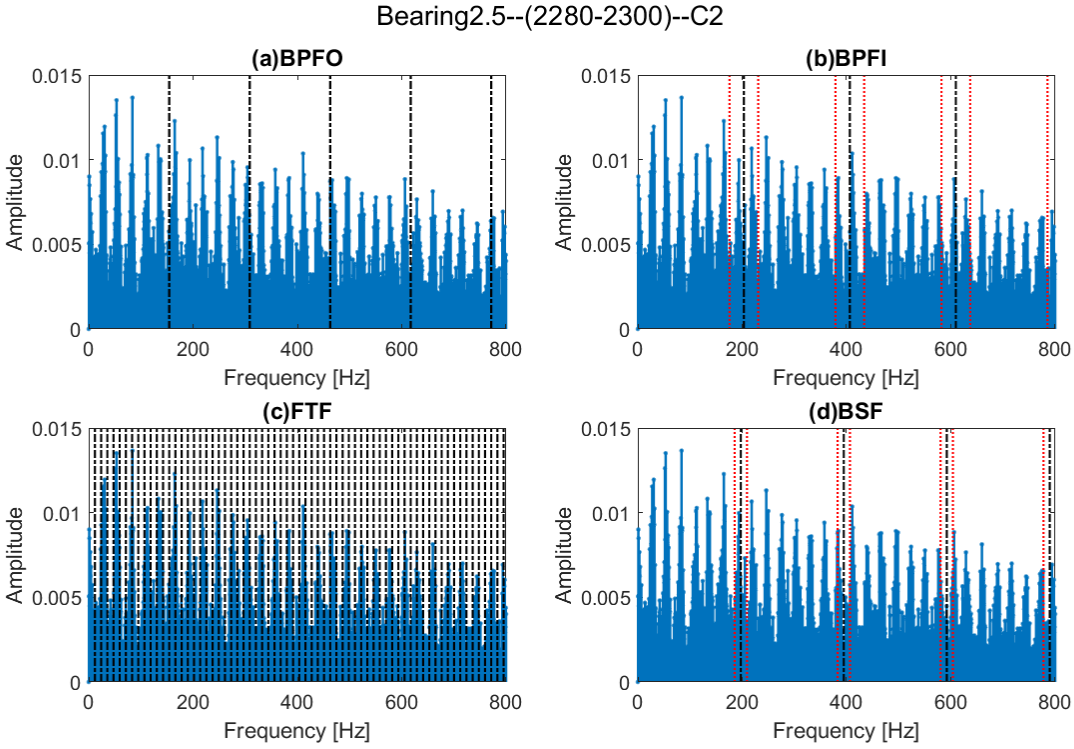


Figure A - 28 Bearing2_5 channel 2 diagnosis result

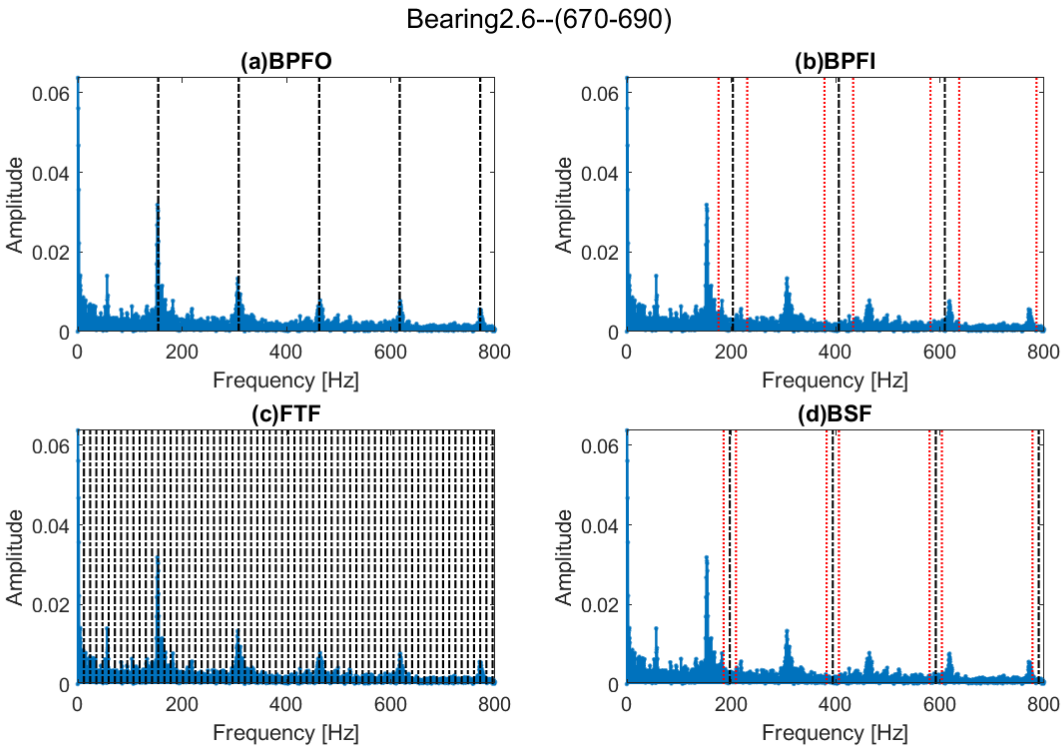


Figure A - 29 Bearing2_6 channel 1 diagnosis result

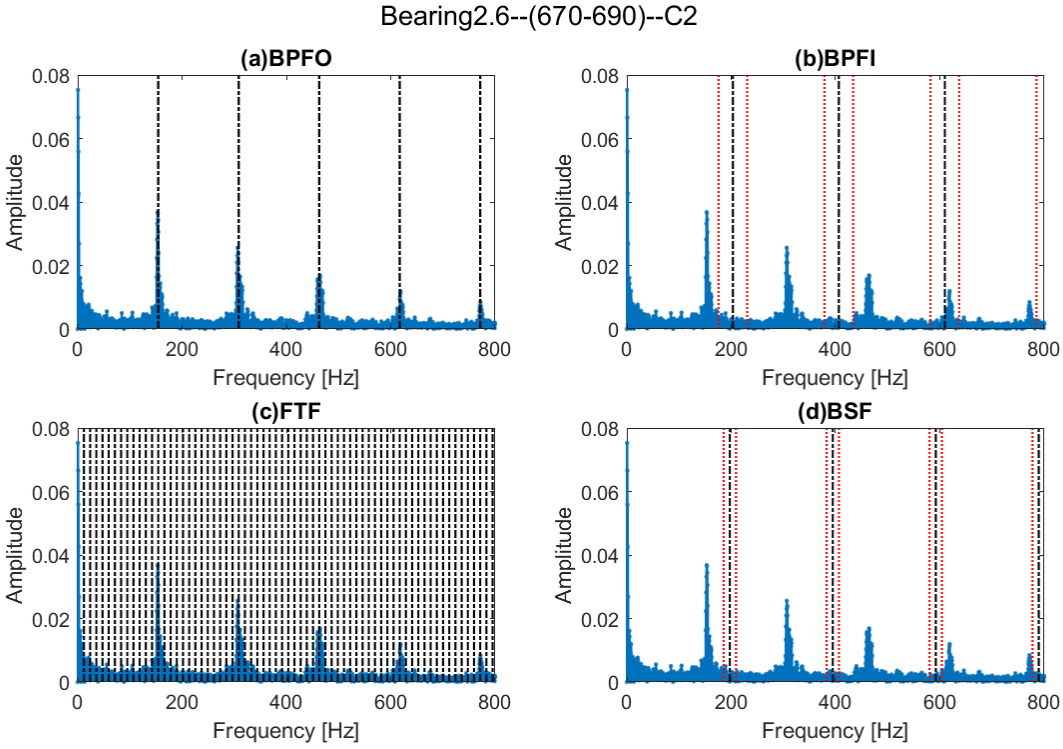


Figure A - 30 Bearing2_6 channel 2 diagnosis result

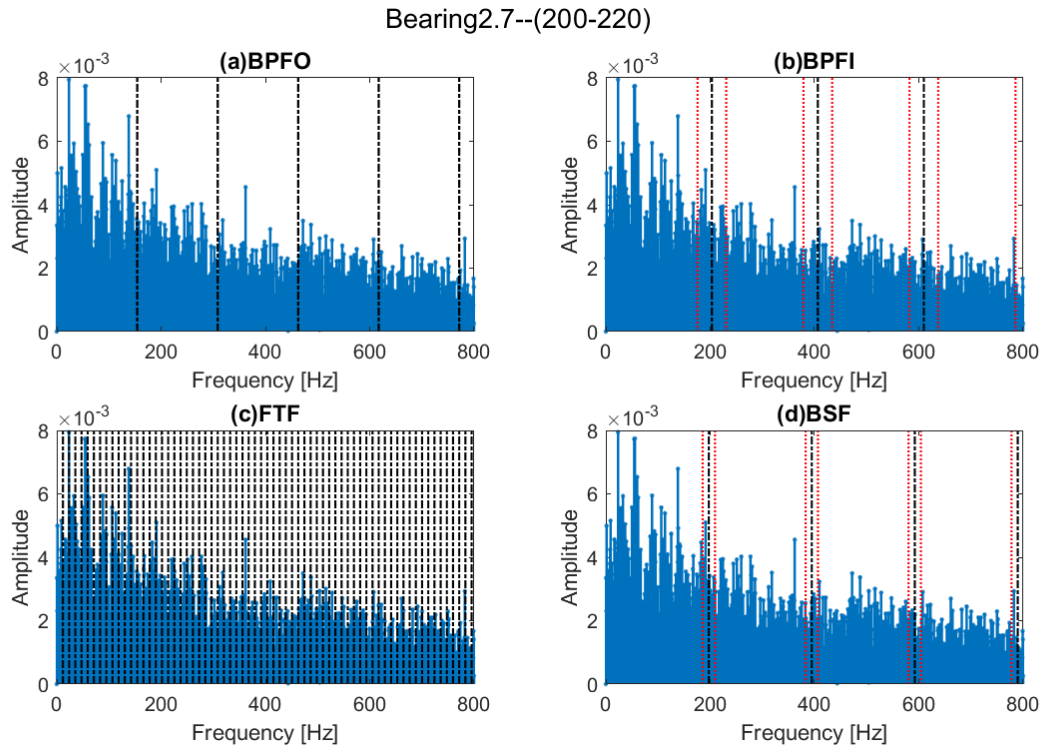


Figure A - 31 Bearing2_7 channel 1 diagnosis result

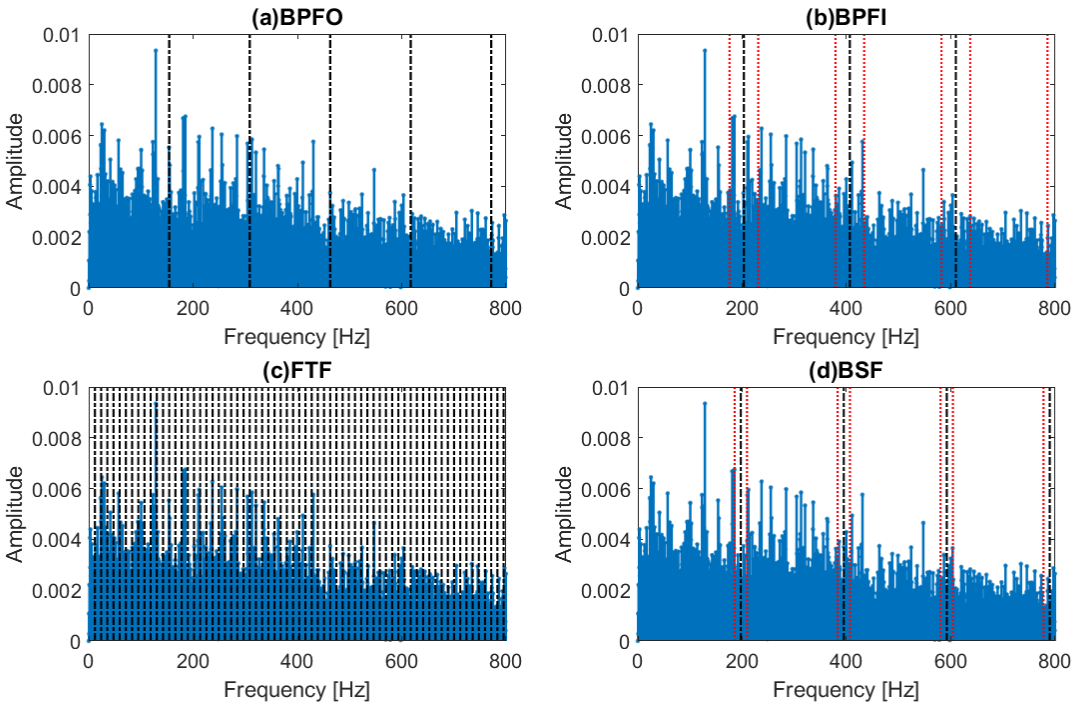


Figure A - 32 Bearing2_7 channel 2 diagnosis result

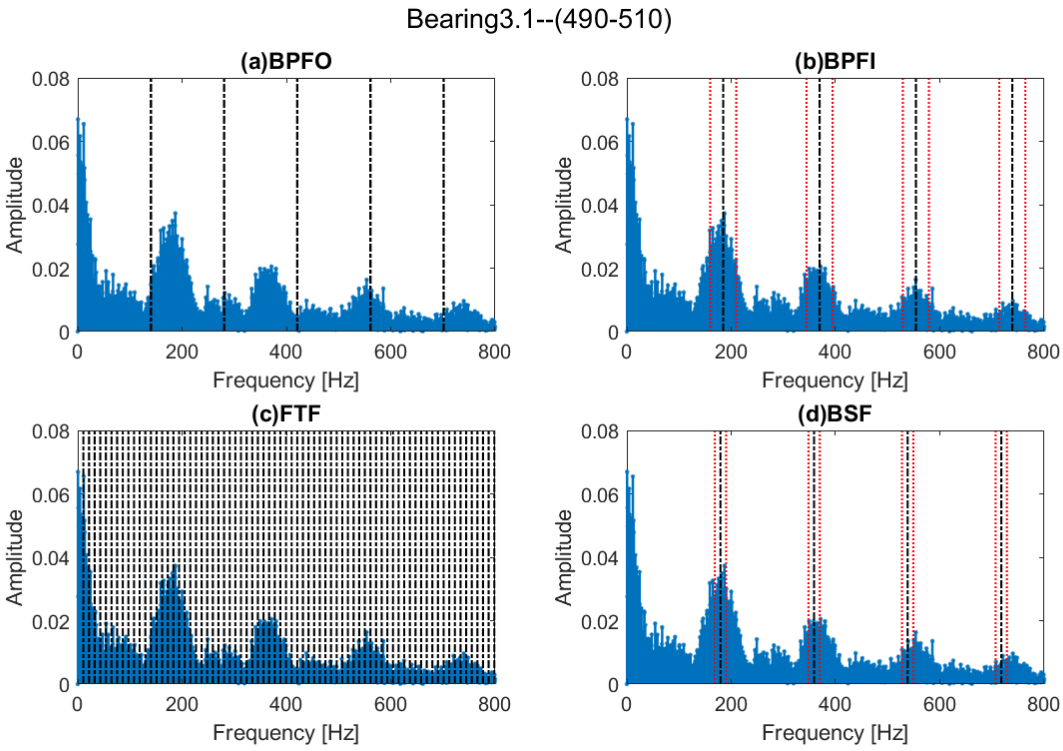


Figure A - 33 Bearing3_1 channel 1 diagnosis result

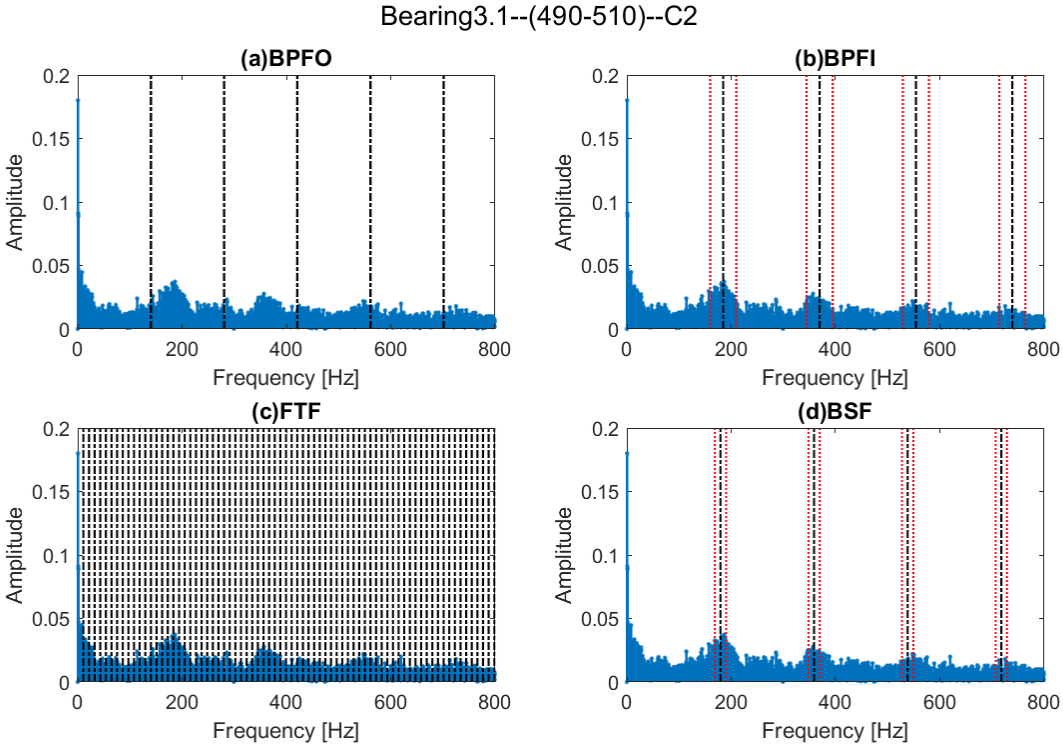


Figure A - 34 Bearing3_1 channel 2 diagnosis result

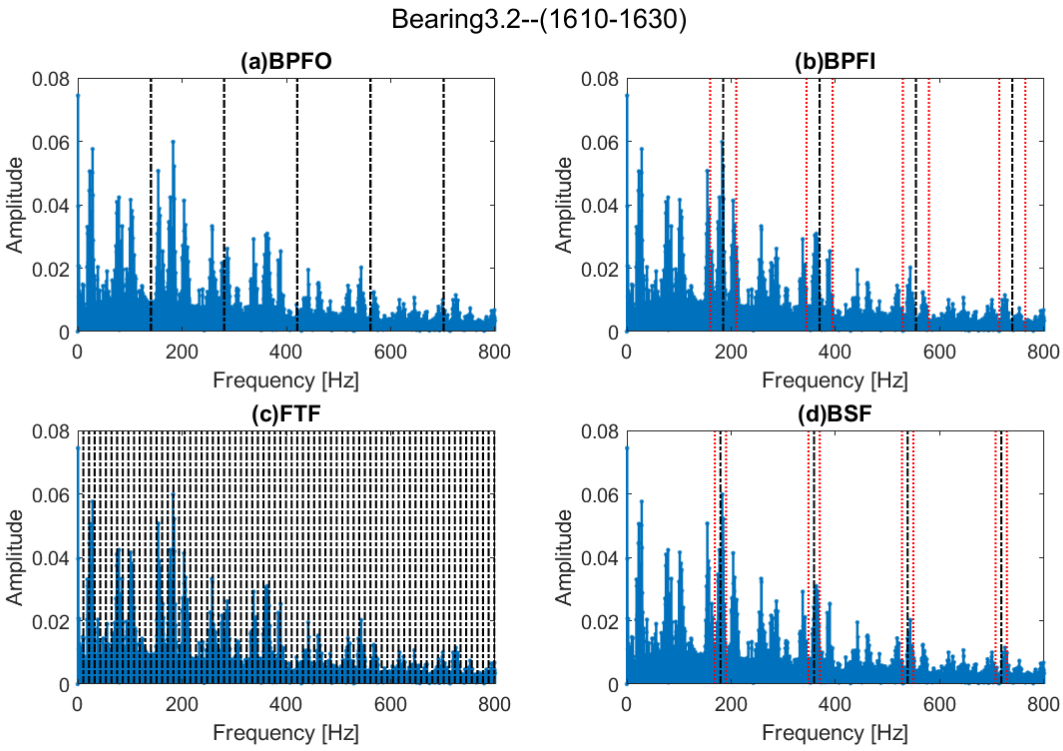


Figure A - 35 Bearing3_2 channel 1 diagnosis result

