POLITECNICO DI TORINO

Master's Degree in Mechanical Engineering

Master's Degree Thesis

Evaluation of bearing damage through Signal Processing and Artificial Intelligence

Supervisors

Prof. Cristiana DELPRETE

Prof. Eugenio BRUSA

Dott. Luigi Gianpio DI MAGGIO

Candidate

Farid JAFARI

Feb 2022

Summary

Every system eventually fails unless it is maintained regularly. To this end each entity might follow a different maintenance procedure to maximize reliability and reduce expenses. There are in general 3 procedures for maintenance. One is so called Reactive Maintenance where the action is taken only after the machine fails. This might make sense if the system is not crucial or safety related. But if a complex and expensive system is being dealt with, Reactive method might cause serious damage and costs and more importantly safety issues. To avoid such consequences, many companies perform regular controls called Preventive Maintenance. One big problem with this method is defining when the part replacement must be done and since we don't know when the Failure will happen, additional costs will be imposed as we might waste machine's useful life. So, if one can predict the failure time, the maintenance can be performed right before that point. This way, the problems mentioned before can be easily tackled.

Predictive maintenance (PM) is now a major sector in the industry. Its benefits now has urged companies and academics to invest in this field which has resulted in considerable advancements. Furthermore, the arrival of the Internet of Things necessitates automation of predictions. Among all, bearings are of the most important parts of any machine and their failure accounts for a major part of dangerous and expensive damages and therefore are exclusively critical. Among different methodologies, recent studies discovered the strength of the Artificial intelligence (AI) in yielding highly accurate estimations of Remaining Useful Life (RUL) of bearings. Researchers have examined different sub-sects of the AI such as Machine Learning (ML) in particular Deep Learning (DL) and developed very precise Health Indicators (HI). In Machine Prognostics, fault feature selection can be complex. DL methods help us overcome this difficulty by automatising the feature extraction process. Deep learning (DL) methods provide effective solutions to surpass the limitations because of their strong feature learning ability. Deep Networks have multiple hidden layers which have the ability to learn hierarchical representations from the raw data directly . Through training procedure, deep Network are able to automatically identify meaningful hierarchical representations for accurate predictions. DL is now being practically applied to many areas of

science and technology.[1] Despite huge leaps in Prognostic and Health Management (PHM) techniques, We are still far from perfect. In this study, we try to improve the accuracy of data-driven methods for prognostics starting from real world vibrational data from bearings, processing them and creating an optimum HI through DL methods.

Acknowledgements

I would like to thank my supervisor Dott. Luigi Gianpio Di Maggio for his generous help. He was always obliging with any information and support required.

Deep Learning training process is naturally a process of trial and error. In this research, We have tried many different Spectrogram sets, normalizations and options but only a few of them are mentioned in order to avoid being verbose.

To Myself

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Predictive maintenance is a set of techniques that enables us to determine the health state of the equipment and machinery which is very useful to estimate when replacements should be done. Knowing exactly when to perform the maintenance contributes to huge cost savings and prevents serious damages to the facility. Machinery condition is evaluated through non-destructive testing like Vibrational analysis, Infrared testing, Thermal imaging, Oil analysis and other online testing methods.[2]

On the other hand, since the notion of the Internet of things is becoming more and more serious, sensor based predictive maintenance is becoming more and necessary but it is important to find a way to define the Remaining Useful Life (RUL) of the machines with no human interaction. Today, vibrational data collection is a very efficient method to evaluate the condition of the high-speed rotatory machines and engineers take advantage of high end accelerometers to collect important data. When correctly acquired, such data can discover exact health status of the equipment.

Ball and roller bearings are widely utilised in the industry and are one of the main parts of the machines, and the bearing failure is one of the usual causes of rotatory machine failure. According to Electric power Research Institute, bearing failure accounts for 50 percent of asynchronous motor failures.[3] [4]

Traditionally, system prognostics and health management (PHM) was based on conventional data and in our case signal processing techniques requiring highly skilled engineers to interpret the data. Furthermore, some meaningful data might be hidden to the naked eye. To resolve the two difficulties, scientists have proposed different automated procedures. Since the Artificial Intelligence (AI) is introduced, its uses and power in different fields of science and industry are being discovered. In this study I am going to take advantage of Machine Learning(ML), specifically Deep Learning (DL), to improve the estimation of RUL of bearings.

Deep learning (DL) is a subgroup of machine learning that consists multiple nonlinear layers originated from artificial neural network (ANN). As a result of the rapid growth of computational capacities, DL's potential power can be put into practice, in this case in the field of prognostics. DL is capable of extracting the hierarchical trends hidden in the structures. In DL multiple layers carefully placed in the network, capture the desired information from the raw input data. For this, DL models have become popular for their remarkable achievements in diverse fields of science, like image and speech recognition, natural language analysis. In the prognostics and the RUL prediction, the potentials of DL have not been fully discovered. So far the literature on DL focuses on the four deep architectures including Deep Belief Network, Convolutional Neural Network (CNN), Auto Encoder and Recurrent Neural Network.[2].

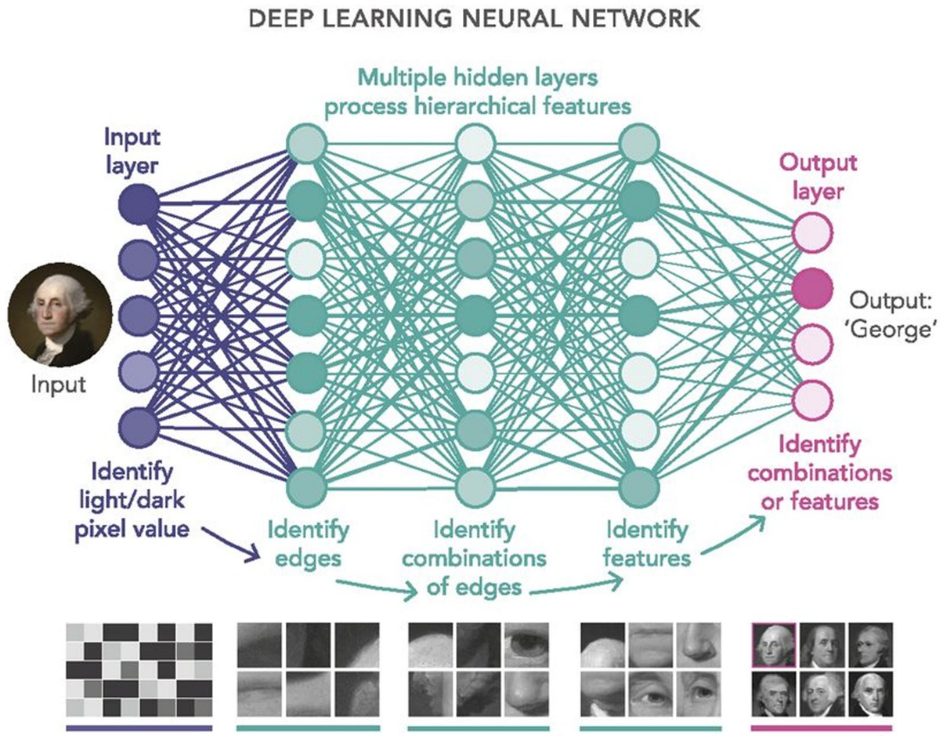

Figure 1.1: Schematic of a Deep Neural Network [5]

The performance of sensor based prognostic methods is highly dependant on the correct feature extraction in order to have meaningful start to failure trends. Deep Neural Networks (DNN) extracts the features automatically with no human interaction once the network is architectured, images are well normalized and labelling is done.

Moreover, recent papers show that Transfer Learning (TL) methods, i.e. using the already architectured Deep Neural Networks (DNN) for the training process saves a great amount of time and yields surprisingly good results.

In order to achieve acceptable estimations, DNN needs a massive amount of data. In this study we have used the IMS datasets provHerided by the centre for Intelligent Maintenance Systems, which is so far the most realistic start to failure dataset. The dataset consists of time domain .CSV format files acquired from accelerometers.

In this research instead of creating an original network architecture we are going to take advantage of networks already created (Transfer Learning) and powering our applications with it. Among all famous networks, We have decided to use ALexnet [6] which was initially designed for object classification.

Alexnet [6] is one of the most influential research papers which is published in the field of computer vision. it has motivated many other researches utilising CNN with use of GPUs to enhance the learning procedure. Alexnet's incredible accuracy on complex datasets, urged us to inspect its power as a base of our transfer-learning process. It is important to know that by removing any of the convolutional layers the performance will considerably decrease. This is also tested in our study. Today, Alexnet is the most used architecture in object-detection projects and computer vision sector. According to studies [7] Alexnet might be even utilized more than CNNs for image tasks. Alexnet is an image classifier network. Image classification is used to identify and predict the class of an object and its features in a group of pixels in an image [8]. In this thesis our work is based on spectrograms which are practically images. These images will be used for the learning process of the CNN.

The input of DNN is specifically an image so somehow the time domain data acquired has to be turned into images . Some studies [9] [10] have used time domain graphs as an input for the DNN training but time domain has little meaningful data for training process and might not yield accurate predictions. In this study we have used spectrograms as a powerful and meaningful input for the training process which is a 3D time and frequency domain image. In this thesis we used the short-time Fourier transform (STFT) to create such images (Spectrograms) but as one of the trials we have also made a unique dataset made by Continuous Wavelet Transform method to obtain the aforementioned time-frequency images.

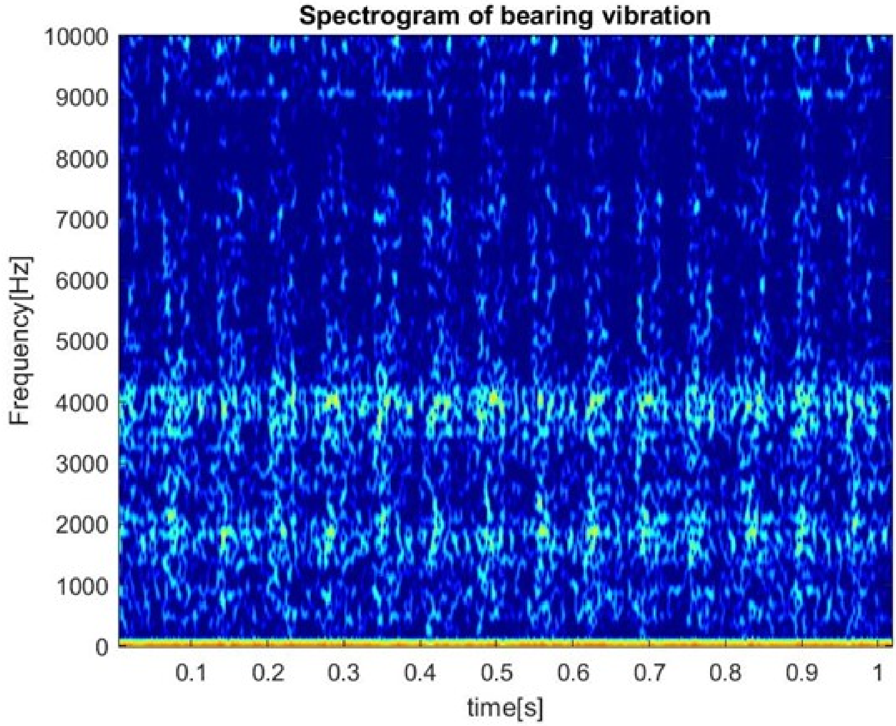

Figure 1.2: A Spectrogram created by short time Fourier transform technique

1.1 Literature Review

The notion of, Predictive Maintenance (PM), was first proposed in the late 80's, where online sensor data was analysed by the engineers to evaluate the health state of the system.[11] Machine learning on the other hand studies computer algorithms that are able to improve automatically over time. Machine learning is a major branch of Artificial Intelligence(AI). Machine learning was first introduced by Arthur Samuel, an American IMB employee in 1959. This literature review will concurrently explain the advancement in Prognostics and Health Management and Deep Learning as main subjects of this thesis.

In 2001 a research by Wenbin Wang [12] Demonstrated that the RUL is not simply a function of the service time but also the condition info received from the sensor. If put in practice, this finding can lead to a big reduction of part replacement and stocking expenses. The following year the same author published the "remaining life prediction of a set of rolling element bearings using the monitored vibration signals on the basis of a chosen distribution".[2]

These two works are particularly important because in them idea of Health Stage (HS) division is discussed and in the first one the finding the first faulty signal place is researched.

In 2006 Robert X.Gao and Ruqiang Yang[13][14] published a paper on transient nature of bearing vibrations. In this research they claim that signals generated by transient vibrations in rolling bearings due to structural defects are non-stationary in nature, and reflect upon the operation condition of the bearings and that non-stationary signals is critical to bearing health monitoring. This research introduces four representative time-frequency analysis methods used for non-stationary signal processing.

Next in 2008, a research "Feature extraction for bearing Prognostics and Health Management (PHM)" [15][16] was published Weizhong Yan, Hai Qiu, and Naresh Iyer as a guideline to perform a correct feature extraction procedure and choosing the correct methods to that end. In the mentioned paper, it is stated that with PHM we are allowed to use different sensors to sense different variables of a bearing including Vibration, Temprature, Chemical and acoustical emission and sound pressure sensors.

The Alexnet Convolutional neural network (CNN) was introduced in 2012 by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton [6]. Since then, the usage of deep convolutional neural network has increased exponentially, inasmuch it has leveraged several machine learning solution [17] [18]. AlexNet is regarded as one of the most influential papers published in computer vision, in that it has inspired many more papers published employing CNNs and GPUs to accelerate deep learning; according to Google Scholar, the AlexNet paper has been cited over 80,000 times [19].

A very simple Wavelet analysis technique using Continuous Wavelet Transform (CWT) that does not require complex signal processing knowledge was proposed by Sunil Tyagi1 , SK Panigrahi [4] [20] in 2015. They compare the study of Envelope Detection (ED) effectiveness, and CWT method for bearing fault diagnosis, showing that the proposed simple CWT based method is better fault diagnostic tool for bearing fault identification. They suggested applying a band-pass filter, which removes the large low-frequency components as well as the high frequency noise only the burst of high frequency vibrations remains and then FFT of this enveloped signal is taken, to obtain a frequency spectrum.