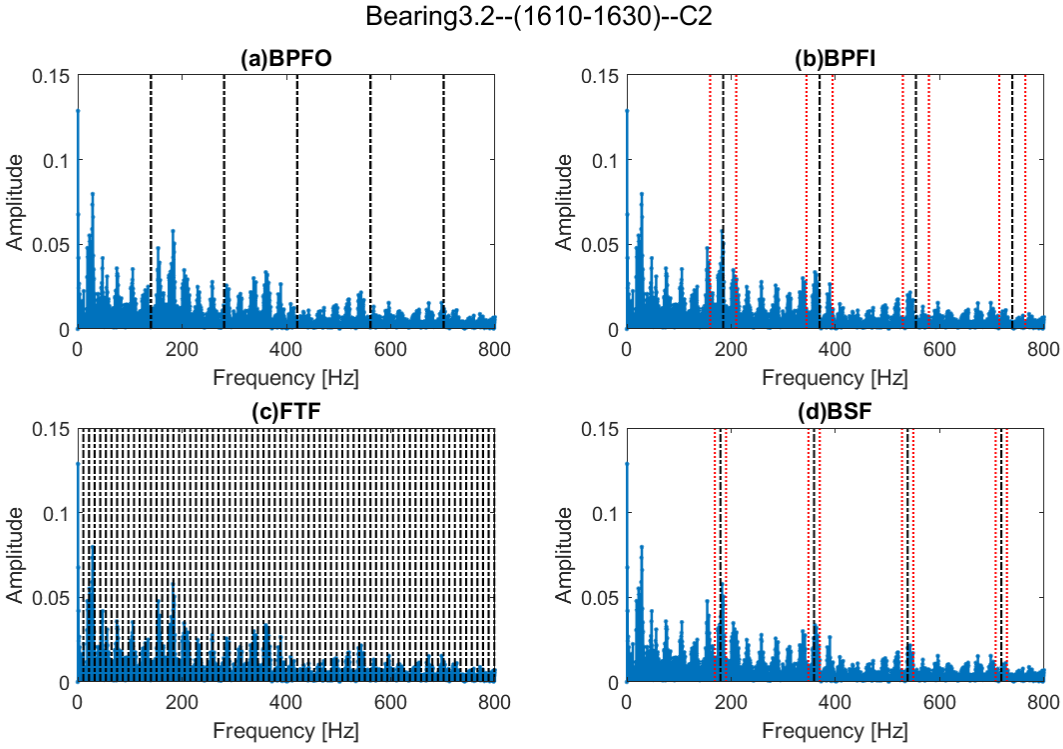


Figure A - 36 Bearing3_2 channel 2 diagnosis result

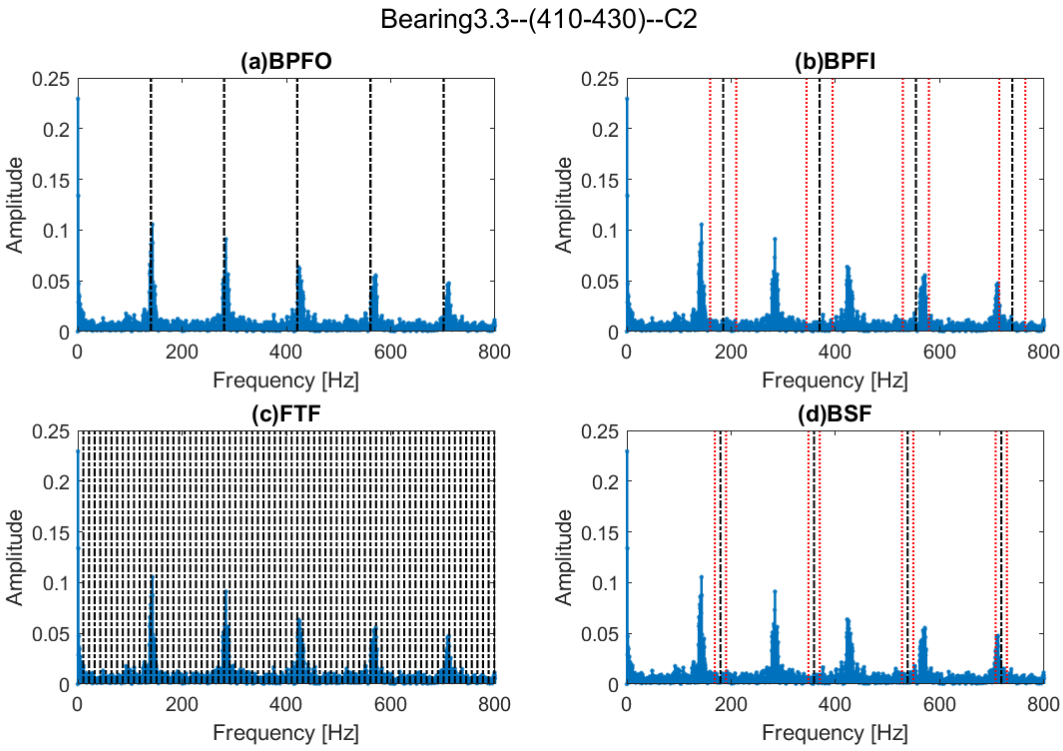


Figure A - 37 Bearing3_3 channel 1 diagnosis result

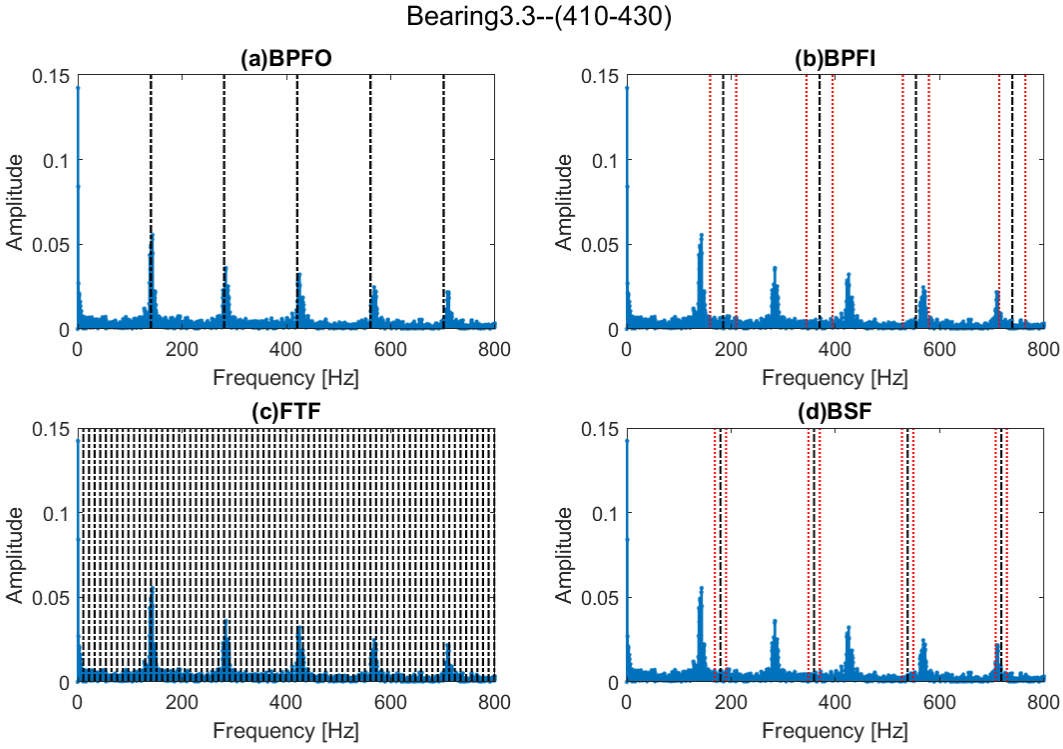


Figure A - 38 Bearing3_3 channel 2 diagnosis result

Appendix B - Prognosis results

(a) Spectrogram Channel 1 plots

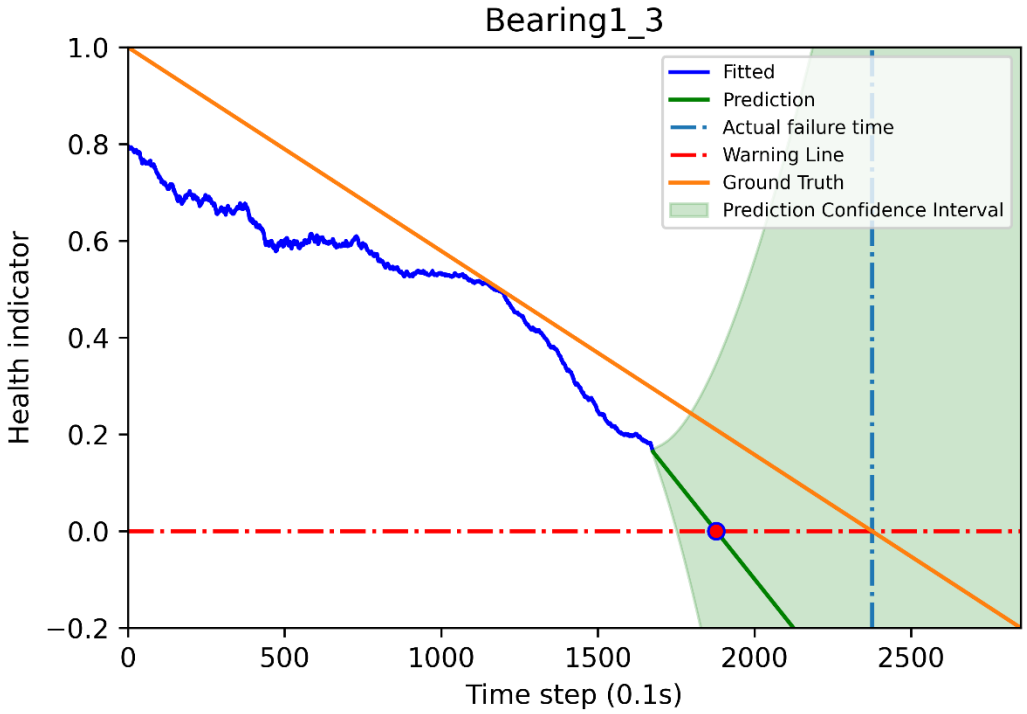


Figure B - 1 Bearing 1_3 RUL prediction

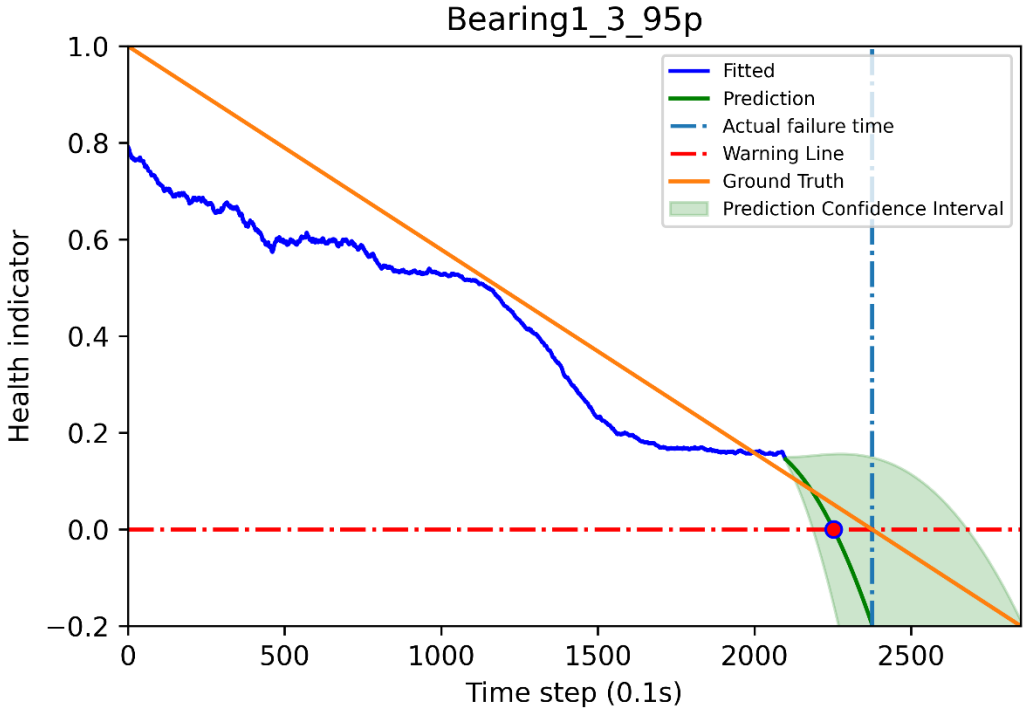


Figure B - 2 Bearing 1_3 RUL prediction at 95% data fed in

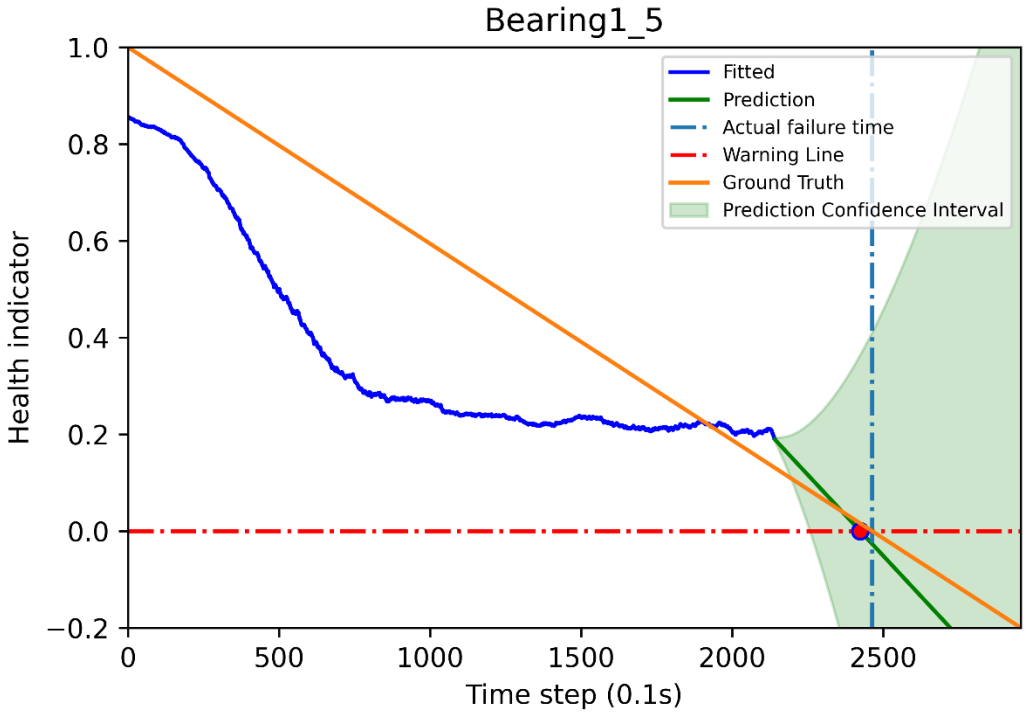


Figure B - 3 Bearing 1_5 RUL prediction

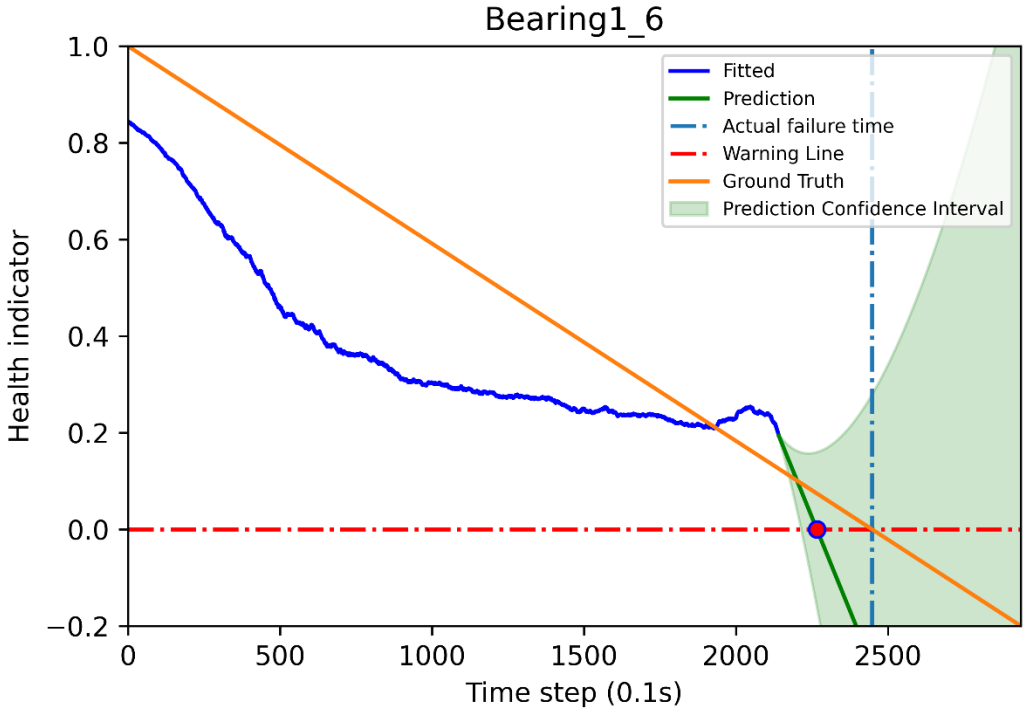


Figure B - 4 Bearing 1_6 RUL prediction

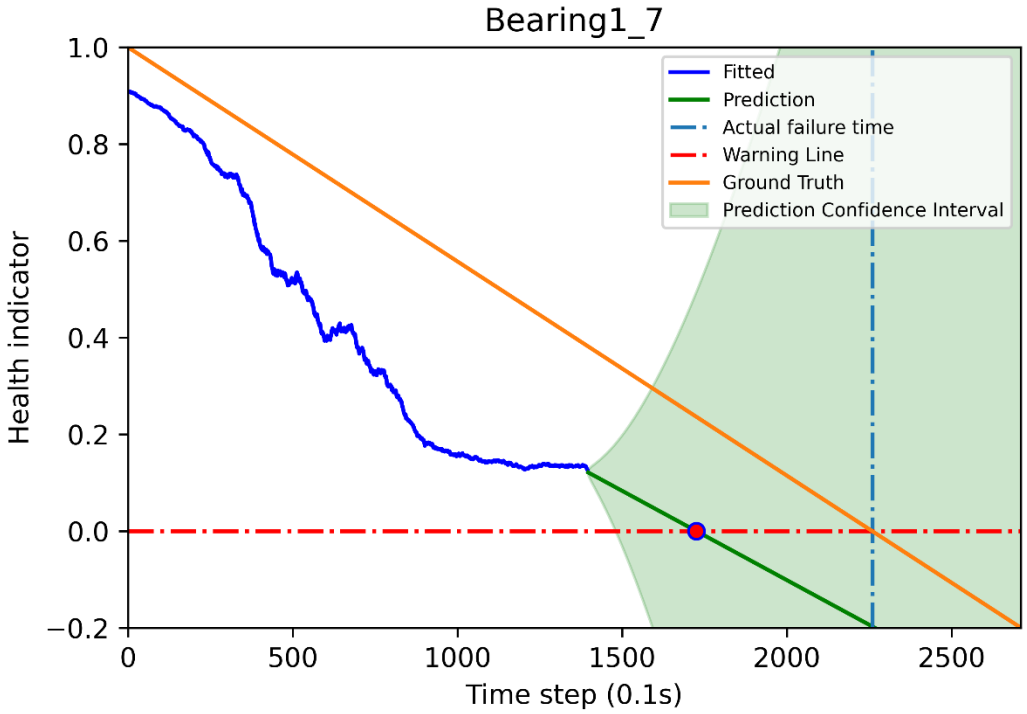


Figure B - 5 Bearing 1_7 RUL prediction

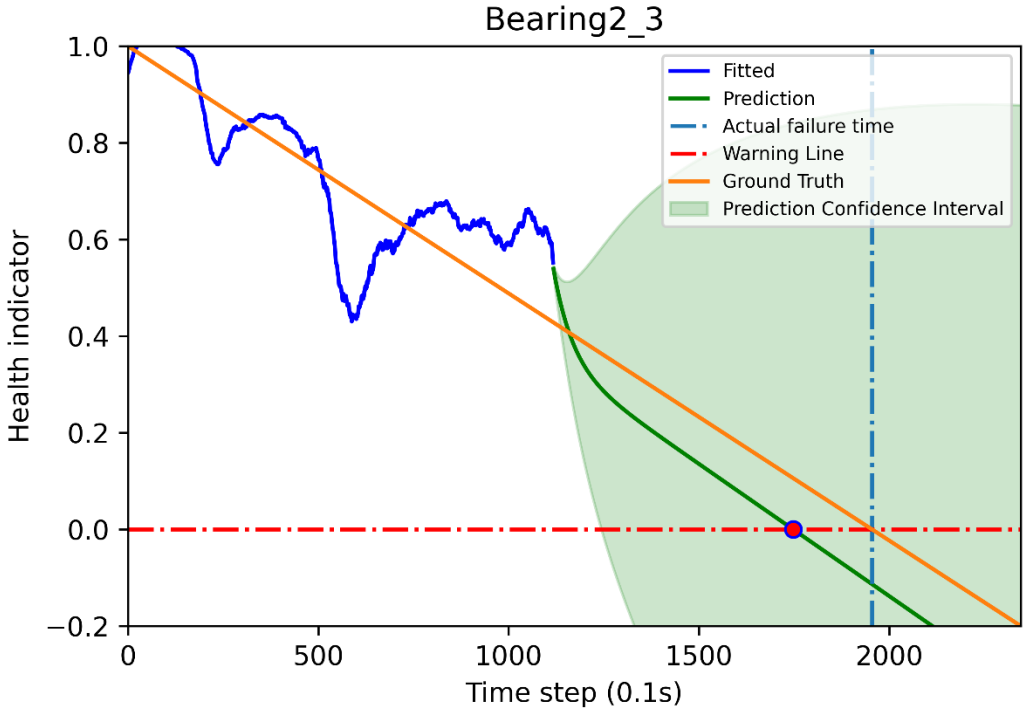


Figure B - 6 Bearing 2_3 RUL prediction

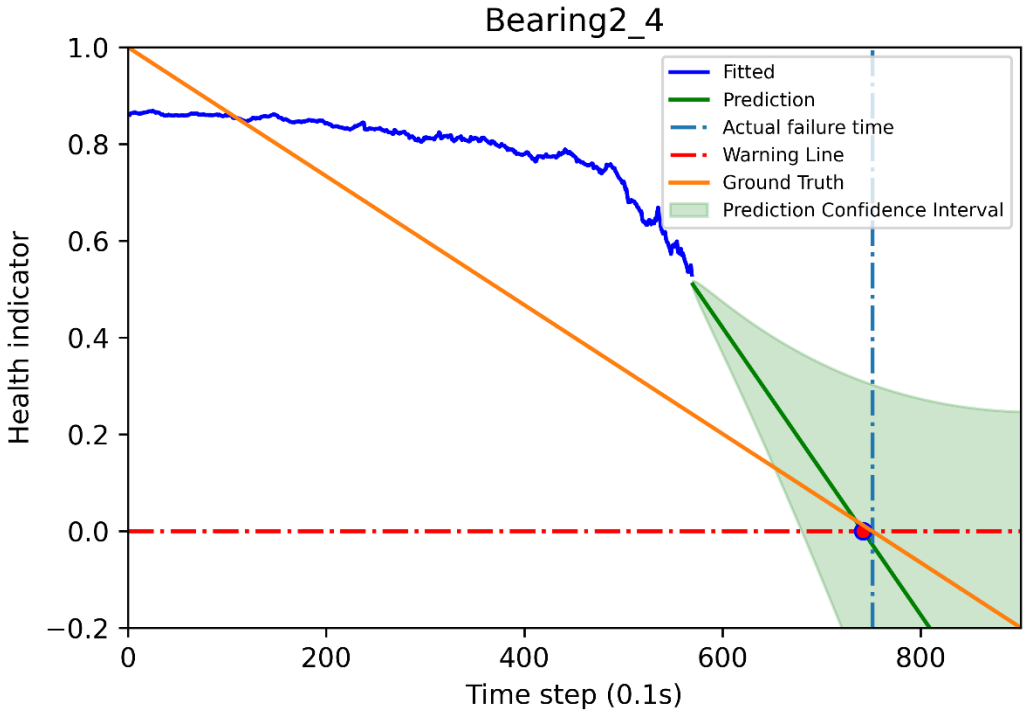


Figure B - 7 Bearing 2_4 RUL prediction

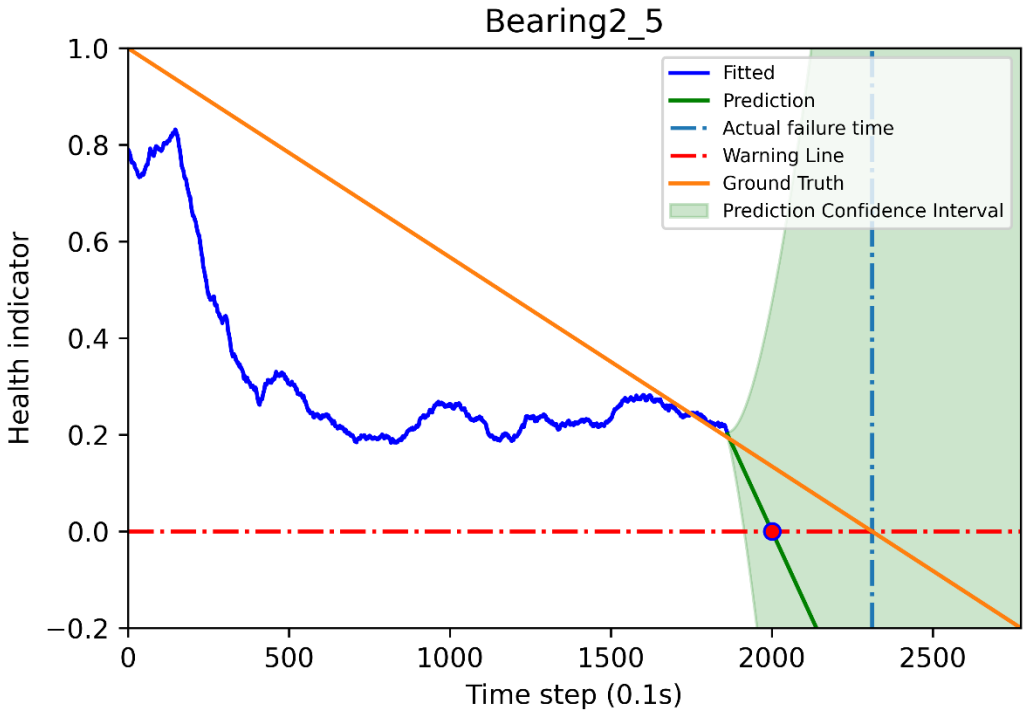


Figure B - 8 Bearing 2_5 RUL prediction

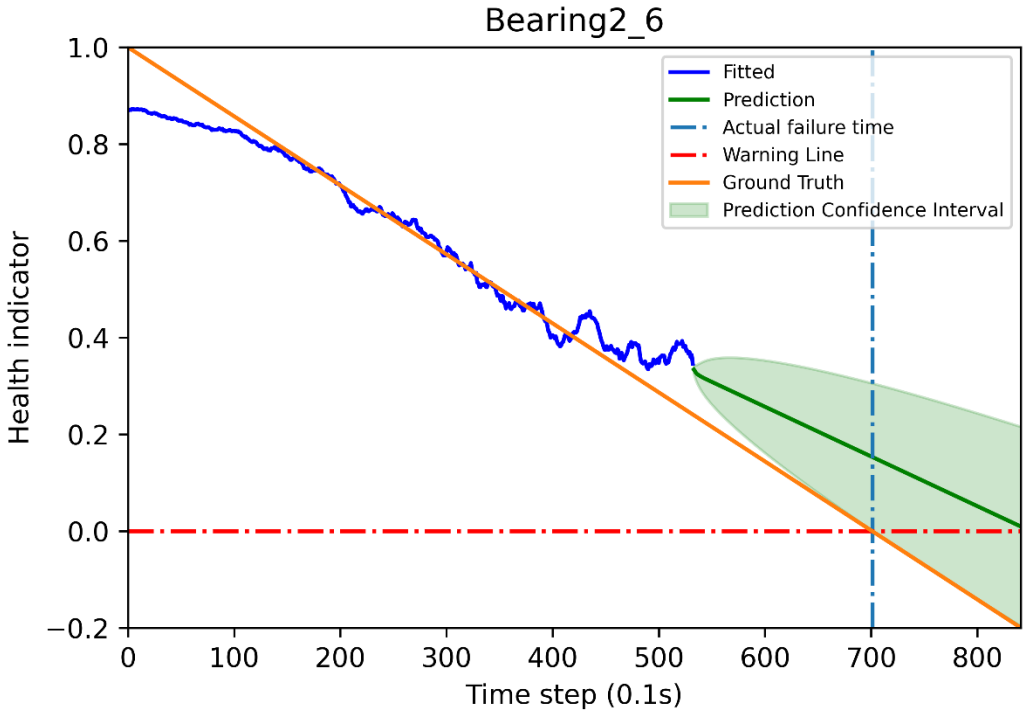


Figure B - 9 Bearing 2_6 RUL prediction

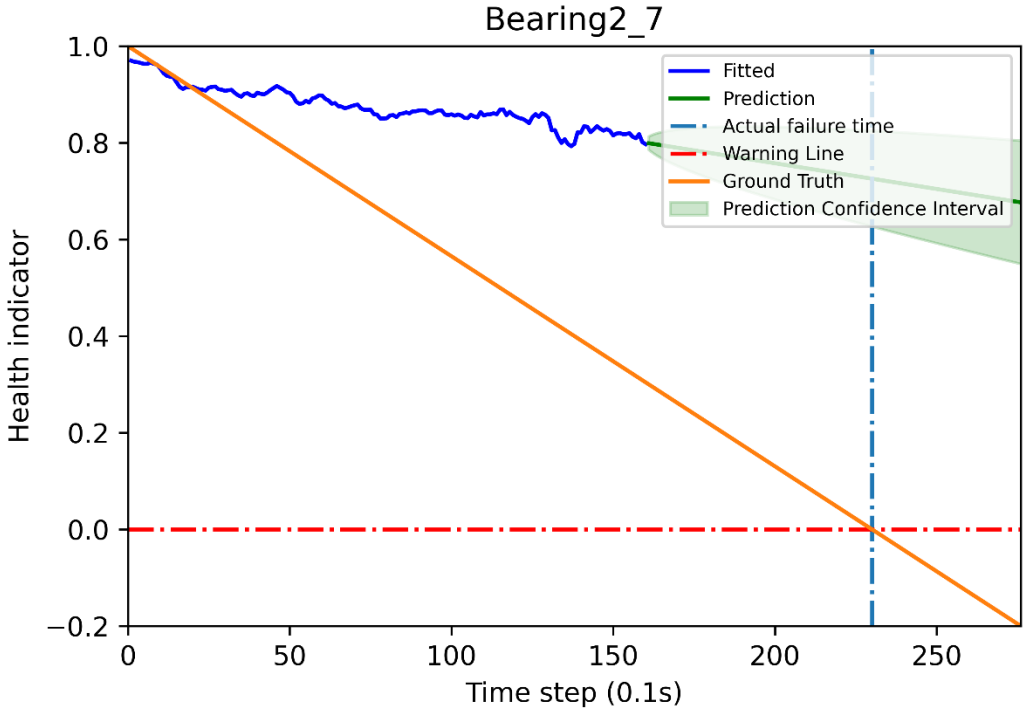


Figure B - 10 Bearing 2_7 RUL prediction

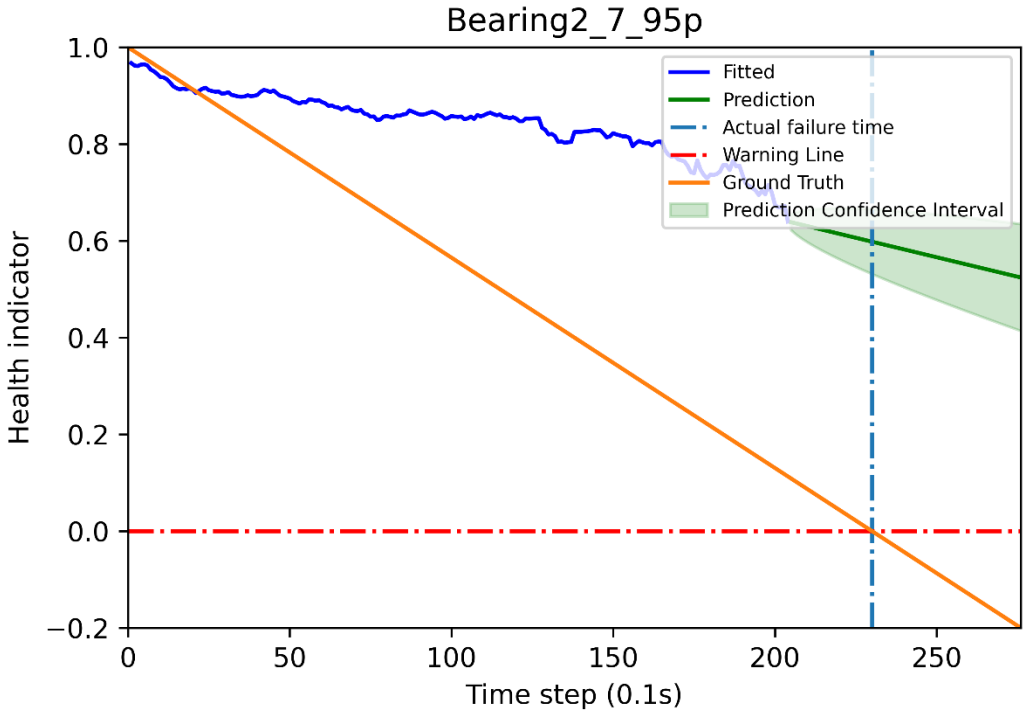


Figure B - 11 Bearing 2_7 RUL prediction at 95% data fed in

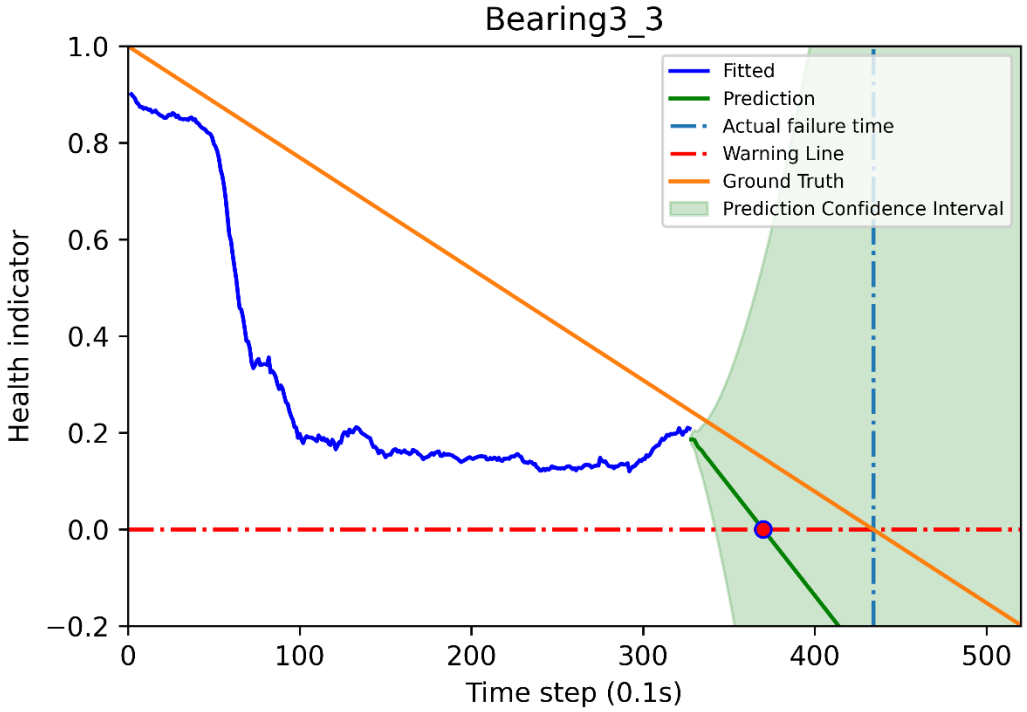


Figure B - 12 Bearing 3_3 RUL prediction

(b) Spectrogram channel 2 plots

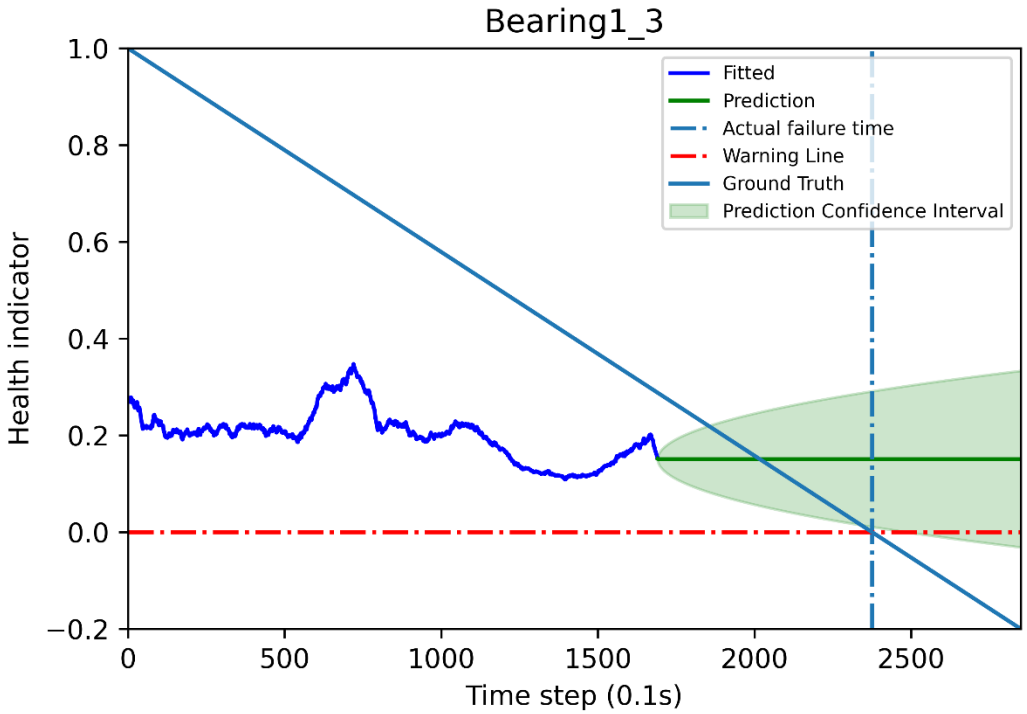


Figure B - 13 Bearing 1_3 RUL prediction

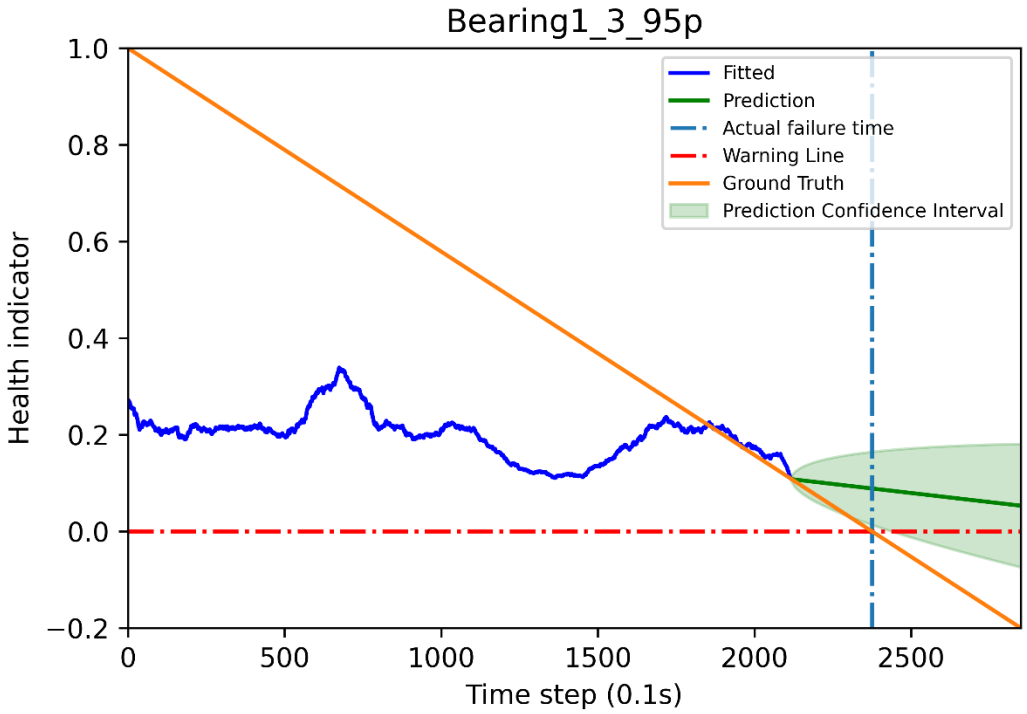


Figure B - 14 Bearing 1_3 RUL prediction at 95% data fed in