In "Bearing remaining useful life estimation based on time–frequency representation and supervised dimensionality reduction" a new approach was proposed by Minghang Zhao Baoping Tang QianTan[17] [20] in May 2016.

Traditionally, system prognostics and health management (PHM) depends on sufficient prior knowledge of critical components degradation process in order to predict the remaining useful life (RUL). However, the accurate physical or expert models are not available in most cases. In 2017, XiangLi, QianDing and Jian-QiaoSun [13] simple proposed a novel data-based method for prognostics utilizing deep convolution neural networks (DCNN). In this research raw collected data with normalization are directly used as inputs to the proposed network, and no prior expertise on prognostics and signal processing is required, that facilitates the application of the proposed method.

Also in 2017, Yaguo [21] systematically reviewed the data acquisition to RUL estimation procedure. In this paper both FEMTO[22] from a PRONOSTIA platform and IMS datasets provided by the center for Intelligent Maintenance Systems, are described and analysed. In the following graphs a start to failure time domain graph of one bearing in each dataset is shown:

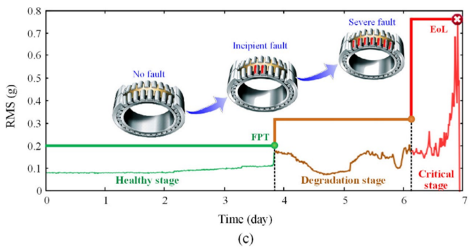

Figure 1.3: Time domain graphs of IMS [15]

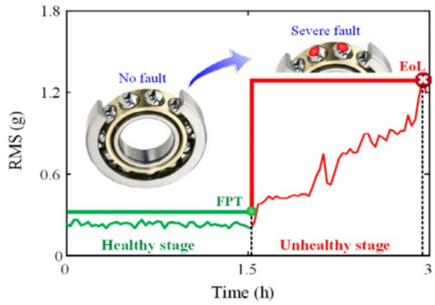

Figure 1.4: Femto Datasets [15]

Then, commonly used HI construction approaches and metrics are discussed. After that, the Health Stage HS division process is summarized by introducing its major tasks and existing approaches. Afterwards, the advancements of RUL prediction are reviewed including the popular approaches and metrics. Finally, the paper provides discussions on current situation, upcoming challenges as well as possible future trends for researchers in this field.

For the prediction of Remaining Useful Lifetime (RUL) of turbofan engine using machine learning, Vimala Mathew et al [23]. used ten machine learning algorithms for comparing the prediction accuracy in 2017's IEEE International Conference. The different algorithms were compared to obtain the prediction model having the closest prediction of remaining useful lifecycle in terms of number of life cycles. Next, in the same year, a TL-based approach for bearing fault diagnosis, where auxiliary data are transferred to improve diagnostic performance among various operating conditions was introduced by Shen[6] [24].

Another research called "A novel time–frequency image feature to construct HI and predict the RUL Based on Continuous Wavelet Transform and Convolutional Neural Network" was proposed by Youngji Yoo and Jun-Geol Baek on 1 June 2018[25] . In this research the proposed method is validated using IMS bearing dataset provided by PRONOSTIA which is also used in this thesis.

Again in the same year, a paper published by Siyu Shao et al [26]. called Highly Accurate Machine Fault Diagnosis Using Deep Transfer Learning , used transfer learning with CWT spectrograms to classify machine gearbox faults with near 100 percent accuracy.

So far, to our knowledge, no researcher has used Alexnet as a pretrained network for transfer learining purposes to estimate RUL of the bearing. This research aims to improve the estimation of remainig useful life of the bearings, trying different normalizations and methodologies. Also, in this research we base our training on the IMS bearing data set in which the bearings experienced longer and more complicated degradation processes than those in the former datasets, which increases the reality of this dataset as well as the difficulty of RUL prediction.

1.2 Work Organization

In the next chapters, we will introduce the step taken forward to achieve the RUL and HI and then we will briefly go through Deep Learning(DL), Convolutional Neural Networks (CNN), Classification and regression problem plus Transfer Learning (TL) and its benefits. Next, We will introduce the problem, its settings the acquisition mode and the database and its features to give a clear idea of the issue. Afterwards, the signal processing procedure, the concept of spectrograms, the steps taken in order to create and normalize them is explained. Then the procedure of transferring the network ,different training option concepts and optimum training options are presented. Finally different training results are shown and analysed comparing to actual experimental results. The results will be then used to reach a conclusion.

Chapter 2

Resources

2.1 RUL prediction

Unlike Reactive and Preventive Maintenance procedures, Predictive maintenance allows us to schedule the maintenance in an optimum time, but the Key to it is knowing the failure time as accurate as possible. If we can predict when the failure will happen, it is possible to schedule maintenance right before it. look at figure 2.1.

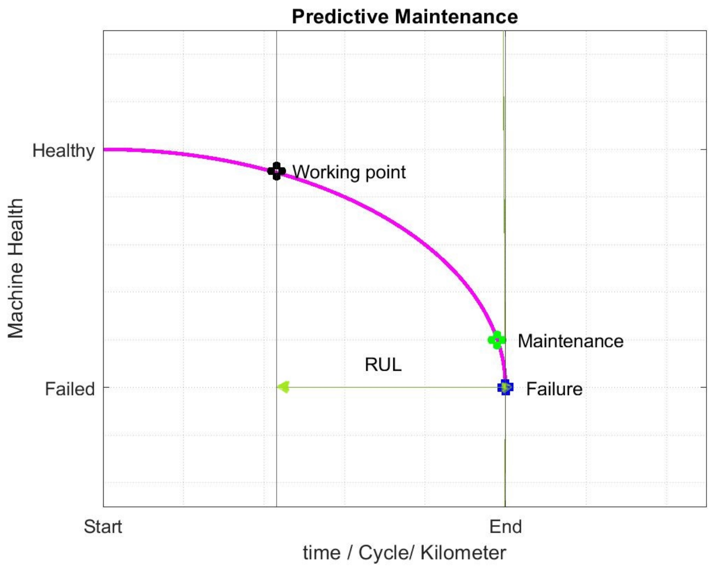

Figure 2.1: RUL graphical description

The Remaining Useful Life (RUL) is, the time between the working point and the time machine fails. Based on our system, the RUL might be represented as seconds, kilometers, cycles etc... . Depending on how much information we have on the degradation of the machinery, our approach to estimate the RUL will differ:

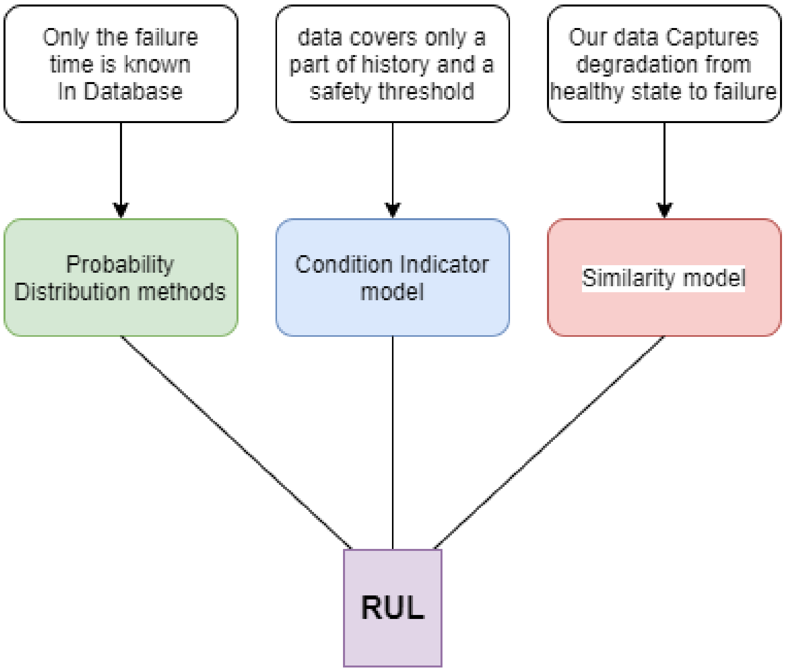

Figure 2.2: RUL estimation approaches

Sometimes, there is no history of machine degradation recorded but we know the the intervals or the time machine has failed. If this is the case, we can benefit from statistical and probability distribution methods to define the RUL.

Some other times, our data covers a part of machine's life history and we have a threshold beyond which it will be no longer safe to use the machine. In this case, a degradation model up to that point can be fit into a condition indicator that uses past information to tell how the condition indicator will change in future.

Our dataset though, covers a run-to-fail history of the system, which means that it includes the data from when the system is healthy, when it is degrading and when it fails. In this case we shall follow a similarity model based approach in which the RUL is estimated based on the similar system's history under similar conditions.

In general the flow that must followed to achieve RUL starts with data acquisition. As a strong base we need a large set of data acquired by the sensors collecting a set of healthy and faulty signals from the machine. The longer and the more diverse under which you machine works during the data collection the more robust our model will be. The reason why we implemented Transfer Learning in this thesis is lack of useful data which makes it impossible to follow normal training procedure and when possible, it often leads to over-fitted models.

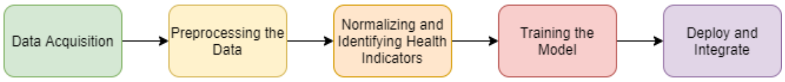

Figure 2.3: RUL estimation workflow

Next is to preprocess the data in a way health indicators are manifested. Preprocessing includes techniques such as noise, outlier, and missing value removal or changing the domain data is represented in the first place. In our case for example, we have moved from time domain to a 3D time-frequency (spectrogram) domain to extract the health indicators. In the next section the feature extraction process is discussed in details.

When the Preprocessing of the data is done, the condition indicators have to be identified. Condition indicators are the features that change in a meaningful way as the machine degrades. In deep learning (DL) this step is automatically done by the filters in each layer in the network.

So far, we have extracted the features that discriminates healthy data from faulty ones, so the data is ready to be fed into the training model. The AI, based on the meaningful features, trains a model upon which it can tell the health state of the system and its remaining useful life.

2.1.1 Feauture Extraction

As mentioned before, correct training and eventually prediction of RUL is strongly dependant on extraction of meaningful features. Such features must somehow represent the health state of the machine by either growing or disappearing linearly or nonlinearly as the machine keeps on working toward failure. In the following graph a complete map of feature extraction methods can be seen:

Stationary Signal

Acquired signals can be either stationary or transient. Unlike stationary signals the mean value and statistical properties of a transient signal is time variant. Almost all real world signals are non-stationary because bearings are inherently dynamic parts since loads and speed change over time. For computation ease specially in short time windows such signals are considered stationary. To analyse stationary signals there are time and frequency domain methods but Time-domain features are usually good for fault detection, and less efficient for classification, which is to determine where the defect is located, inner race, outer race, rolling elements, and cage. For fault classification, frequency-domain features are better.

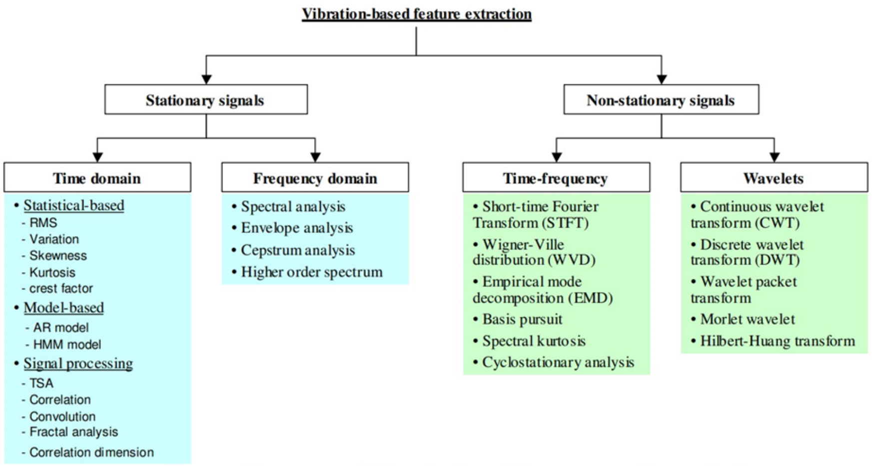

Figure 2.4: Taxonomy of vibration-based feature extraction methods [3]

Frequency-domain analysis include spectral analysis, envelope analysis, cepstrum, and higher-order spectra. [3]

Non-Stationary Signal

In such condition since the statistical properties are time variant, prior methods become ineffective. Techniques for analysing transient signals are time-frequency and wavelet analysis. In this research our focus is on the time-frequency method. Time-frequency analysis: Time-frequency analysis techniques analyze signals in both time and frequency simultaneously for identifying time-dependent variations of frequency components within the signal, which makes time-frequency analysis techniques a powerful tool for analyzing non-stationary signals. The most commonly used time-frequency analysis techniques are the short-time Fourier transform (STFT), the Wigner-Ville distribution, and the wavelet transform. Other newly developed time-frequency analysis techniques include spectral kurtosis, empirical mode decomposition, and cyclostationary analysis.[3]

Short-time Fourier transform (STFT)

STFT tackles non-stationary signals by applying the conventional FFT to a sliding window of the signal, which can be assumed to be locally stationary. The squared magnitude of the STFT, often referred as the spectrogram, provides the energy density spectrum of the signal as a function of time. Time resolution is determined by segment length. Thus the success of STFT is hinged on properly choosing window length, which often time is difficult. In this research we have tried different spectrograms with different window lengths to gain the optimum length. Using

STFT for bearing monitoring and diagnosis have shown in many publications, for example.