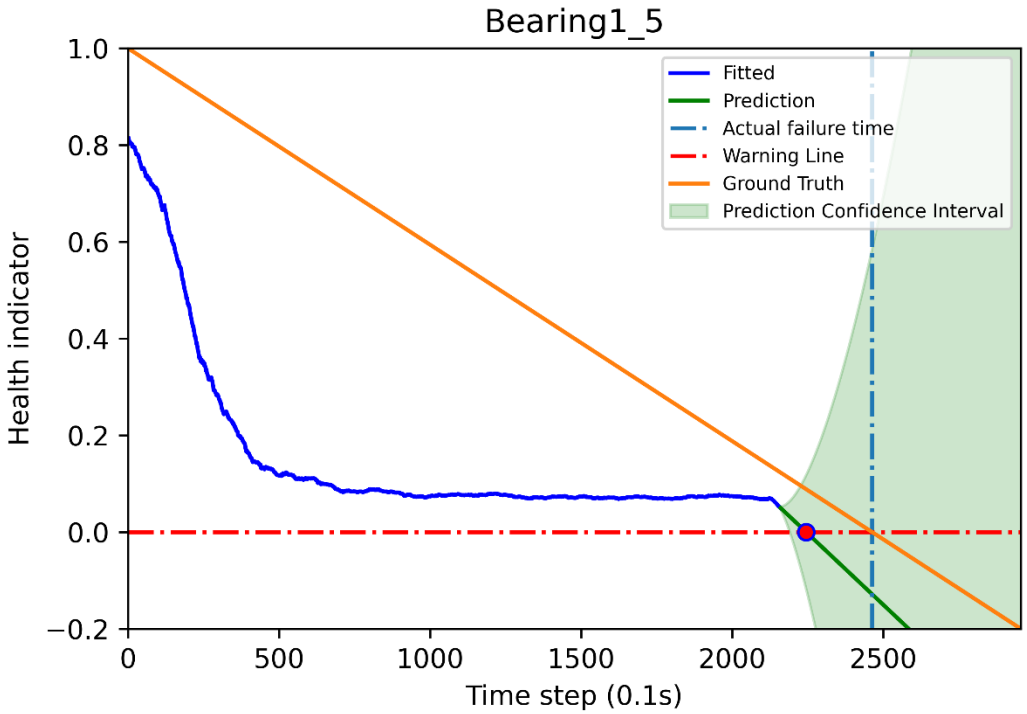


Figure B - 15 Bearing 1_5 RUL prediction

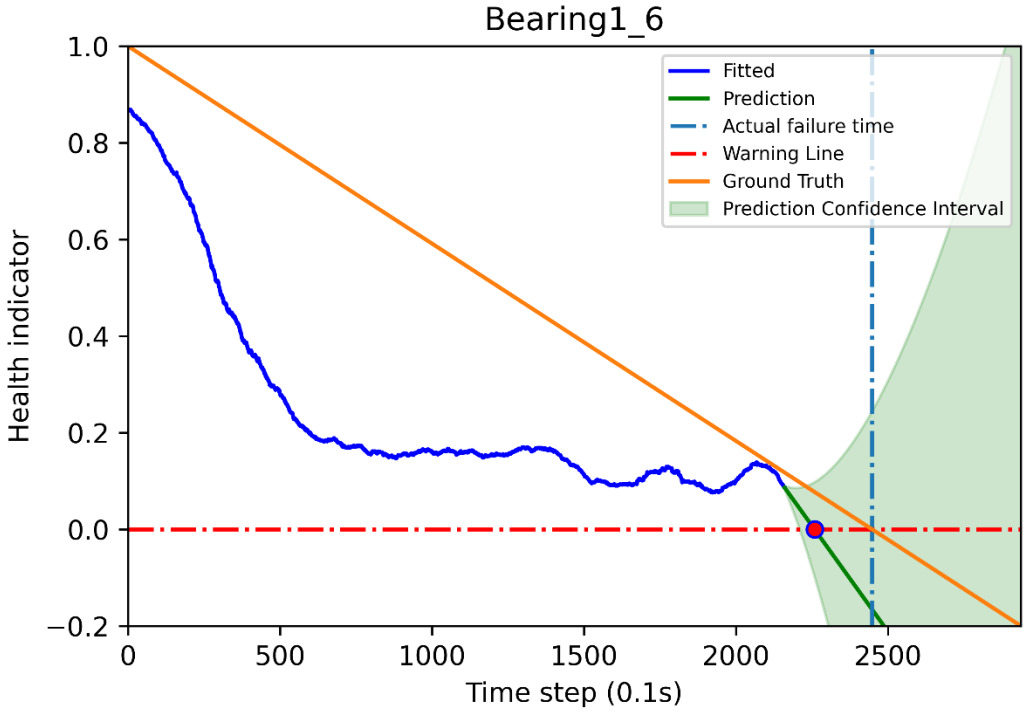


Figure B - 16 Bearing 1_6 RUL prediction

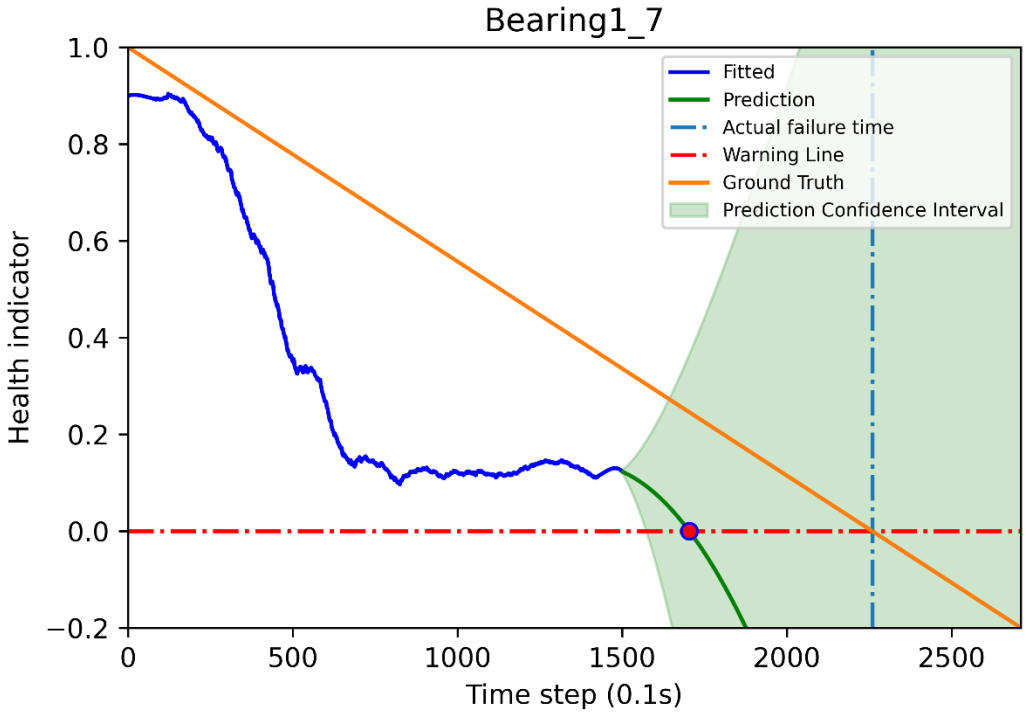


Figure B - 17 Bearing 1_7 RUL prediction

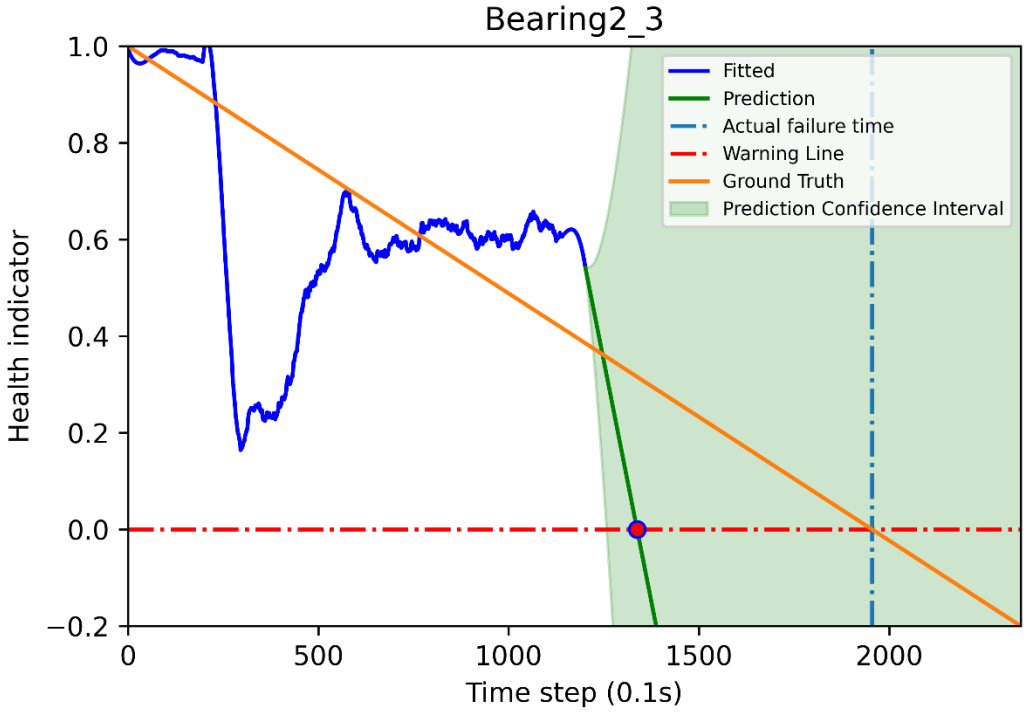


Figure B - 18 Bearing 2_3 RUL prediction

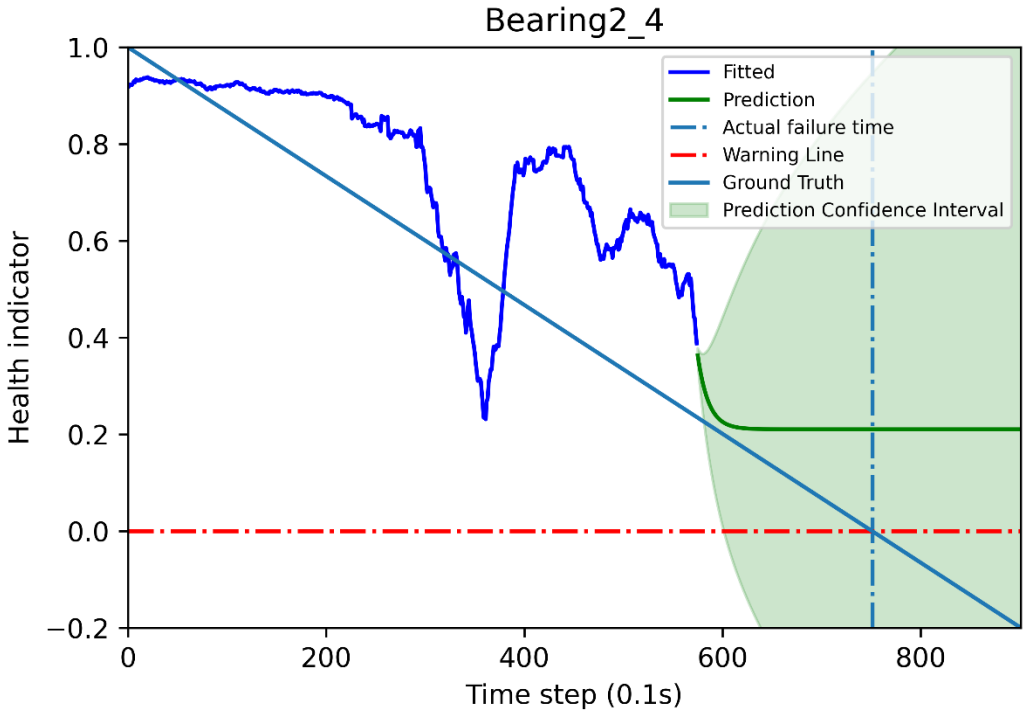


Figure B - 19 Bearing 2_4 RUL prediction

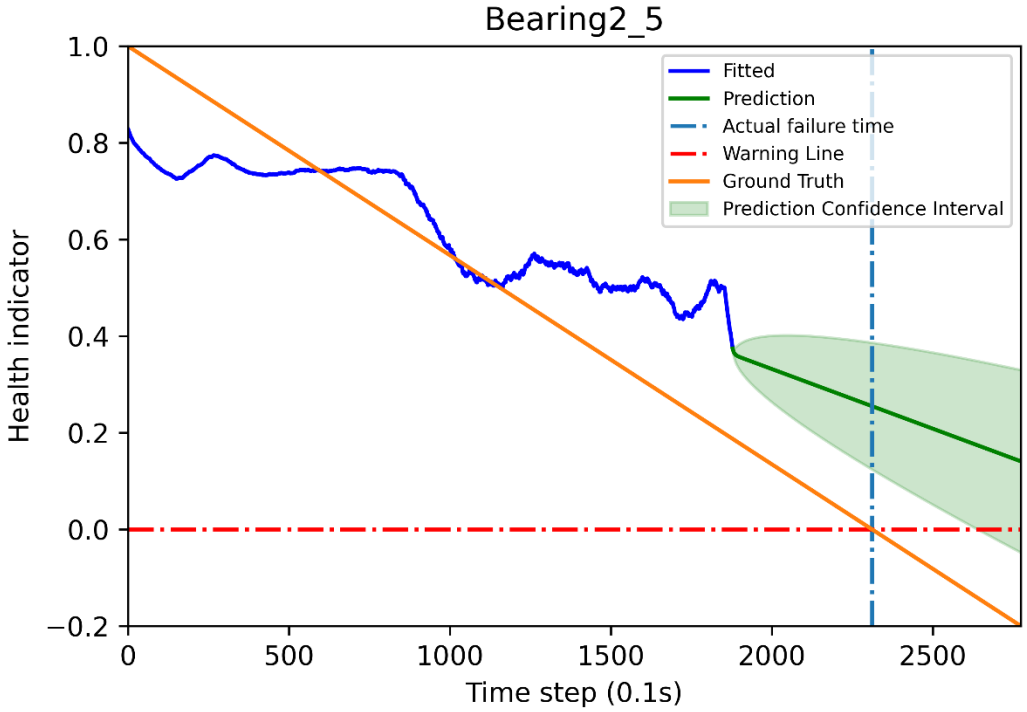


Figure B - 20 Bearing 2_5 RUL prediction

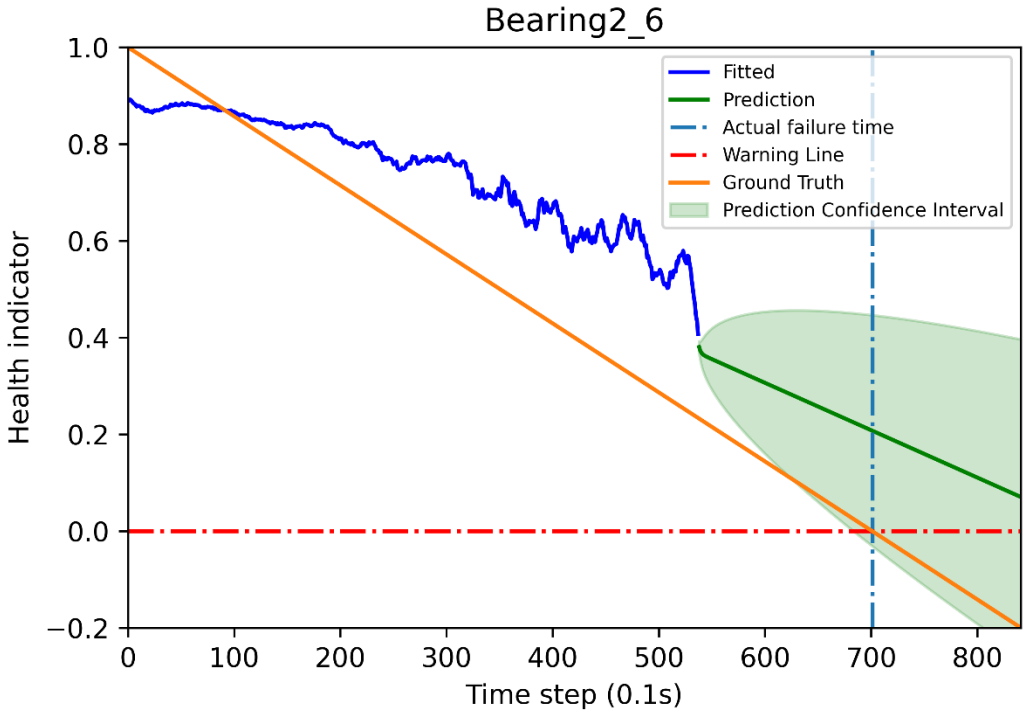


Figure B - 21 Bearing 2_6 RUL prediction

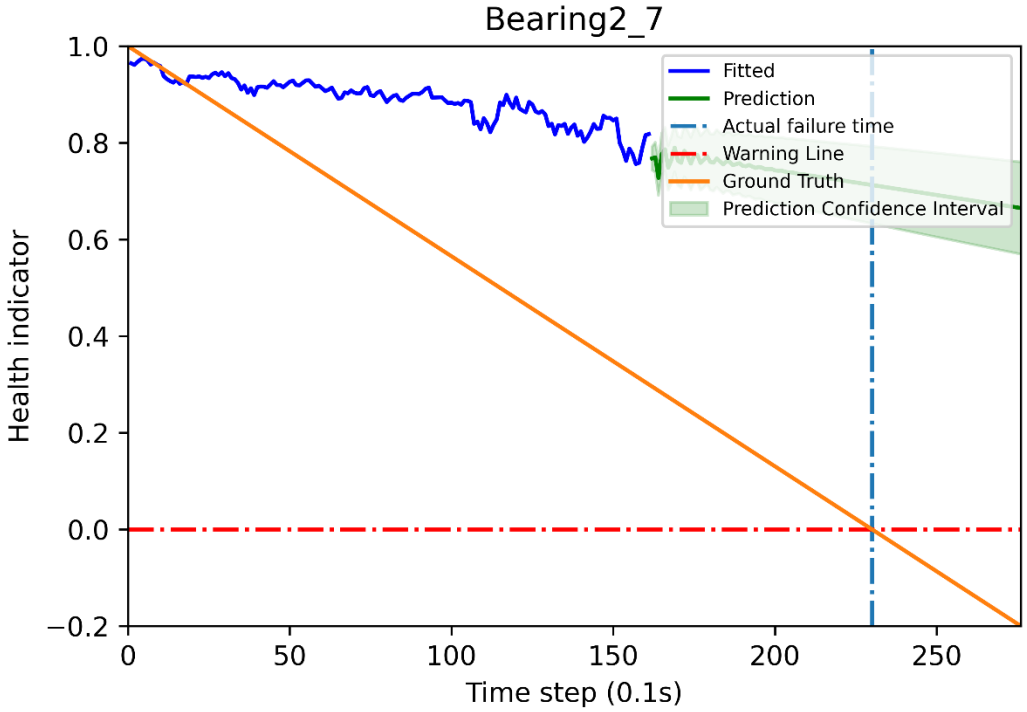


Figure B - 22 Bearing 2_7 RUL prediction

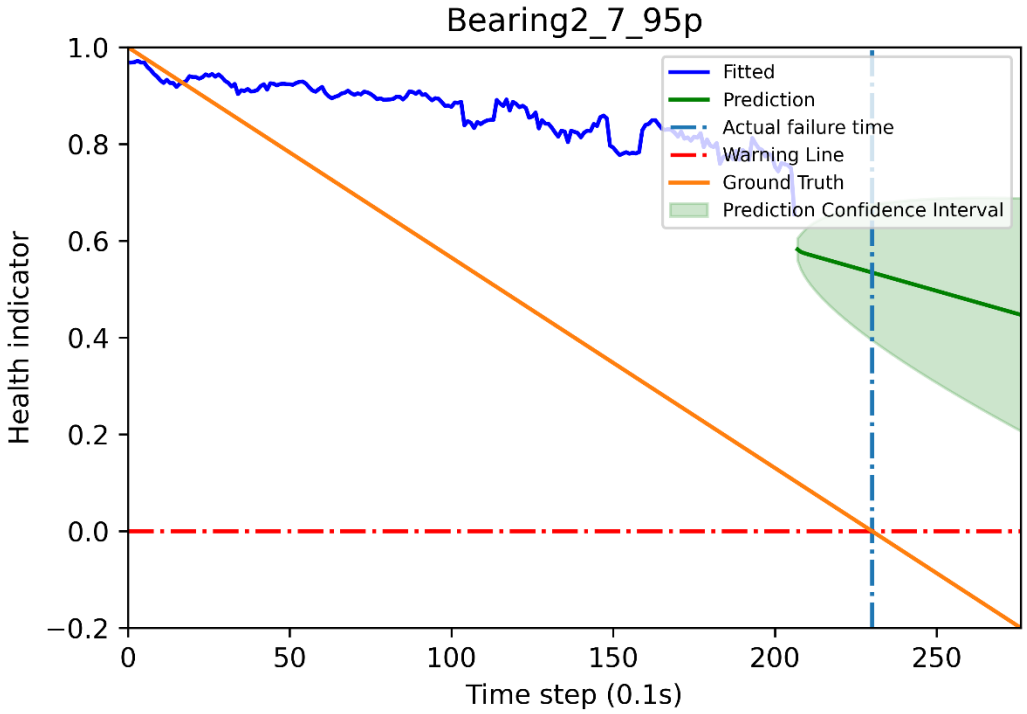
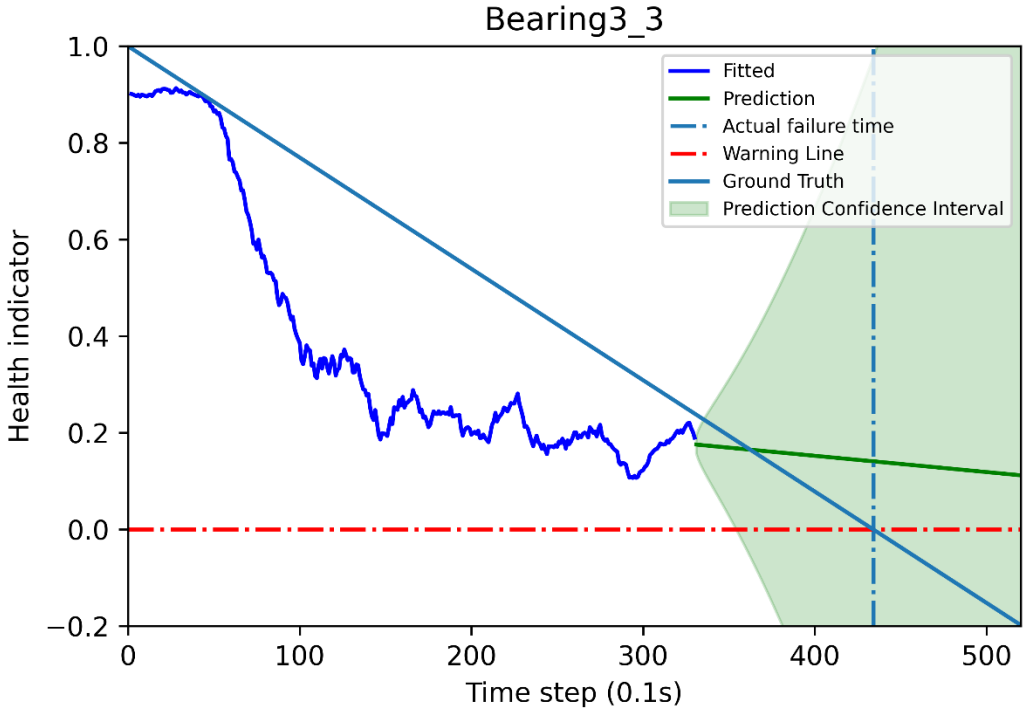