The STFT employs a sliding window function g(t) centered at time W to perform 'timelocalised' Fourier transform of the signal x(t) consecutively, and the result reveals variation of the signal's frequency content as time evolves, as illustrated in the following figure:

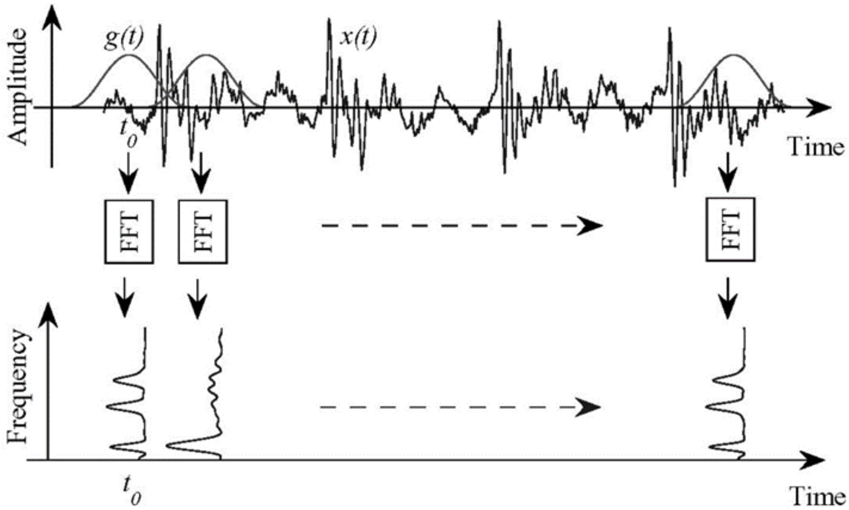

Figure 2.5: Illustration of short-time Fourier transform on the test signal x(t). [13]

Wavelet Analysis

Waveletes are used to apply a multiresolution analysis of any vibrational signal. This consists of applying a cascade of adjacent bandpass filter to the time domain signal. Wavelet analysis enables us to assess the signal content in different frequencies. In our case, an increase in the higher frequency signal's energy might indicate an early spall or lubrication faults. Performing a Multiresolutional analysis helps us in detecting the frequencies related to the faults.Periodic impacts of a defected bearing cause the defect frequencies. Such impacts transfer energy in a wide band of resonance frequencies. Given that in multiresolutional analysis we do not lose information, also the time domain analysis and features might be applied to the data in a desired resolution. Since each fault's contribution is often different in each frequency, Wavelet method's ability to decompose signal to its components in changing frequencies gives us the possibility to discriminate multiple fault types.[3]

Health Prognostics (HP) in Condition based maintenance plays a major role. That is predicting the RUL of the components relying on the degradation trends observed from previous tests. (HP) in machinery is often composed of four technical steps. First the data acquisition and then construction of a health indicator (HI). Next the healthstage (HS) division and finally the estimation of the RUL. Initially, in ,our case the vibrational data is acquired from accelerometers to monitor the condition of the bearings. Next, from the acquired data, HI is constructed by utilizing one of the common methods in order to represent the degradation trend of the bearings. After HI is constructed then the life history of the bearing is devided into desired stages (i.e. Health Stages). In the HSs finally, which discovers an obvious and sensible degradation trend, we can estimate the RUL by analysis of the trend and setting a failure threshold. In my thesis we will firstly skip the HS division as we benefit from DNN power to see the outcome and if the results aren't satisfying we will go for the HS division. [15]

2.2 Deep Learning

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, graph neural networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analogue. The idea of Artificial Neural networks (ANN)was initially inspired by how the information is processed and distributed in brain. However ANN has several differences from biological brains. Above all, Neural Networks are more static and symbolic while brains are mostly dynamic and analogue.

The word "deep" in deep learning refers the number of layers extracting different features one after another. We now know that a linear perceptron cannot be utilised as a universal classifier but a network with nonpolynomial activation function equipped with one hidden layer of unbounded width can. In ML , perceptron is a kind of algorithm used in supervised learning of binary classifiers. Binary classifiers are functions that are able to decide if an input, represented by a vector of numbers, belongs to some specific class [27]. These concepts will be explained in detail later. Deep learning is more concerned with an unbound number of layers of bounded size, that allows us to use it practically. In DL the layers are allowed to be heterogeneous and to differ widely from biological models to be efficient, trainable and understandable.

2.3 Basic Theory of CNN

Convolutional Neural Network (CNN) is a sect of deep learning that works in a similar manner to brain's visual cortex. We use CNN in Fields such as computer vision, Speech Recognition and language processing since it can receive raw images without complex pre-processing steps.[28] Each typical CNN is constructed by 3 layer groups: Convolutonal layers, Subsampling (Pooling) layers and Fully connected layers. In order to extract features from image data we alternatively place the convolutional and subsampling layers and place fully conneted layers as the output layer. Each convolutional network receives the previous layer's output and co-involves it with multiple kernels and pass it to an activation function to create a feature map. This feature map is the result of other input maps and is calculated as shown below:

$$X_j^l = f(\sum X_i^{l-1} * k_{ij}^l + b_j^l) \tag{2.1}$$

where * is a convolutional operator, xi is ith input map, k is a S*S convolutional kernel, bj is an additive bias, Mj is a feature map of the convolutional layer, and l is the lth layer in the network. In the End, the convolutional outputs are passed to the activation functions of each layer, f. The two most used activation functions (i.e., rectified linear unit (ReLU) and sigmoid function) are defined as follows:

$$ReLU(x) = max(0, x) \tag{2.2}$$

$$sigmoid(x) = 1/(1 + e^{-x}) \tag{2.3}$$

The subsampling layer which located after each convolutional layer exists to downsample the output of a convolution layer along both the spatial dimensions of height and width. It creates low-resolution maps by extracting the most significant local features. The most common subsampling layes is max-pooling layer that extracts the maximum value from each region, as shown in Figure 2.6:

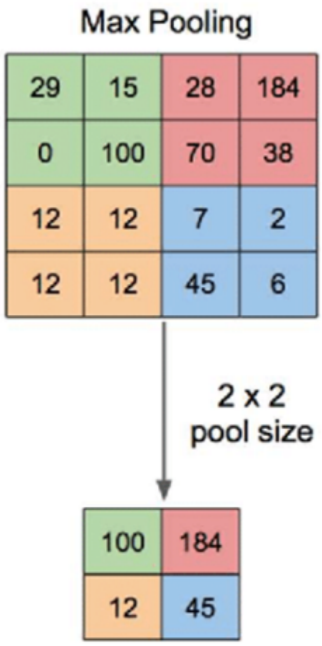

Figure 2.6: A brief schematic of Max pooling [29]

All feature maps are then mixed into a one dimensional vector to make a flatten layer. Next, in the FC layer, all neurons of both layers are connected to each other, like in the traditional multilayer neural networks.

The output of FC layer can be expressed by:

$$O = f(\sum_{j=1}^{d} x_j^F w_j + b) \tag{2.4}$$

where O is the output value, where O is the output value, x_j^F is the jth neuron in the fully connected layer, w_j is the weight corresponding to O and x_j^F , b_i is the bias corresponding to O, and f is a sigmoid function.