Figure B – 23 Bearing 2_7 RUL prediction at 95% data fed in



Figure_APX B - 24 Bearing 3_3 RUL prediction

(c) Channel fusion plots

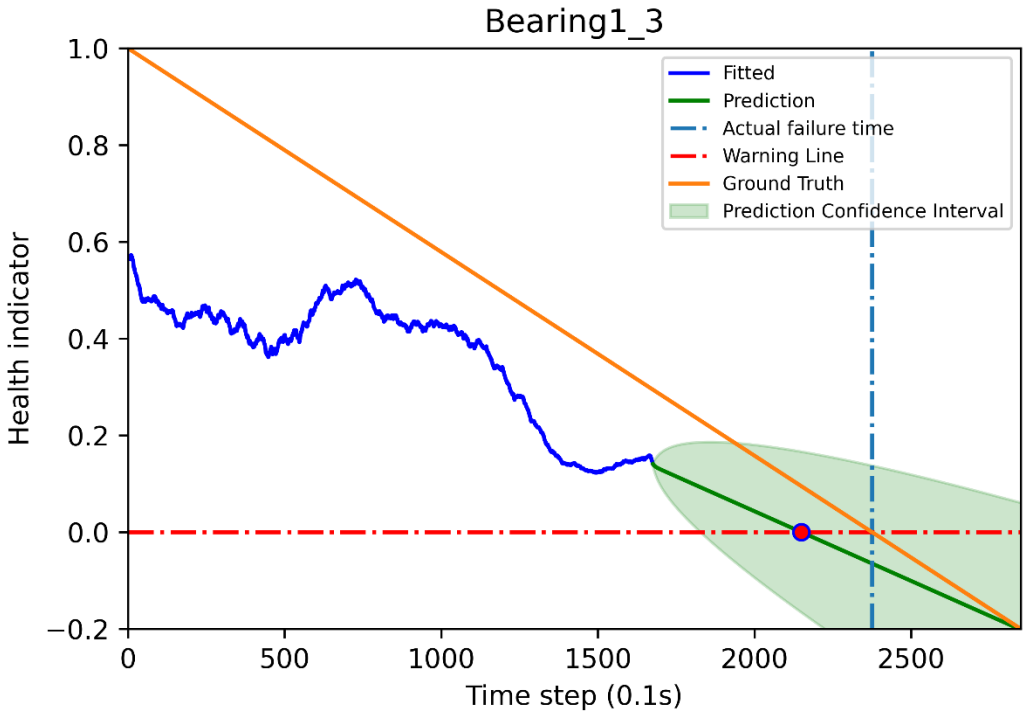
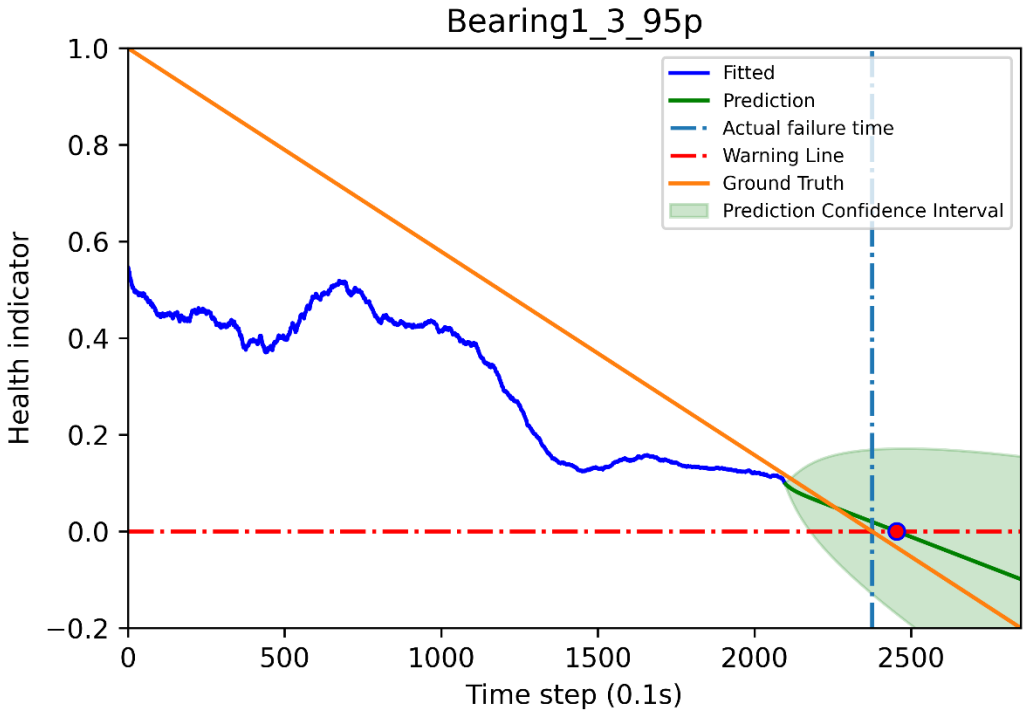


Figure B - 25 Bearing 1_3 RUL prediction



Figure_APX B - 26 Bearing 1_3_95p RUL prediction

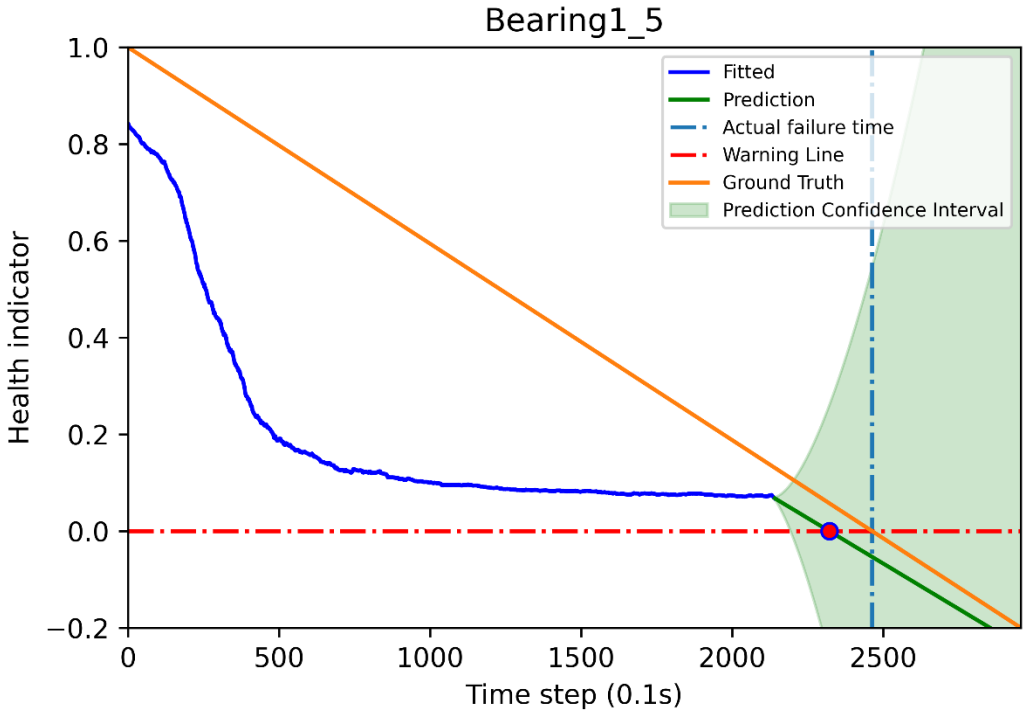


Figure B - 27 Bearing 1_5 RUL prediction

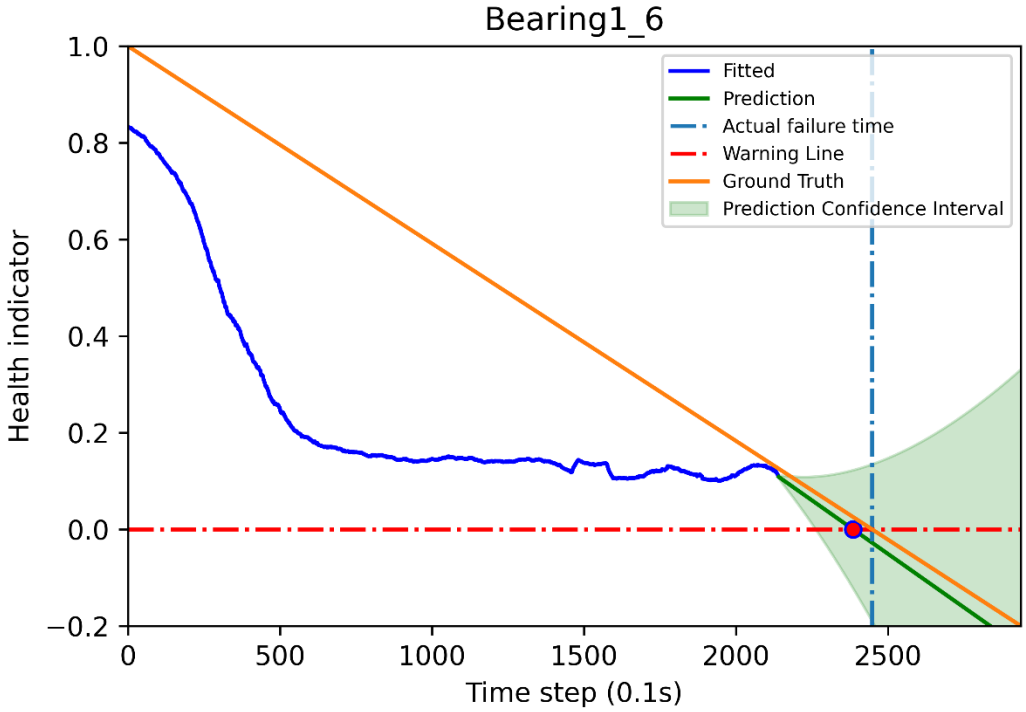


Figure B - 28Bearing 1_6 RUL prediction

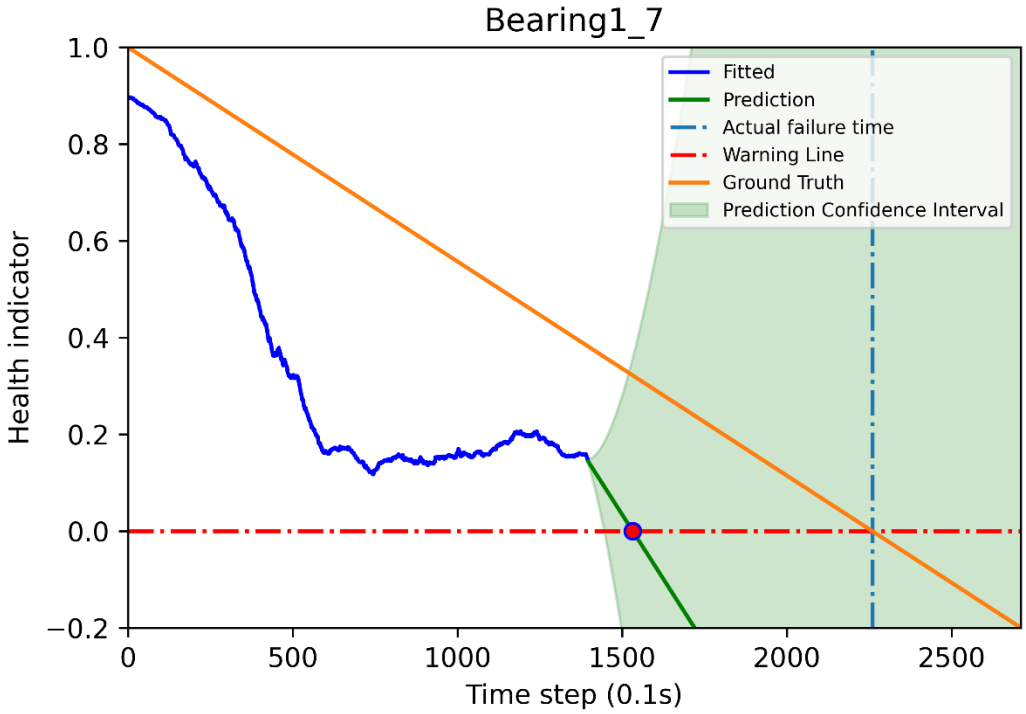


Figure B - 29 Bearing 1_7 RUL prediction

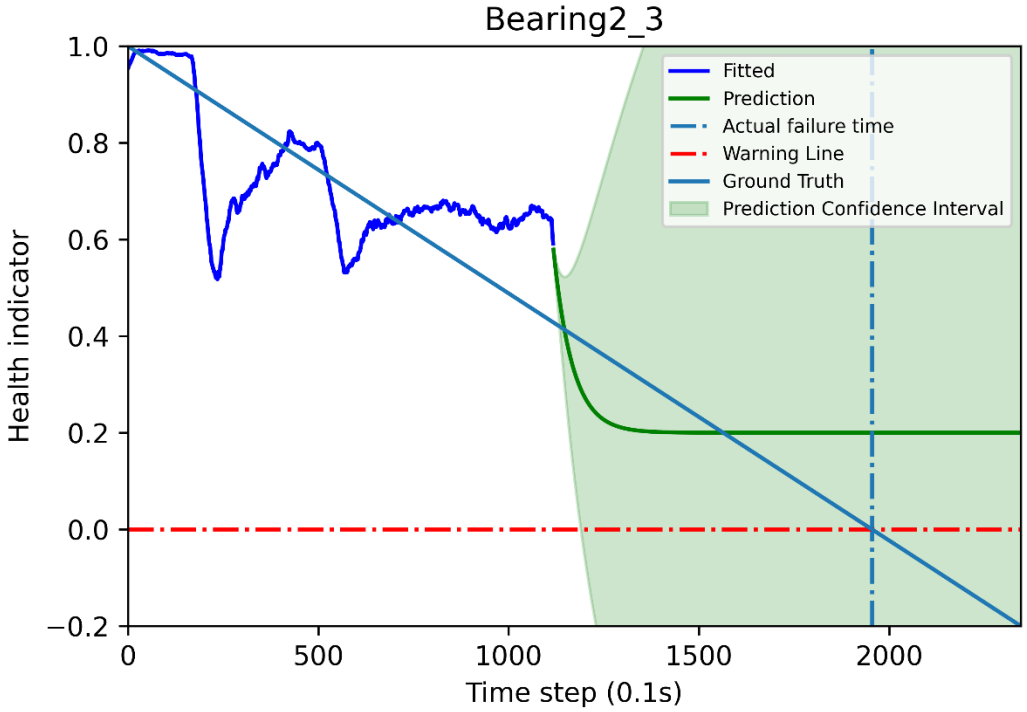


Figure B - 30 Bearing 2_3 RUL prediction

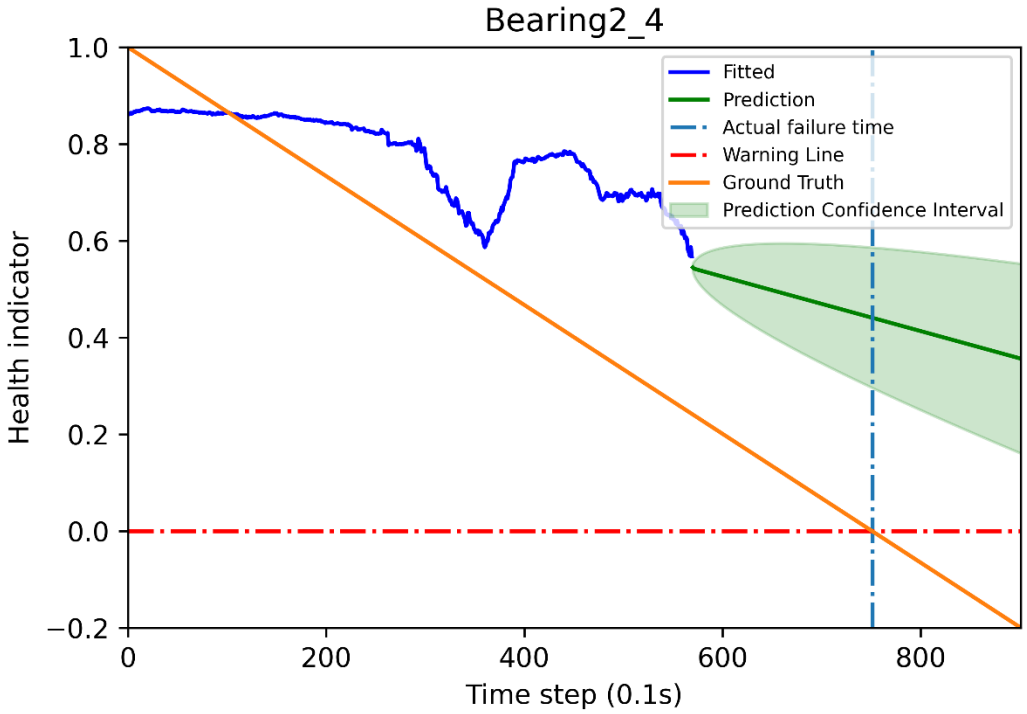


Figure B - 31 Bearing 2_4 RUL prediction

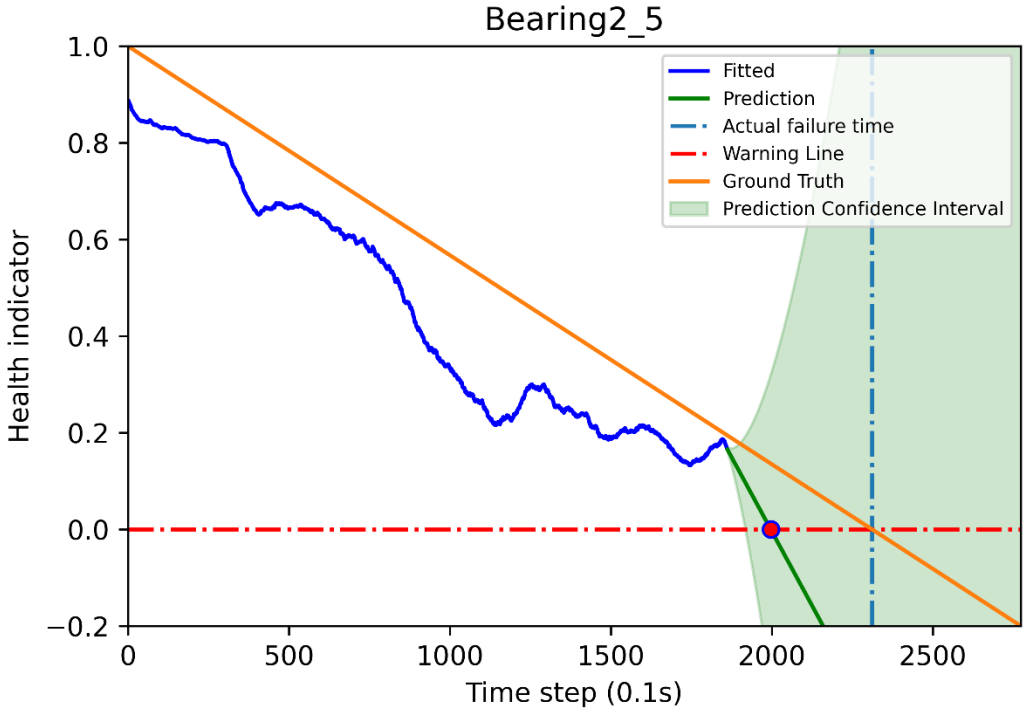


Figure B - 32 Bearing 2_5 RUL prediction

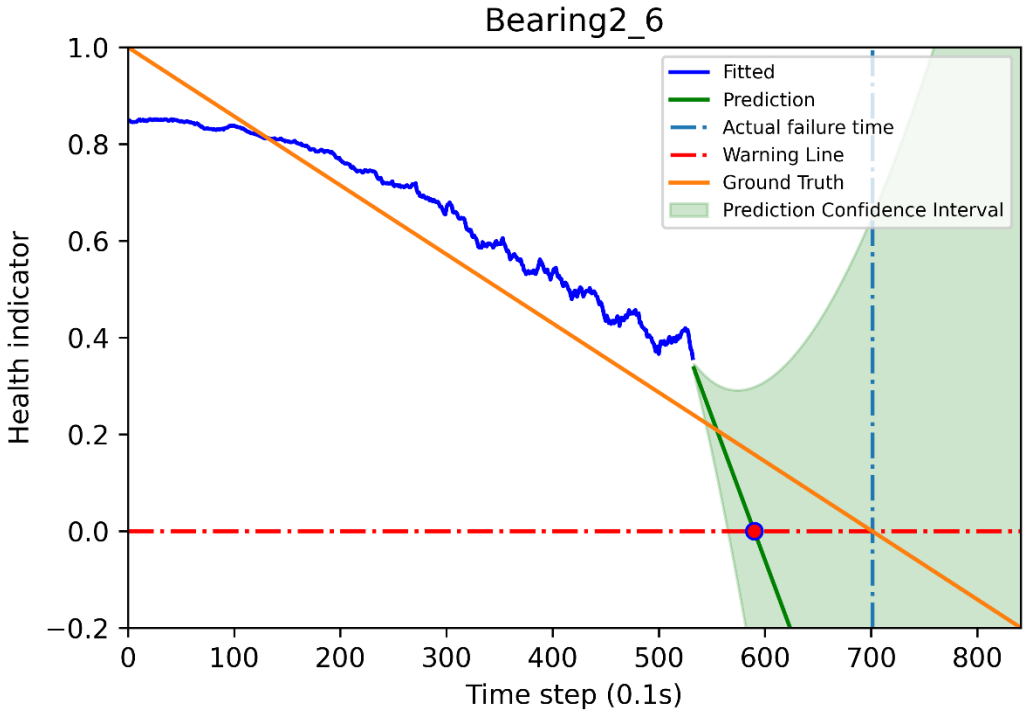


Figure B - 33 Bearing 2_6 RUL prediction

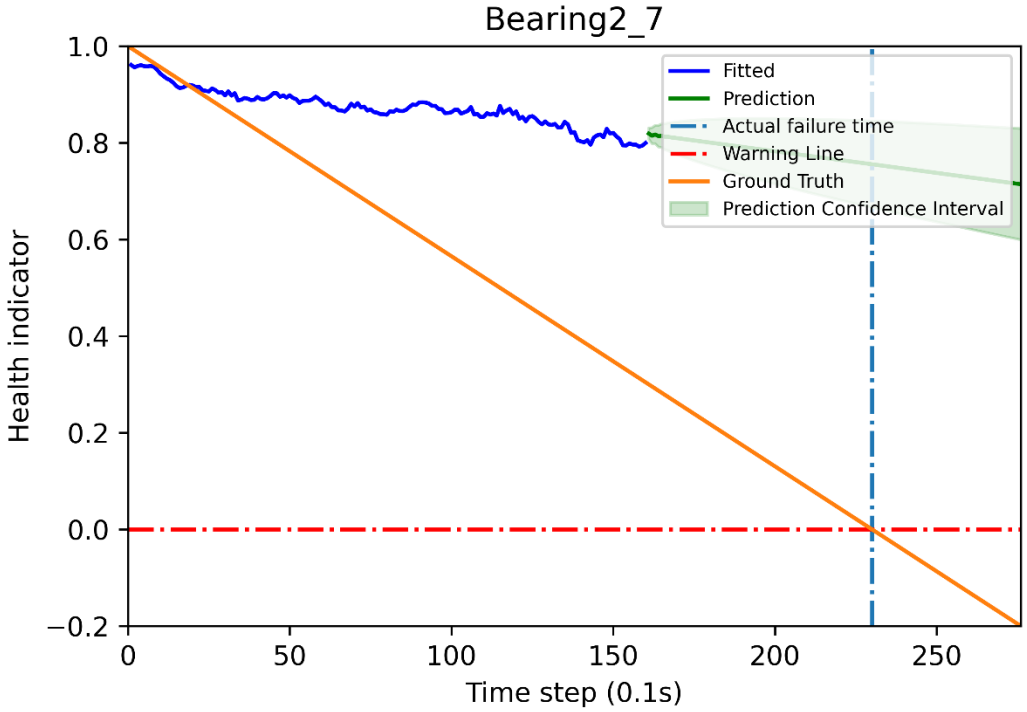


Figure B - 34 Bearing 2_7 RUL prediction

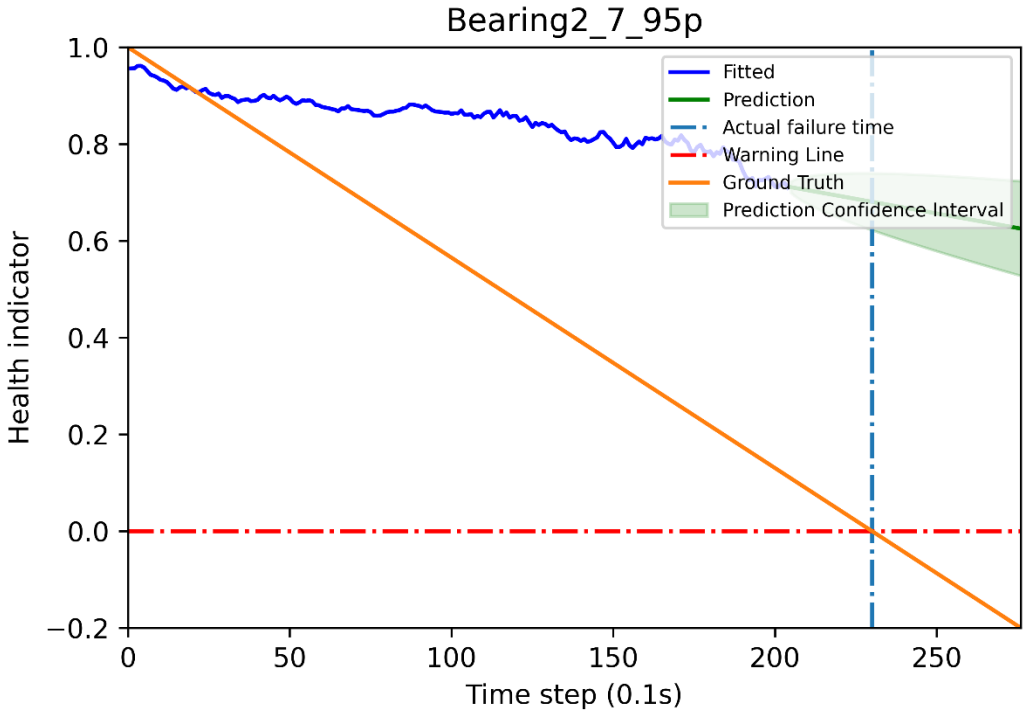


Figure B - 35 Bearing 2_7_95p RUL prediction

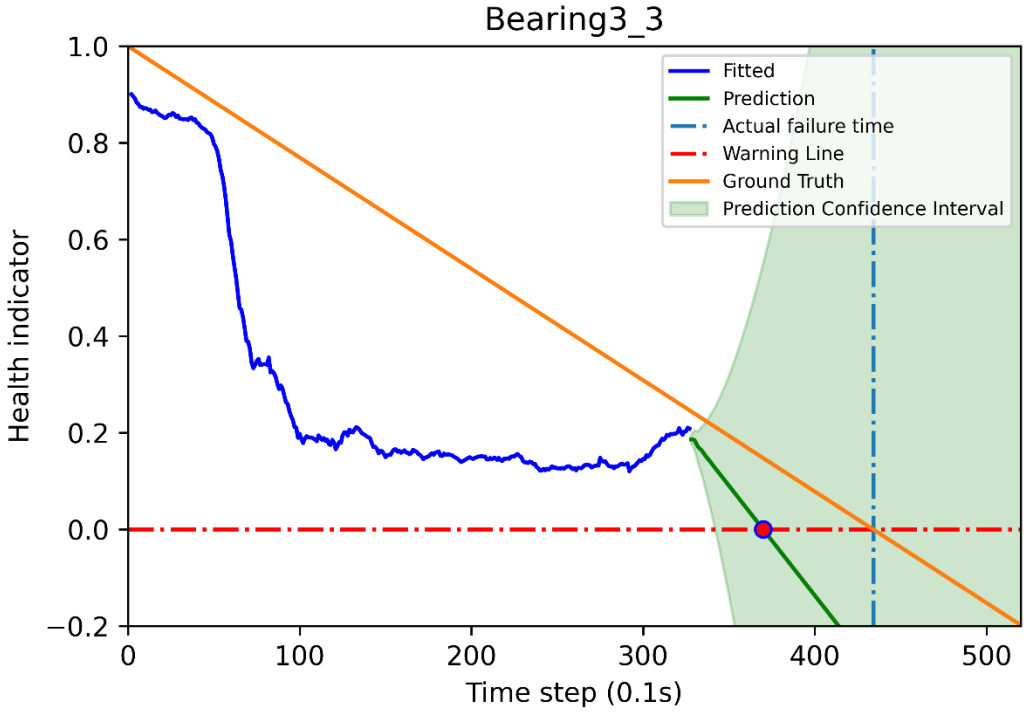


Figure B - 36 Bearing 3_3 RUL prediction

(d)Scalogram channel 1 plots

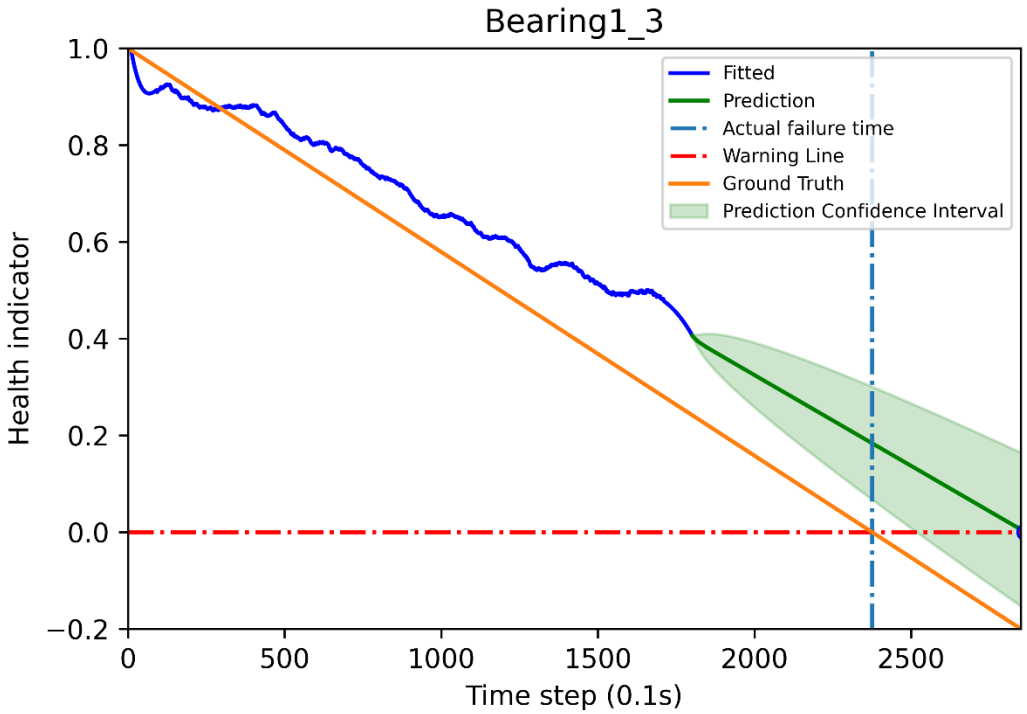


Figure B - 37 Bearing 1_3 RUL prediction

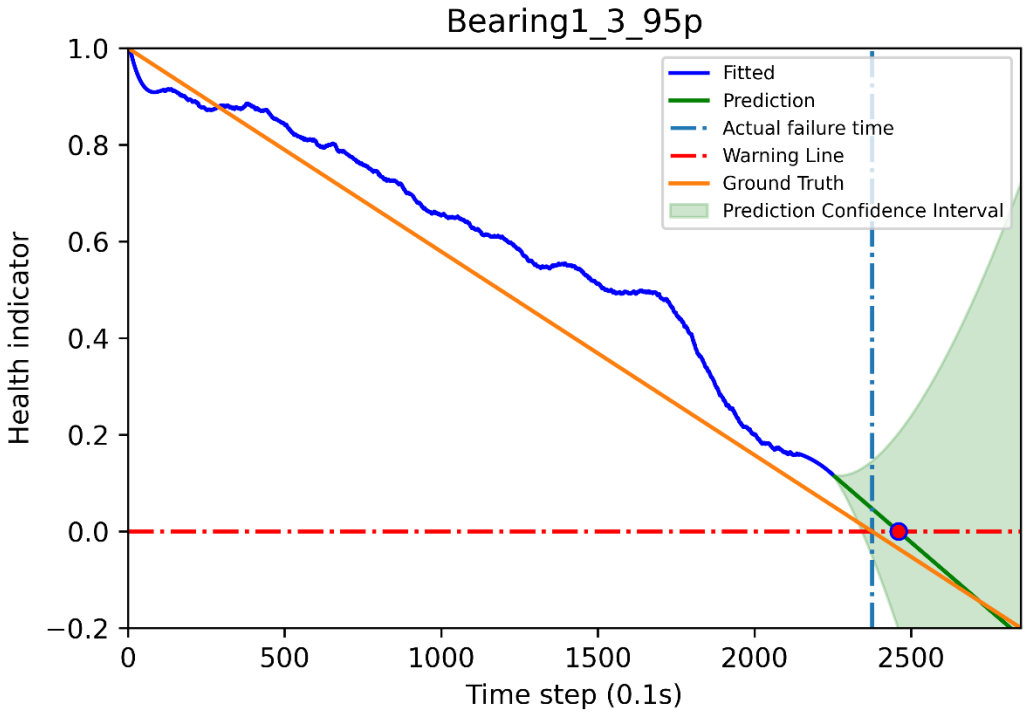


Figure B - 38 Bearing 1_3_95p RUL prediction

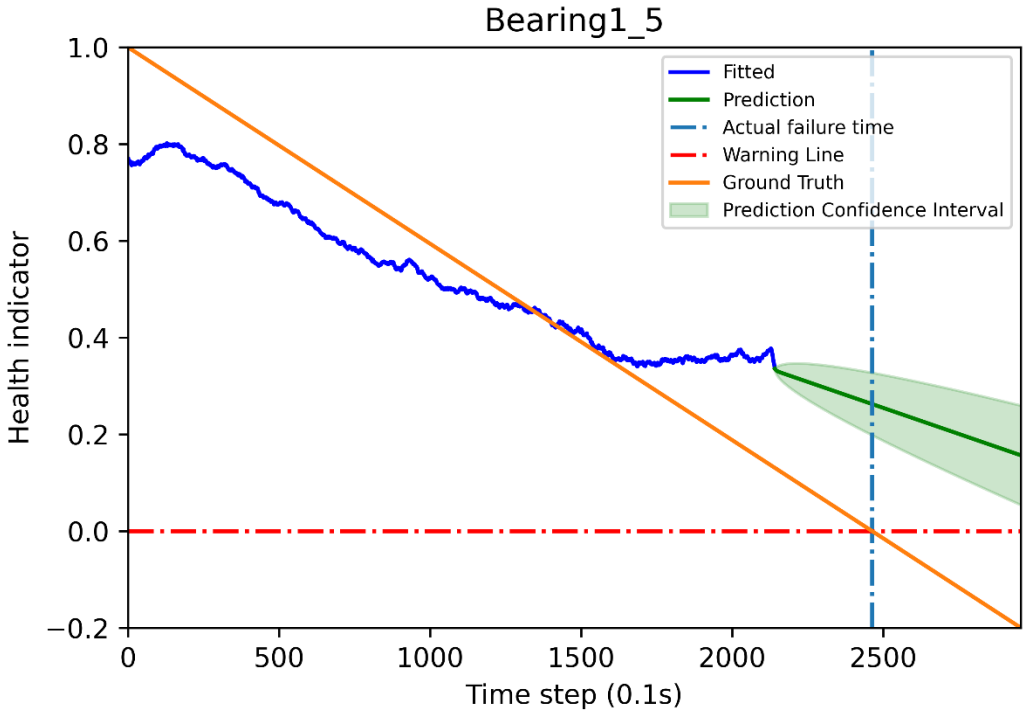


Figure B – 39 Bearing 1_5 RUL prediction

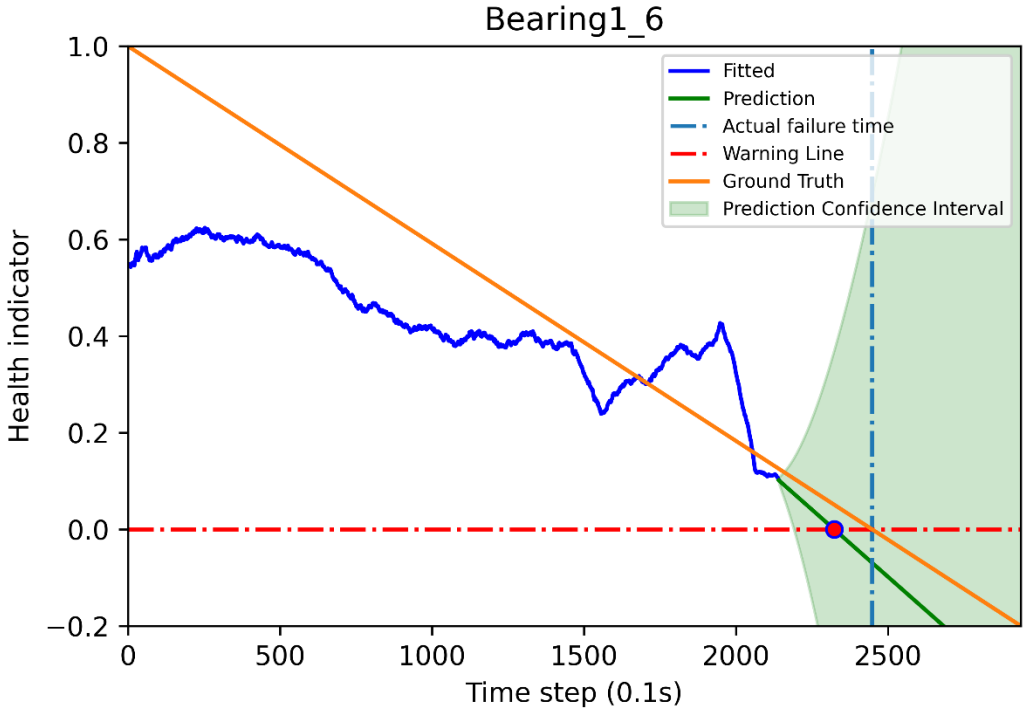


Figure B - 40 Bearing 1_6 RUL prediction

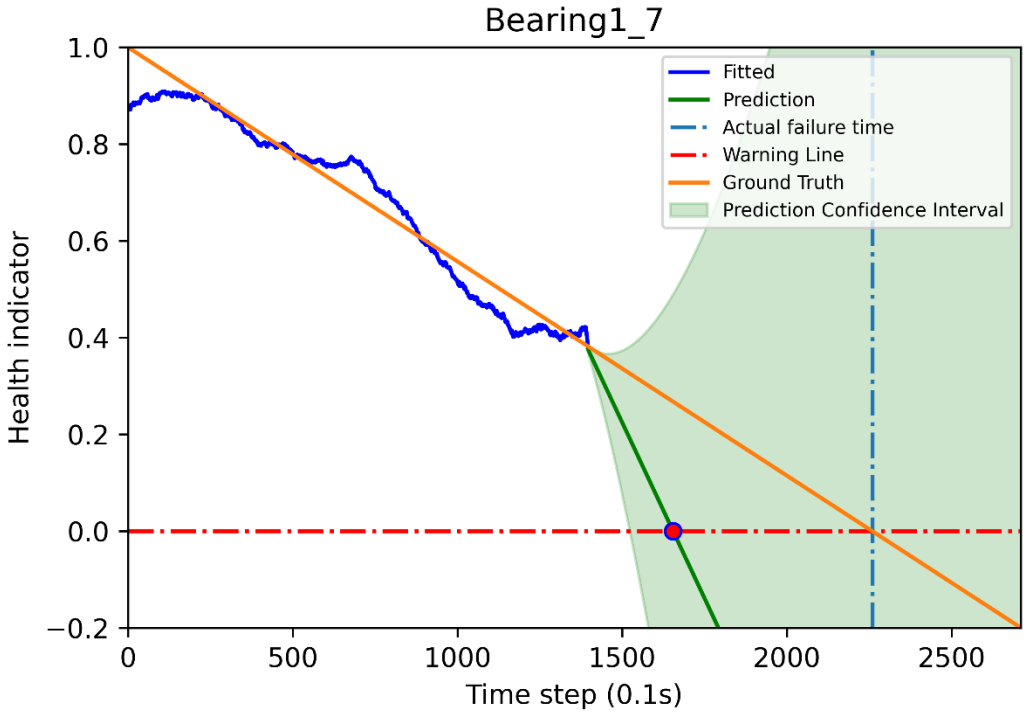


Figure B - 41 Bearing 1_7 RUL prediction

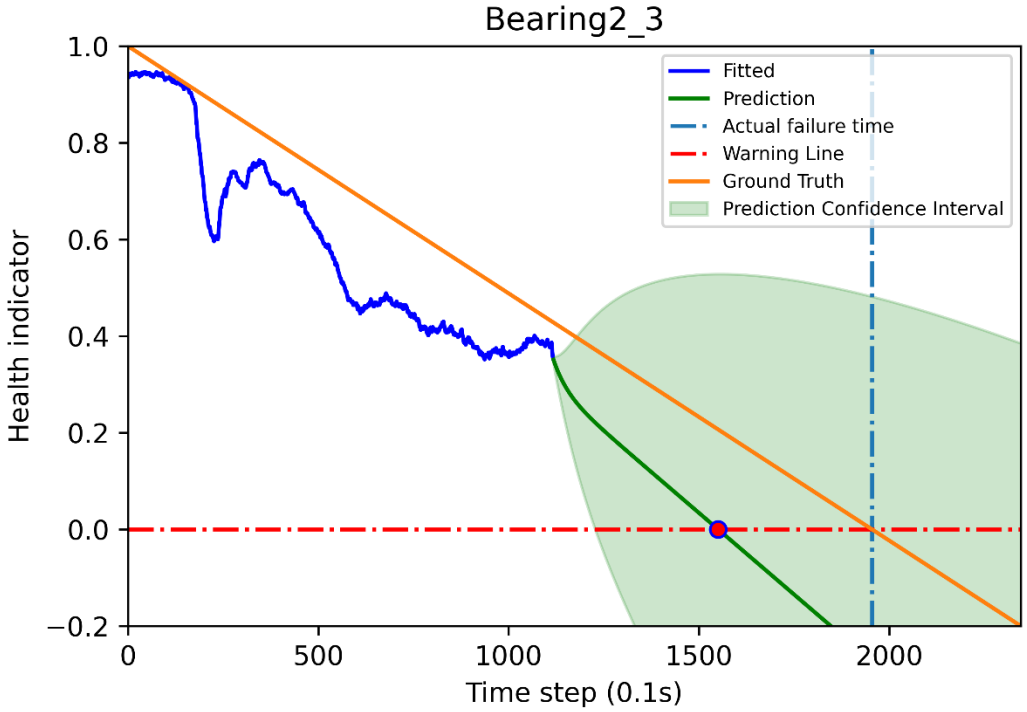


Figure B – 42 Bearing 2_3 RUL prediction

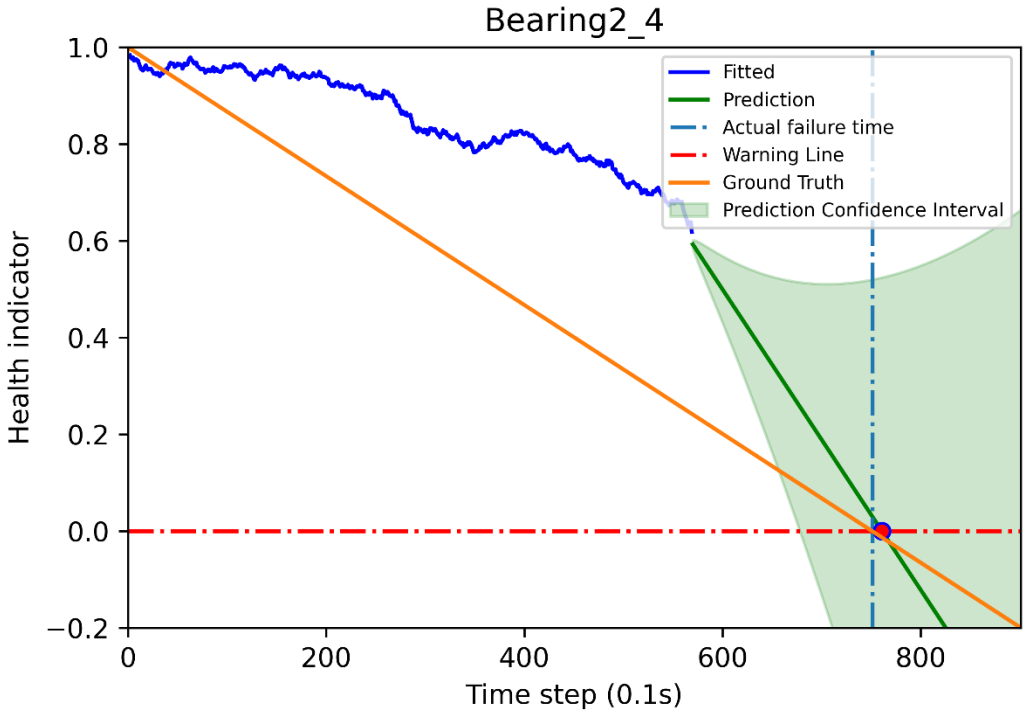


Figure B – 43 Bearing 2_4 RUL prediction

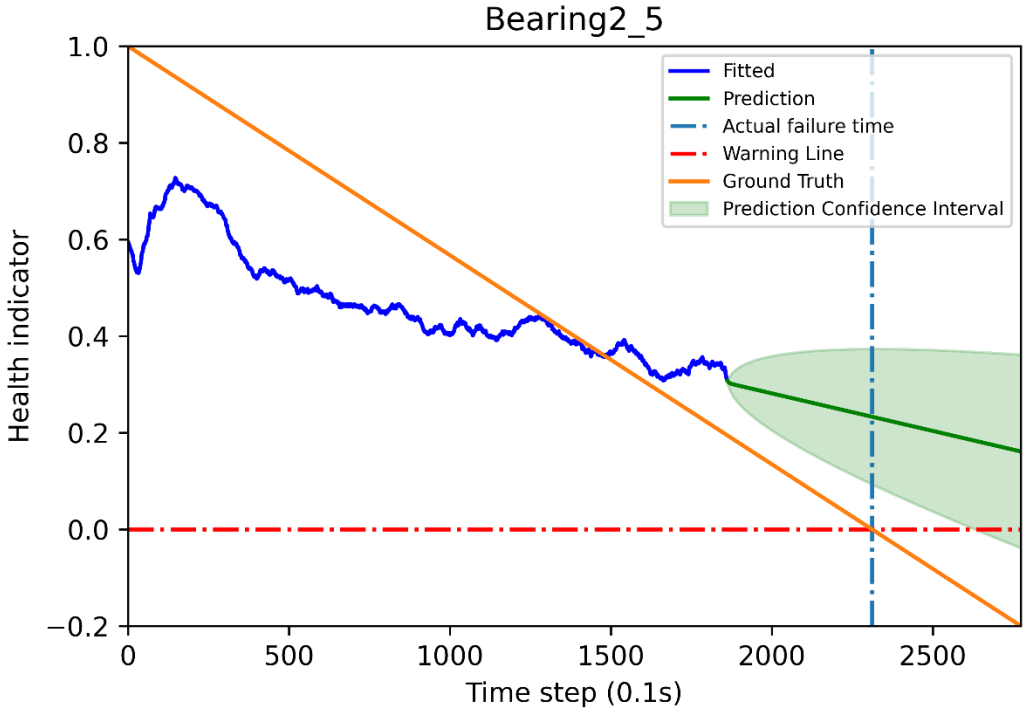


Figure B – 44 Bearing 2_5 RUL prediction

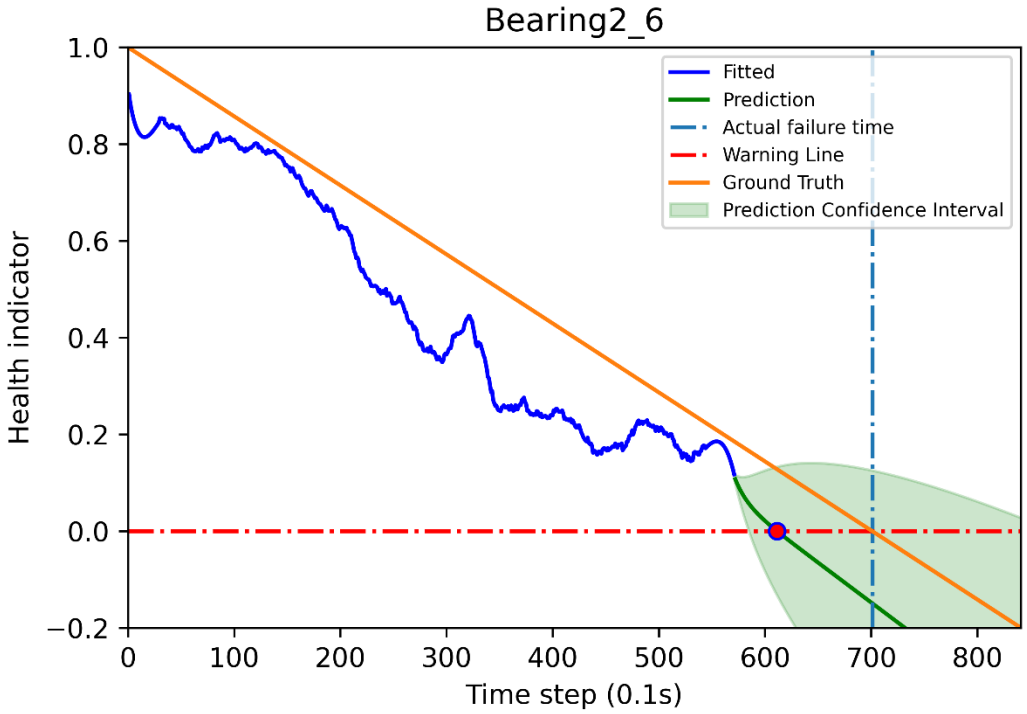


Figure B – 45 Bearing 2_6 RUL prediction

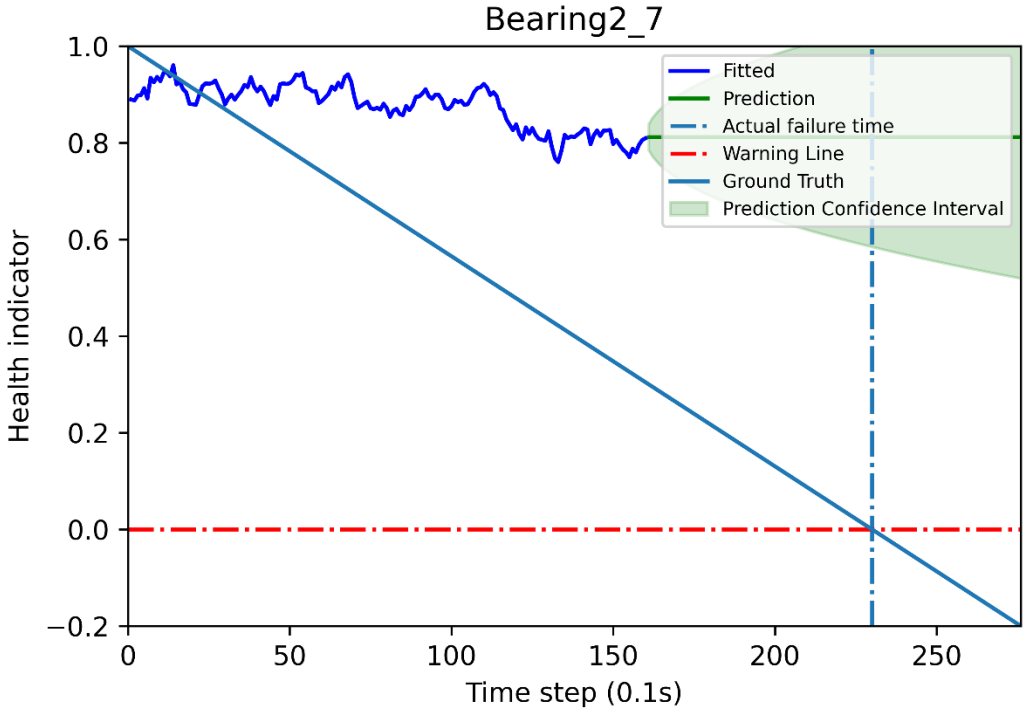


Figure B – 46 Bearing 2_7 RUL prediction

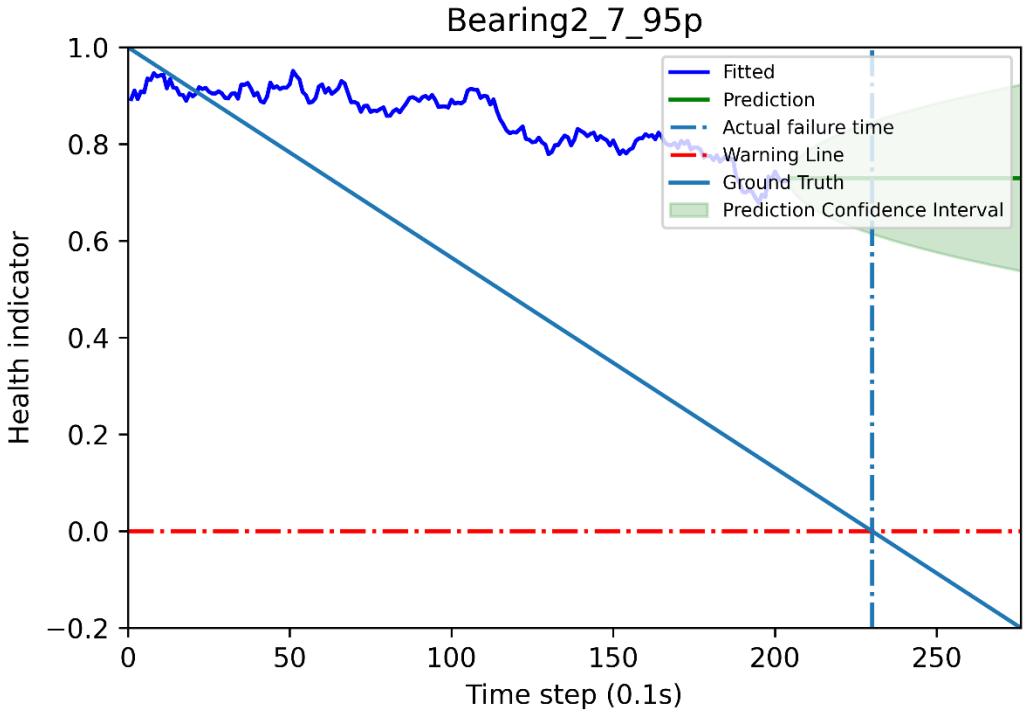


Figure B – 47 Bearing 2_7_95p RUL prediction

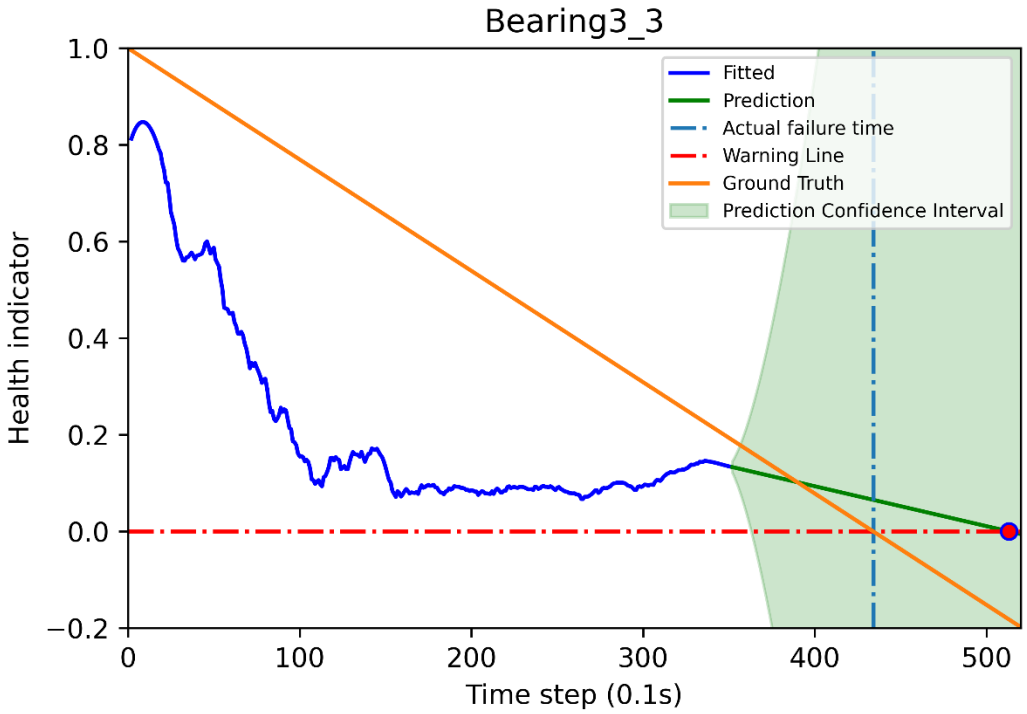


Figure B – 48 Bearing 3_3 RUL prediction

Appendix C - Diagnosis Codes

```
%========================== PRONOSTIA==========================
rpm = 1500;
sampRate = 25600;
file_num=20;

if file_num==1
    Vibdata = readmatrix("acc_02250.csv");
    rawData = Vibdata(:,6);
    titlename= "Bearing1.1-test";
    filename= "Bearing1.1-test";
else

    bearing="Bearing3.3";
    csv_num="--(410-430)--C2";
    filename= bearing+csv_num;
    [filenames, folder] = uigetfile('*.csv','Select the data
file','MultiSelect','on');   % gets directory from any folder
    if isnumeric(filenames); return; end   %user cancel
    filenames = cellstr(filenames);   %in case user only selected one
    fullnames = fullfile(folder, filenames);
    nfiles = length(fullnames);
    existingvars = {};
    combined = [];
    for K = 1 : nfiles
        thisfile = fullnames{K};
        thisdata = readtable(thisfile);
        thisdata=thisdata(:,6);
        %the variable names might already existing in another file. Every
        %variable name for a table must be unique.

        thisvars = thisdata.Properties.VariableNames;
        combinedvars = [existingvars, thisvars];
        U = matlab.lang.makeUniqueStrings(combinedvars);
        thisvars = U(length(existingvars)+1:end);
        thisdata.Properties.VariableNames = thisvars;
        existingvars = combinedvars;

        if K == 1
            combined = thisdata;
        else
            if height(thisdata) > height(combined)
                numnew = height(thisdata) - height(combined);
                E = combined(1,:);
                C = repmat({missing}, 1, width(E));
                C(table2array(varfun(@iscell, E))) = {{missing}};
                E(1,:) = C;
                E = repmat(E, numnew, 1);
                combined = [combined; E];
            elseif height(thisdata) < height(combined);
                numnew = height(combined) - height(thisdata);
                E = thisdata(1,:);
                C = repmat({missing}, 1, width(E));
```

```
                C(table2array(varfun(@iscell, E))) = {{missing}};
                E(1,:) = C;
                E = repmat(E, numnew, 1);
                thisdata = [thisdata; E];
            end
            combined = [combined, thisdata];

        end
    end
    combined=combined{:,:};
    rawData=combined(:);
end
%===========
fr=rpm/60;
n=13;
d=3.5;
D=25.6;
BPFO = ((n*fr)/2)*(1-d/D);
BPFI = ((n*fr)/2)*(1+d/D);
FTF = (fr/2)*(1-d/D);
BSF = 2*((fr*D)/(2*d))*(1-(d/D)^2);
bearFreq = [BPFO BPFI FTF BSF]; % BPFO, BPFI, FTF, BSF
maxP = round(sampRate/BPFI);
windLeng = [32 64 128 256];
%============================ PRONOSTIA============================

%==============Discrete signal separation (AR model) ==================
x=rawData(:);
N=length(x);
for p = 1 : maxP
  if rem(p,50)==0
      disp(['p=' num2str(p)]);
  end
  a = aryule(x,p); % aryule returns the AR model parameter, a(k)
  X = zeros(N,p);
  for i = 1 : p
      X(i+1:end,i) = x(1:N-i);
  end
  xp = -X*a(2:end)';
  e = x-xp;
  tempKurt(p,1) = kurtosis(e(p+1:end));
end
optP = find(tempKurt==max(tempKurt)); %==== Optimum solution
optA = aryule(x,optP);
xp = filter([0 -optA(2:end)],1,x);
e = x(optP+1:end) - xp(optP+1:end); % residual signal

%============Demodulation band selection (STFT & SK) ================
Ne = length(e);
numFreq = max(windLeng)+1;
for i = 1 : length(windLeng)
  windFunc = hann(windLeng(i )); %==== Short Time Fourier Transform
  numOverlap = fix(windLeng(i)*0.2);
  numWind = fix((Ne-numOverlap)/(windLeng(i)-numOverlap));
  n = 1:windLeng(i);
  STFT=zeros(numWind,numFreq);
```

```
  for t = 1 : numWind
      stft = fft(e(n).*windFunc, 2*(numFreq-1));
      stft = abs(stft(1:numFreq))/windLeng(i)/sqrt(mean(windFunc.^2))*2;
   STFT(t,:) = stft';
   n = n + (windLeng(i)-numOverlap);
  end