2.4 Alexnet

Alexnet [7], a CNN architecture designed by Alex Krizhevsky and his collaborators and His Ph.D supervisor respectively Ilya Sutskever and Geoffrey Hinton showed up in 2012 for the first time. the network was designed to compete in the ImageNet Large Scale Visual Recognition contest where Alexnet achieved a top-5 error in the competition. Despite being computationally expensive, the initial results of their paper declared that the depth of the network was vital for the accuracy, however it was feasible if graphics processing units (GPU) was utilised for the training.

The architecture consists of eight layers: five convolutional layers and three fully-connected layers. But this isn't what makes AlexNet special; these are some of the features used that are new approaches to convolutional neural networks:

2.4.1 ReLU Nonlinearity

In Alexnet instead of tanh function we have Rectified Linear Units(ReLU) which was common when alexnet was created. ReLU decreases the trainig time and was shown in a test by reaching a 25 percent error on the CIFAR-10 dataset up to 6 times quicker than other CNNs using tanh.

2.4.2 Multiple GPUs

Back then when the Alexnet structure was designed GPUs were quite weak compared to 2021 standards (around 3GB). When training the network with a big amount of images, low amount of memory was specially an obstacle. Alexnet resolved this problem by allowing a multi-GPU training, putting half of the models' neurons on the GPU and the other half on another. It both increases the volume that can be trained and decreases the training time.

2.4.3 Overlapping Pooling

CNNs traditionally "pool" outputs of neighboring groups of neurons with no overlapping. However, when the authors introduced overlap, they saw a reduction in error by about 0.5 percent and found that models with overlapping pooling generally find it harder to over-fit. CNNs usually pool the outputs of neighboring groups of neurons without overlapping. Alexnet's designer could reduce the error by 0.5 percent and discovered that models with overlapping maxpooling are more resisting to overfitting.

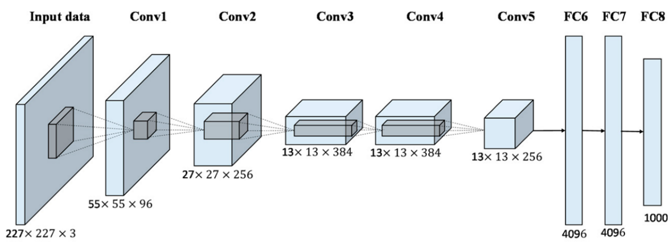

Figure 2.7: Overall Structure of the Alexnet [30]

2.4.4 Alexnet Structure in Matlab

All steps of this study including transfer learning and training were conducted utilising matlab where the pre-trained version of the network trained on more than a million images from the ImageNet database is downloaded from mathworks is accessible. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 227-by-227 in MATLAB.

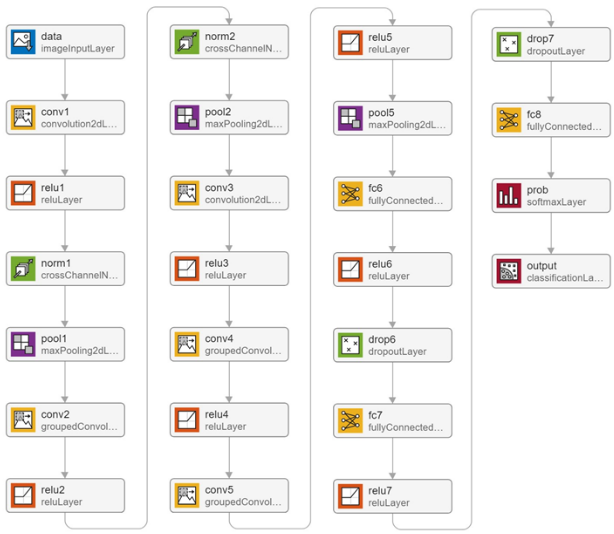

Figure 2.8: Alexnet in Matlab

2.5 Transfer Learning

One of the important assumptions in most of ML algorithms is that the feature data and training data must be in the same space regarding their features and distribution. In reality however, this assumption proves impractical. As an example, a lot of times we are interested in classification of domain A but there are only ample data in domain B. where B might have another feature space or its data is distributed differently. In this case, We can take advantage of transfer learning which if done correctly would increase the performance of training by avoiding labour consuming labeling task.[31]

My thesis focuses on using TL for three reasons, firstly because we do not have sufficient data to train a regression model from scratch. Secondly the TL in field of Prognostics is not fully exploited. And Thirdly, it could accelerate the training process of the deep network and learning the hierarchical representations. This can be done using CNNs that are pre-trained using big datasets of natural images. In our case we will see how the network architecture, model parameter and hyper parameters are transferred to the aimed model for the HI construction and fault diagnosis. In TL the lower level weights of the aimed model are obtained from the pre-trained network and the higher ones are tuned during the training process. This reduces the number of times parameters need to be updated. This way, TL can considerably decrease the training time of the Neural Network.

2.6 Experiment

2.6.1 Bearing Fundamentals

Bearings are basically made to provide a friction-less base to support a rotating shaft.Our focus in this thesis, is on a specific type of rolling element bearings only. Roller bearings work well in non-ideal conditions, but what happens is that bearings sometimes, surprisingly fail by minor causes. as an example, a stationary loading, minor vibrations will progressively push the lubricant inside the bearing out, and eventually lead to its failure. Bearing failure models are very well-understood at the moment since their failure mechanism and the theoretical aspects have been studied for decades. Usually, there are three elements limiting bearing's life expectancy and load capacity: abrasion, fatigue and pressure-induced welding, Although there are many other apparent causes of bearing failure, most can be reduced to these three. Abrasion is the erosion of the surface by hard material scraped from the bearing itself. Fatigue on the other hand happens when the specimen is under fluctuating form of loading. pressure induced welding happens when different metal parts of a system are put under high pressure and temperatures.

A roller element bearing's four major components are: inner racer, outer race, rolling elements (roller), and cage, All of which components might fail under operation. In general, the signatures of a damaged bearing consists of exponentially decaying ringing that occurs periodically at the characteristic frequency. The vibration signal of a defective bearing usually considers being amplitude modulated at characteristic defect frequency. Matching the measured vibration spectrum with the defect characteristic frequency enables defect detection and enables diagnosis on the defective area. [32]

2.6.2 Test Setup

In this test Four Rexnord roller bearings are installed on a shaft. An AC motor is coupled to the shaft via rub belts and the RPM is kept constant at 2000 RPM. All bearings are force lubricated and a radial load equal to 6000 lbs is put onto the shaft and bearing by a spring mechanism. Rexnord ZA-2115 double row bearings were installed on the shaft.[33] Rexnord ZA-2115 double row bearings have a pitch diameter of 2.815in., roller diameter of 0.331in., and a tapered contact angle of 15.17°.

Figure 2.9: Rexnord ZA-2000 bearing series[33]

2.6.3 Sensors and threshold

PCB 353B33 High Sensitivity Quartz ICP accelerometers are the sensors used in this setup installed on the bearings case (two accelerometers for [x- and y-axes] for each bearing for data-set 1, one accelerometer for each bearing for data-sets 2 and 3). Sensor placement is also shown in figure 2.10. All failures occurred after exceeding designed life time of the bearing which is more than 100 million revolutions. The debris collected by magnetic plug is used to indicate the degradation in bearing health. Data is collected until the accumulated debris which adhered to the magnetic plug exceeds a fixed-threshold level. The test was stopped adaptively by an electrical switch when the accumulated debris exceeded a certain level.