  for j = 1 : numFreq %==== Spectral Kurtosis
     specKurt(i,j) = mean(abs(STFT(:,j)).^4)./mean(abs(STFT(:,j)).^2).^2-2;
  end
  lgd{i} = ['window size = ',num2str(windLeng(i))];
end
figure(1) %==== Results
freq = (0:numFreq-1)/(2*(numFreq-1))*sampRate;
plot(freq,specKurt);
legend(lgd,'location','best')
xlabel('Frequency[Hz]'); ylabel('Spectral kurtosis'); xlim([0 sampRate/2]);
[freqRang] = input('Range of bandpass filtering, [freq1,freq2] = ');
title(filename);
set(gcf,'position',[0,0,700,400])
saveas(gcf,'D:\CNN\PRONOSTIA_diagnosis\Final_diagnosis\'+bearing+'\'+filename
+'_Spectral Kurtosis.png');
[b,a] = butter(2,[freqRang(1) freqRang(2)]/(sampRate/2),'bandpass');
X = filter(b,a,e); % band-passed signal

%======================Envelope analysis ============================
aX = hilbert(X); % hilbert(x) returns an analytic signal of x
envel = abs(aX);
envel=envel-mean(envel); % envelope signal
fftEnvel = abs(fft(envel))/Ne*2;
fftEnvel = fftEnvel(1:ceil(N/2));

%============subplot for 4 different faults============
subplot(2,2,1)
freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
hold on;
harmImpact = (1:150)*(bearFreq(1));
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,'-.k', 'LineWidth',1.1);
fprintf("BPFO");
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title('(a)BPFO','fontsize',11);

subplot(2,2,2)
freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
hold on;
harmImpact = (1:150)*(bearFreq(2));
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,'-.k', 'LineWidth',1);
plot(X+fr,Y,':r', 'LineWidth',1)
plot(X-fr,Y,':r', 'LineWidth',1)
fprintf("BPFI");
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title('(b)BPFI','fontsize',11);
```

```
hold off;
%
subplot(2,2,3)
freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
hold on;
harmImpact = (1:150)*(bearFreq(3));
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,'-.k', 'LineWidth',1);
fprintf("FTF");
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title('(c)FTF','fontsize',11);
hold off
%
subplot(2,2,4)
freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
hold on;
harmImpact = (1:150)*(bearFreq(4));
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,'-.k', 'LineWidth',1);
plot(X+FTF,Y,':r', 'LineWidth',1)
plot(X-FTF,Y,':r', 'LineWidth',1)
fprintf("BSF");
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title('(d)BSF','fontsize',11);
hold off;

sgtitle(filename,'fontsize',13);
set(gcf,'position',[0,0,800,500])
saveas(gcf,'D:\CNN\PRONOSTIA_diagnosis\Final_diagnosis\'+bearing+'\'+filename
+'.png');
clf
% %============MRF plot============
freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
hold on;
harmImpact = (1:150)*(fr);
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,':k', 'LineWidth',1);
fprintf("MRF");
lgnd=legend('Envelope spectrum','MRF');
lgnd.FontSize=9;
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title(filename);
hold off;
set(gcf,'position',[0,0,700,400])
saveas(gcf,'D:\CNN\PRONOSTIA_diagnosis\Final_diagnosis\'+bearing+'\'+filename
+'_MRF.png');
clf
%==================4 seperate faulty plots==============
freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
```

```
hold on;
harmImpact = (1:150)*(bearFreq(1));
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,'-.k', 'LineWidth',1);
fprintf("BPFO");
lgnd=legend('Envelope spectrum','BPFO');
lgnd.FontSize=9;
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title(filename);
set(gcf,'position',[0,0,700,400])
saveas(gcf,'D:\CNN\PRONOSTIA_diagnosis\Final_diagnosis\'+bearing+'\'+filename
+'_BPFO.png');
clf

freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
hold on;
harmImpact = (1:150)*(bearFreq(2));
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,'-.k', 'LineWidth',1);
plot(X+fr,Y,':r', 'LineWidth',1)
plot(X-fr,Y,':r', 'LineWidth',1)
fprintf("BPFI");
lgnd=legend('Envelope spectrum','BPFI');
lgnd.FontSize=9;
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title(filename);
set(gcf,'position',[0,0,700,400])
saveas(gcf,'D:\CNN\PRONOSTIA_diagnosis\Final_diagnosis\'+bearing+'\'+filename
+'_BPFI.png');
hold off;
clf

freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
hold on;
harmImpact = (1:150)*(bearFreq(3));
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,'-.k', 'LineWidth',1);
fprintf("FTF");
lgnd=legend('Envelope spectrum','FTF');
lgnd.FontSize=9;
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title(filename);
set(gcf,'position',[0,0,700,400])
saveas(gcf,'D:\CNN\PRONOSTIA_diagnosis\Final_diagnosis\'+bearing+'\'+filename
+'_FTF.png');
hold off
clf

freq = (0:Ne-1)/Ne*sampRate;
freq = freq(1:ceil(N/2));
stem(freq,fftEnvel,".", 'LineWidth',1.2);
hold on;
```

```
harmImpact = (1:150)*(bearFreq(4));
[X,Y] = meshgrid(harmImpact,ylim);
plot(X,Y,'-.k', 'LineWidth',1);
plot(X+FTF,Y,':r', 'LineWidth',1)
plot(X-FTF,Y,':r', 'LineWidth',1)
fprintf("BSF");
lgnd=legend('Envelope spectrum','BSF');
lgnd.FontSize=9;
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800]);
title(filename);
set(gcf,'position',[0,0,700,400])
saveas(gcf,'D:\CNN\PRONOSTIA_diagnosis\Final_diagnosis\'+bearing+'\'+filename
+'_BSF.png');
hold off;
%% ==============FTT and Hilbert alone====

rawData_1 = Vibdata_single(:,5);
titlename= "Bearing2.4--(735)";
filename= "Bearing2.4--(735)";

X=rawData_1(:);
% X=filtered_data(:);
X = abs(hilbert(X- mean(X)));
X = X - mean(X);
N=length(X);
NFFT = N;
enveloped_spectrum=fft(X, NFFT)/N;
F_index = sampRate/2 * linspace(0, 1, NFFT/2+1);
plot(F_index, abs(enveloped_spectrum(1: length(F_index))));
set(gca, 'xlim', [0, 800]);
freq = F_index;
fftEnvel = abs(enveloped_spectrum(1: length(F_index)));

subplot(3,1,1)
plot(freq,fftEnvel,"-", 'LineWidth',1.2);
xlabel('Frequency [Hz]'); ylabel('Amplitude'); xlim([0 800])
title('(a)Singal 0.1s of data','fontsize',11);
```

Appendix D - Prognosis Codes (Spectrogram_C1 only)

```
#Stage 1
from google.colab import drive
drive.mount('/content/drive/', force_remount=True)
!pip install keras-tuner --upgrade
import numpy as np
import scipy.signal as signal
from tqdm import tqdm
import matplotlib.pyplot as plt
import matplotlib
import math
from tensorflow import keras
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.callbacks import EarlyStopping
import keras_tuner as kt
def spectrogram_channel_1(sample):
    _, _, ch1_spec = signal.spectrogram(sample[:,0],
                                        fs=1.0,
                                        window=('tukey', 0.25),
                                        nperseg=22,
                                        noverlap=0,
                                        nfft=230,
                                        detrend='constant',
                                        return_onesided=True,
                                        scaling='density',
                                        axis=-1)

    ch1_spec = (ch1_spec - np.min(ch1_spec)) / np.ptp(ch1_spec)
    return ch1_spec
def dataset_prepare(bearing_names):
    X, Y = [], []

    [dataset, labels] = [np.load('/content/drive/My Drive/PRONOSTIA_Oridat
a/Linear_femto_data.npy', allow_pickle='TRUE').item(),
                         np.load('/content/drive/My Drive/PRONOSTIA_Oridat
a/Linear_femto_label.npy', allow_pickle='TRUE').item()]

    for bearing_name in bearing_names:
        data = dataset[bearing_name]
```

```
        data_size = list(np.shape(spectrogram_channel_1(data[:,:,0])))
        data_size.insert(0, np.shape(data)[-1])
        X_bearing = np.zeros(data_size)

        print("Processing " + str(bearing_name))
        for measurement_idx in tqdm(range(len(data[1,1,:]))):
            measurement = data[:,:, measurement_idx]
            X_bearing[measurement_idx,:,:] = spectrogram_channel_1(measure
ment)

        try:
            X = np.concatenate((X, X_bearing))
        except:
            X = X_bearing

        try:
            Y = np.concatenate((Y, labels[bearing_name][::-1]),0)
        except:
            Y = labels[bearing_name][::-1]

    return [X, Y]
train_bearings = ["Bearing1_1", "Bearing1_2",  "Bearing2_1", "Bearing2_2",
 "Bearing3_1",  "Bearing3_2"]
test_bearings = ["Bearing1_3", "Bearing1_5", "Bearing1_6" , "Bearing1_7" ,

                 "Bearing2_3", "Bearing2_4", "Bearing2_5", "Bearing2_6", "
Bearing2_7",
                 "Bearing3_3"]

[X_train, Y_train] = dataset_prepare(train_bearings)
[X_test, Y_test] = dataset_prepare(test_bearings)
print('X_train.shape:', X_train.shape)
print('Y_train.shape:', Y_train.shape)
print('X_test.shape:', X_test.shape)
print('Y_test.shape:', Y_test.shape)
X_train = X_train.reshape(X_train.shape + (1,))
X_test = X_test.reshape(X_test.shape + (1,))

X_train.shape
X_train_splits = []
Y_train_splits = []

for bearing in train_bearings:
  [X_train_temp, Y_train_temp] = dataset_prepare([bearing])
  X_train_temp = X_train_temp.reshape(X_train_temp.shape + (1,))
```

```
  X_train_splits.append(X_train_temp)
  Y_train_splits.append(Y_train_temp)
X_test_splits = []
Y_test_splits = []

for bearing in test_bearings:
  [X_test_temp, Y_test_temp] = dataset_prepare([bearing])
  X_test_temp = X_test_temp.reshape(X_test_temp.shape + (1,))

  X_test_splits.append(X_test_temp)
  Y_test_splits.append(Y_test_temp)
input_shape = X_train.shape[1:]
def build_model(hp):
  # hyper-parameter searching space
  lr = hp.Choice('lr', [0.001, 0.005, 0.01])
  cnn_filters_1 = hp.Choice('cnn_filters_1', [16, 32])
  cnn_filters_2 = hp.Choice('cnn_filters_2', [32, 64])
  cnn_filters_3 = hp.Choice('cnn_filters_3', [64, 128])
  fc_units_1 = hp.Choice('fc_units_1', [256, 128])
  fc_units_2 = hp.Choice('fc_units_2', [128, 64])
  kernel_size = hp.Choice('kernel_size', [3, 5])
  bn = hp.Boolean('bn')
  activation = "relu"

  # model architecture
  model = models.Sequential()
  model.add(layers.Conv2D(filters=cnn_filters_1, kernel_size=kernel_size,
strides=[1,1], activation=activation, input_shape=input_shape))
  model.add(layers.Conv2D(filters=cnn_filters_1, kernel_size=kernel_size,
strides=[1,1], activation=activation))

  model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(1,1)))
  if bn:
    model.add(layers.BatchNormalization(axis=-1))

  model.add(layers.Conv2D(filters=cnn_filters_2, kernel_size=kernel_size,
strides=[1,1], activation=activation))
  model.add(layers.Conv2D(filters=cnn_filters_2, kernel_size=kernel_size,
strides=[1,1], activation=activation))

  model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
  if bn:
    model.add(layers.BatchNormalization(axis=-1))
```

```
  model.add(layers.Conv2D(filters=cnn_filters_3, kernel_size=kernel_size,
strides=[1,1], activation=activation))
  model.add(layers.Conv2D(filters=cnn_filters_3, kernel_size=kernel_size,
strides=[1,1], activation=activation))

  model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
  if bn:
    model.add(layers.BatchNormalization(axis=-1))

  model.add(layers.Flatten())
  model.add(layers.Dense(fc_units_1, activation=activation))
  model.add(layers.Dense(fc_units_2, activation=activation))
  model.add(layers.Dense(1))

  model.compile(optimizer=optimizers.Adam(learning_rate=lr), loss="MSE", m
etrics=["MSE", "MAE", "MAPE", "MSLE"])
  return model
tuner = kt.RandomSearch(hypermodel=build_model,
                        objective="val_loss",
                        max_trials=30,
                        executions_per_trial=3,
                        overwrite=True,
                        directory="SearchResults",
                        project_name="Spectrogram_C1")
tuner.search_space_summary()
tuner.search(X_train, Y_train, epochs=5, validation_data=(X_test, Y_test))
tuner.results_summary()
# Check the best hyperparameters
best_hps = tuner.get_best_hyperparameters(5)

best_hp = best_hps[0]
print('lr = ', best_hp['lr'])
print('cnn_filters_1 = ', best_hp['cnn_filters_1'])
print('cnn_filters_2 = ', best_hp['cnn_filters_2'])
print('cnn_filters_3 = ', best_hp['cnn_filters_3'])
print('fc_units_1 = ', best_hp['fc_units_1'])
print('fc_units_2 = ', best_hp['fc_units_2'])
print('kernel_size = ', best_hp['kernel_size'])
# print('bn = ', best_hp['bn'])
best_models = tuner.get_best_models(num_models=2)
best_model = best_models[0]
best_model.build(input_shape=input_shape)
best_model.summary()
early_stopping_monitor = EarlyStopping(
    monitor='loss',
```

```
    min_delta=0,
    patience=30,
    verbose=0,
    mode='auto',
    baseline=None,
    restore_best_weights=True
)
history = best_model.fit(X_train, Y_train, epochs=200, validation_data=(X_
test, Y_test), callbacks=[early_stopping_monitor])
best_model.save('/content/drive/My Drive/ColabNotebooks/Pictures/test_el/S
pectrogram_C1_Linear(MSE)')
plt.figure(figsize=(15, 9))

for idx, bearing in enumerate(train_bearings):

  Y_pred_temp = best_model.predict(X_train_splits[idx])

  plt.plot(Y_pred_temp, label=bearing)
  if idx == len(train_bearings) - 1:
    plt.plot(Y_train_splits[idx], label='Ground Truth', color='black')
  else:
    plt.plot(Y_train_splits[idx], label='_nolegend_', color='black')

plt.xlabel("Time Step")
plt.ylabel("Remaining Useful Life")
plt.legend(loc='best')
title_name='Train Accuracy'
plt.title(title_name)
plt.rcParams['figure.dpi'] = 600
plt.rcParams['savefig.dpi'] = 600
plt.savefig('/content/drive/My Drive/ColabNotebooks/Pictures/test_el/TEST_
'+title_name+'.png')
plt.show()
```

Stage 2

```
test_bearings = ["Bearing1_3", "Bearing1_3_50p", "Bearing1_3_95p", "Bearin
g1_5", "Bearing1_6" , "Bearing1_7",
                 "Bearing2_3", "Bearing2_3_95p" , "Bearing2_4", "Bearing2_
4_50p", "Bearing2_4_95p", "Bearing2_5", "Bearing2_6",
                 "Bearing2_6_50p", "Bearing2_6_95p",
                 "Bearing2_7", "Bearing2_7_95p", "Bearing3_3"]
X_test_splits = []
Y_test_splits = []
```

```
for bearing in test_bearings:
  [X_test_temp, Y_test_temp] = dataset_prepare([bearing])
  X_test_temp = X_test_temp.reshape(X_test_temp.shape + (1,))

  X_test_splits.append(X_test_temp)
  Y_test_splits.append(Y_test_temp)
model = keras.models.load_model('/content/drive/My Drive/ColabNotebooks/Pi
ctures/test_el/Spectrogram_C1_Linear(MSE)')
Y_test_preds = []
for idx, bearing in enumerate(test_bearings):
  Y_pred_temp = model.predict(X_test_splits[idx])
  Y_test_preds.append(np.array(Y_pred_temp).squeeze())
from scipy.signal import savgol_filter
image_num = len(test_bearings)
col = math.ceil(math.sqrt(image_num))
if col * (col - 1) < image_num:
  row = col
else:
  row = col - 1
# Full_test=[2375, 2463, 2448, 2259, 1955, 751, 2311, 701, 230, 434]
Full_test=     [2375, 2375, 2375, 2463, 2448, 2259, 1955, 1955, 751, 751,
751, 2311, 701, 701, 701, 230, 230,434 ]
split_points = [1801,1188, 2256, 2302, 2302, 1502, 1202, 1858, 612, 375, 7
14, 2002, 572, 350, 666, 172, 219, 352]
file_name=['SCI']
Channel=['_C1']
# for singal plots
# row=2
# col=2
y_est= []
for i_row in range(row):
  for i_col in range(col):
    i = i_row * col + i_col
    if  i < image_num:
      t = np.linspace(0, len(Y_test_preds[i]), len(Y_test_preds[i]))
      if file_name[0] == 'Smooth':
      # Smooth
          y_est_temp = smooth(np.squeeze(Y_test_preds[i]),int(0.05*max(t))
)
          plt.plot(t,smooth(np.squeeze(Y_test_preds[i]),int(0.05*max(t))))

      # SCI
      else:
```

```
            y_est_temp = savgol_filter(np.squeeze(Y_test_preds[i]), 4*int(0.
025*max(t))+1, 3) # window size 51, polynomial order 3
            plt.plot(t,y_est_temp)

        y_est_temp.tolist()
        y_est.append(y_est_temp)

        # plot individually
        plt.axhline(0,ls='-.', color='red', label='Warning Line')
        plt.plot(t, Y_test_preds[i], 'o', color='tab:blue', markersize=1, al
pha=0.15, label='Prediction')

        x=np.arange(0, Full_test[i],1)
        plt.plot(x, -x/(Full_test[i])+1, '-', label='Ground Truth')
        plt.xlim(0,int(1.2*(Full_test[i])))
        plt.ylim(-0.2,1)

        plt.axvline(Full_test[i],ls='-.',label='Actual failure time')

        plt.xlabel('Time Step (0.1s)')
        plt.ylabel('Remaining Useful Life')
        plt.title(test_bearings[i])
        plt.legend(loc='upper right', fontsize=7)
        plt.rcParams['figure.dpi'] = 600
        plt.rcParams['savefig.dpi'] = 600
        plt.savefig('/content/drive/My Drive/ColabNotebooks/Pictures/'+file_
name[0]+Channel[0]+'/Stage_1/'+test_bearings[i]+'.png')
        plt.show()

if file_name[0] == 'Smooth':
    np.save('/content/drive/My Drive/ColabNotebooks/Pictures/'+file_name[0
]+Channel[0]+'/Smooth_y_est.npy', y_est, allow_pickle=True)
else:
    np.save('/content/drive/My Drive/ColabNotebooks/Pictures/'+file_name[0
]+Channel[0]+'/SCI_y_est.npy', y_est, allow_pickle=True)
if file_name[0] == 'Smooth':
    y_est=np.load('/content/drive/My Drive/ColabNotebooks/Pictures/'+file_
name[0]+Channel[0]+'/Smooth_y_est.npy', allow_pickle='TRUE').tolist()
    for i_row in range(row):
      for i_col in range(col):
        i = i_row * col + i_col
        if  i < image_num:
          y_est[i]=y_est[i][int(len(y_est[i])*0.05):-
int((len(y_est[i])*0.02))]
else:
```

```
    y_est=np.load('/content/drive/My Drive/ColabNotebooks/Pictures/'+file_
name[0]+Channel[0]+'/SCI_y_est.npy', allow_pickle='TRUE').tolist()
from skforecast.ForecasterAutoreg import ForecasterAutoreg
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
arima_y_trains = []
arima_y_tests = []

arima_fh_trains = []
arima_fhs = []

arima_y_preds = []
arima_y_fits = []

arima_y_pred_ints = []
arima_y_fits_ints = []

arima_maes = []
arima_mses = []
arima_mapes = []

arima_forecasters = []
arima_params = []

past_series=y_est

for i in range(len(past_series)):
  y_train = Series(past_series[i].squeeze(), index = list(range(0, len(pas
t_series[i].squeeze()))))
  fh = np.arange(1, int(10*len(y_est[i])))

  forecaster = AutoARIMA(suppress_warnings=True)
  forecaster.fit(y_train)
  y_pred = forecaster.predict(fh)
  y_pred_int = forecaster.predict_interval(fh, coverage=0.95)

  fh_train = ForecastingHorizon(y_train.index, is_relative=False)  # in-
sample forecasting horizon
  y_fit = forecaster.predict(fh_train)
  y_fit_int = forecaster.predict_interval(fh_train, coverage=0.95)

  arima_y_trains.append(y_train)
  arima_y_preds.append(y_pred)
  arima_y_fits.append(y_fit)
  arima_fh_trains.append(fh_train)
```

```
    arima_fhs.append(fh)

    arima_y_pred_ints.append(y_pred_int)
    arima_y_fits_ints.append(y_fit_int)

    arima_forecasters.append(forecaster)
    arima_params.append(forecaster.get_fitted_params())
for i, params in enumerate(arima_params):
    print('\nParameters of ' + str(i) + 'th ARIMA model:')
    print(params)
image_num = len(test_bearings)
col = math.ceil(math.sqrt(image_num))
if col * (col - 1) < image_num:
    row = col
else:
    row = col - 1
# for singal plots
ACTRUL=[]
for i_row in range(row):
    for i_col in range(col):
      i = i_row * col + i_col
      if  i < image_num:
        plt.xlim(0,int(1.2*(Full_test[i])))
        plt.ylim(-0.2,1)
        plt.plot(arima_y_fits[i], color='blue', label="Fitted")
        plt.plot(arima_y_preds[i], color='green', label="Prediction")
        # plt.axhline(0,ls='-.', color='red', label='Warning Line')
        plt.axvline(Full_test[i],ls='-.',label='Actual failure time')

        x=np.arange(0, len(np.array(arima_y_preds[i])),1)
        y=np.zeros(len(np.array(arima_y_preds[i])))
        plt.plot(20*x,y,ls='-.', color='red', label='Warning Line')


        plt.plot(x, -x/(Full_test[i])+1, '-', label='Ground Truth')

        plt.fill_between(np.array(arima_y_preds[i].index),
                         np.array(arima_y_pred_ints[i], dtype=float)[:,0],
                         np.array(arima_y_pred_ints[i], dtype=float)[:,1],
                         color='green', alpha=0.2, label='Prediction Confid
ence Interval')

        plt.xlabel('Time step (0.1s)')
        plt.ylabel('Health indicator')
        plt.title(test_bearings[i])
```

```
      plt.legend(loc='upper right', fontsize=7)
      plt.rcParams['figure.dpi'] = 600
      plt.rcParams['savefig.dpi'] = 600
      plt.show()
Full_test= np.array([2375, 2375, 2375, 2463, 2448, 2259, 1955, 1955, 751,
751, 751, 2311, 701, 701, 701, 230, 230,434 ])
Split = np.array([1801,1188, 2256, 2302, 2302, 1502, 1202, 1858, 612, 375,
 714, 2002, 572, 350, 666, 172, 219, 352])
ACTUAL_RUL=Full_test-Split
ACTUAL_RUL
Pre_RUL=[]
for element in ACTRUL:
   Pre_RUL.append(element*-1)

np.save('/content/drive/My Drive/ColabNotebooks/Pictures/'+file_name[0]+Ch
annel[0]+'/Pred_RUL.npy',Pre_RUL)
error=ACTUAL_RUL-Pre_RUL
pe=error/ACTUAL_RUL
def score(pe):
  As = []
  for element in pe:
    if element <= 0:
      A = np.exp(-np.log(0.5)*(element/5))
    else:
      A = np.exp(np.log(0.5)*(element/20))
    As.append(A)
  return np.mean(As)
score(pe)
```