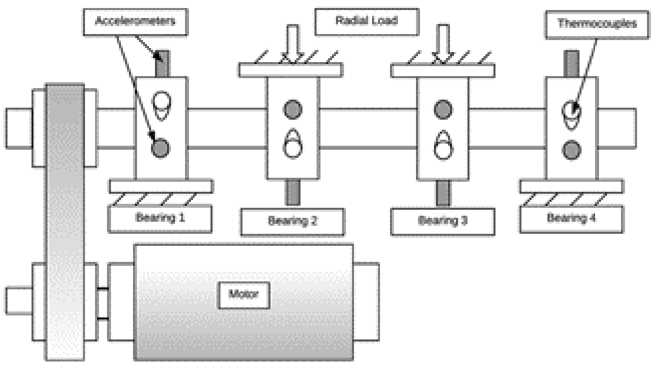

Figure 2.10: IMS bearing test rig and sensor placement illustration [33]

2.6.4 Data set

IMS Bearing Dataset

The IMS dataset was Introduced by the center for Intelligent Maintenance Systems (IMS) of University of Cincinnati [34], and shared on the website of the Prognostic Data Repository of NASA[35]. the Dataset is composed of 3 tests recording 3 full bearing degradation. After the tests, the bearings were checked and their fault patterns were recorded in detail. The data is collected at 20kHz for one second from the sensors installed on each bearing housing once every twenty minutes until the failure threshold were reached. The data has 20,480 points at each recording with sampling rate of 20kHz. [35]

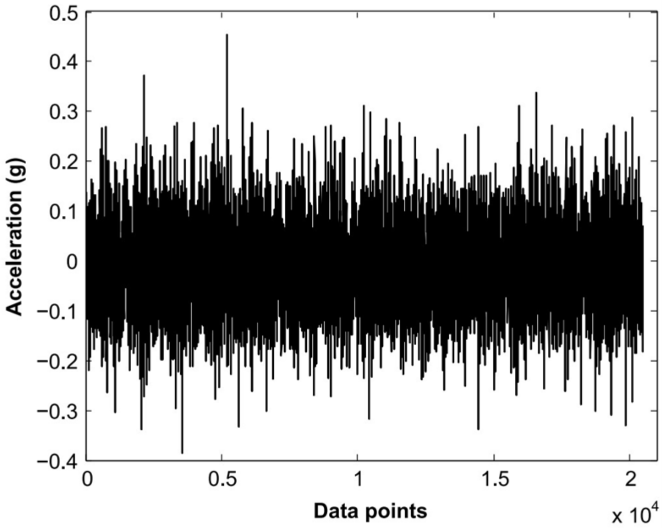

Figure 2.11: Vibration sample in 1 second

2.6.5 Empirical Results

	FirstTest	Second test	Third Test
Duration DD.MM	22.10-25.11	12.2-19.2	04.03-04.04
Number of Files	2156	984	6528
Number of Channcels	8	4	4
Acquisition Interval [min]	10	10	10

Table 2.1: IMS bearing test results

First Test

In the first dataset, At the end of the test-to-failure experiment, inner race defect happened in bearing 3 (sensor 5,6) and roller element defect in bearing 4 (sensor 6,7)

Time domain representation of the 1st test sensor data, each row represents the sensors mounted on one bearing.

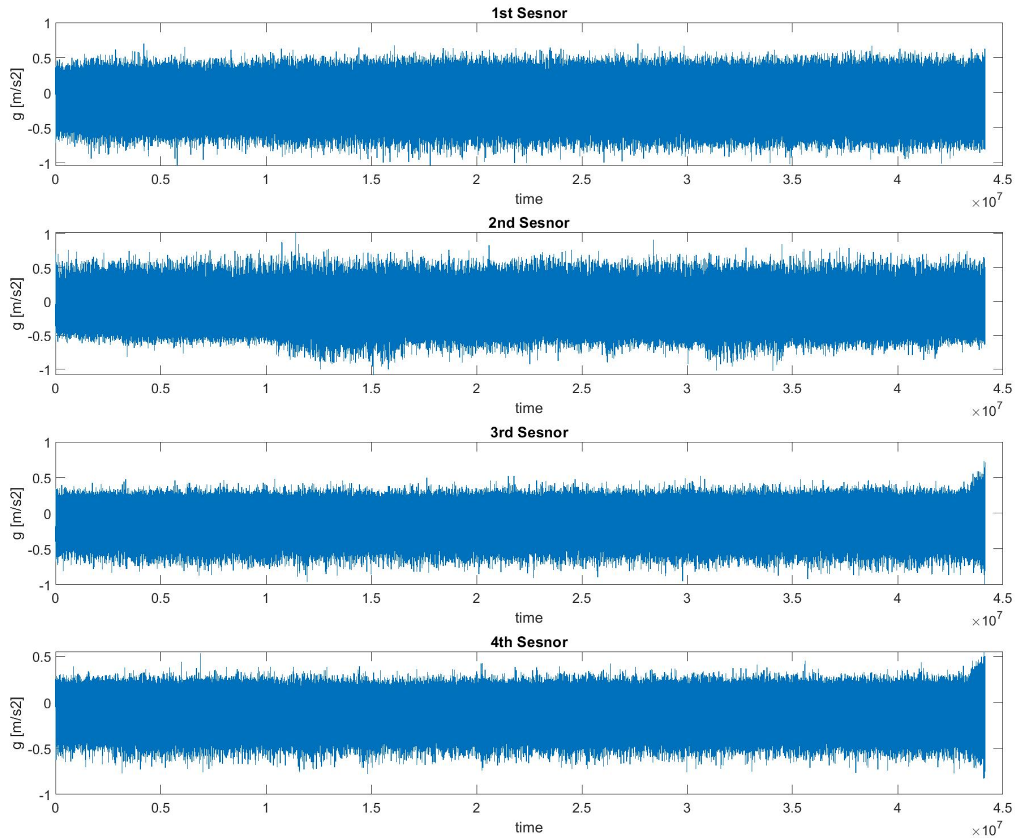

Figure 2.12: Start to Failure Time Domain representation of the First Test

Second Test

At the end of the test-to-failure experiment, outer race failure occurred in bearing 1. The most commonly reported results concern outer-race defects. This is primarily because outer-race defects are easy to seed in the laboratory and often produce the most salient fault signatures. [36]

Figure 2.13: bearing outer race failure [37]

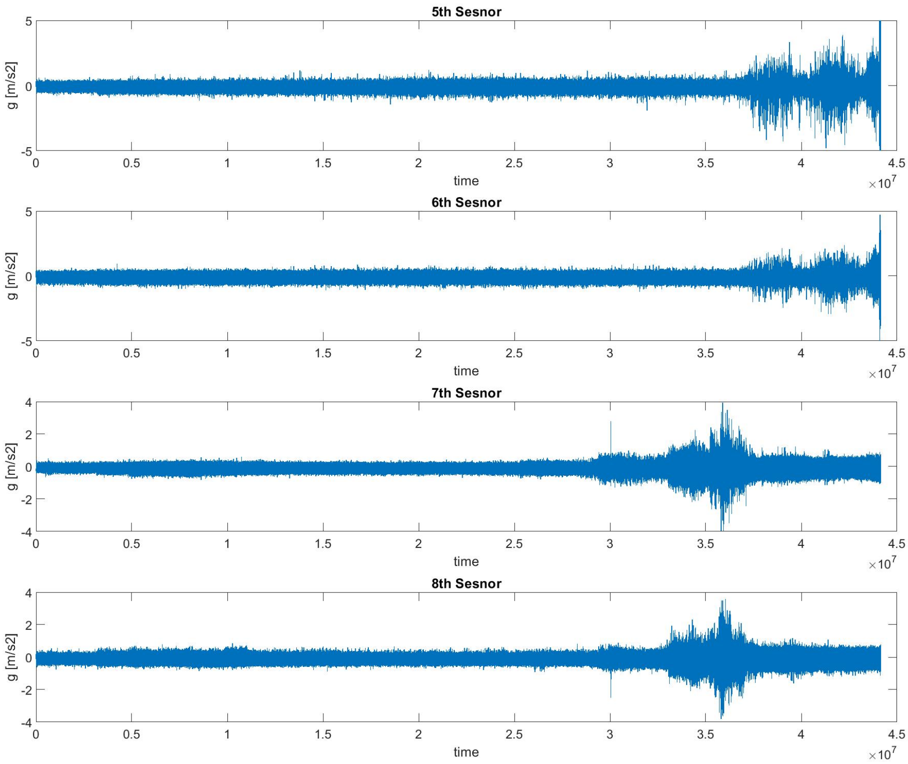

Figure 2.14: Start to Failure Time Domain representation of the First Test

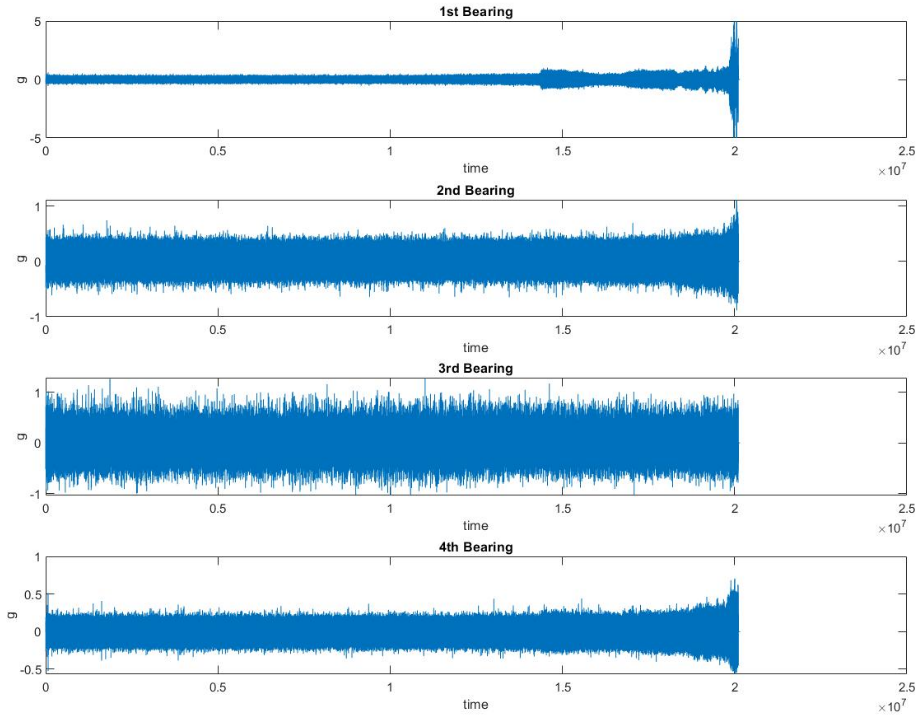

Figure 2.15: Start to Failure Time Domain representation of the Second Test

Third Test

At the end of the test-to-failure experiment, outer race failure occurred in bearing 3.

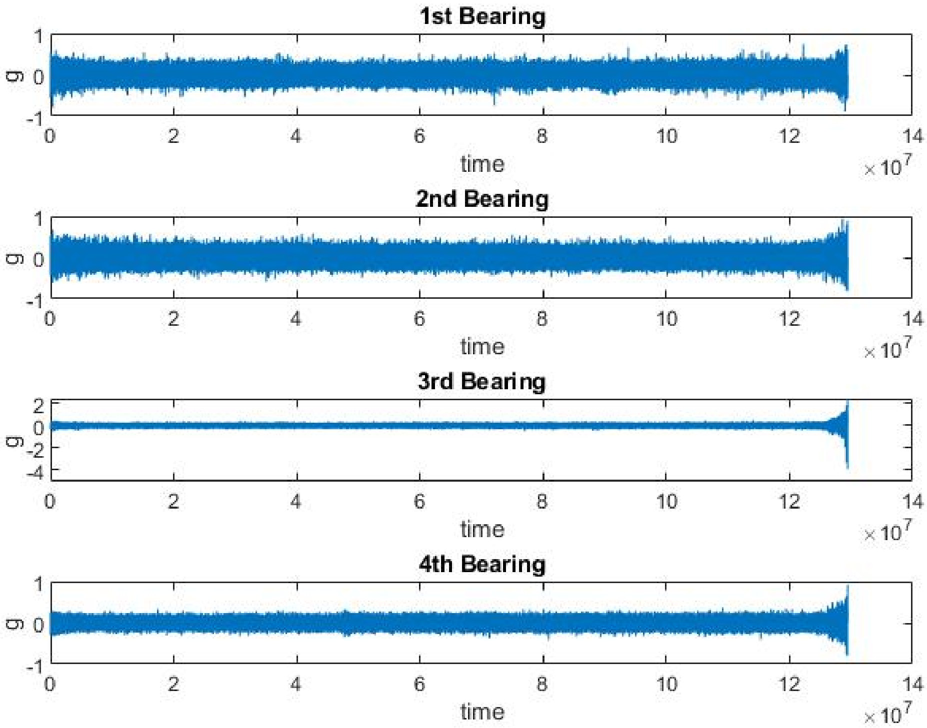

Figure 2.16: Start to Failure Time Domain representation of the Third Test

2.6.6 Affecting Variables

Various conditions effect the datasets usually but different datasets give different results on same training models. A faulty bearing dataset is composed of a signal, and a signal is made up of three components: (1) frequency, (2) amplitude, (3) phase. For each type of fault these three properties might differ, and variations in the signals can be observed within the signals of same health type if they are collected under different working conditions. So, the type of data utilised in fault diagnosis of the part has a great significance, as it affects the accuracy of the developed model. [38]

No	Affecting Parameters
1	Bearing specification
2	Outer race diameter
3	Inner race diameter
4	Ball diameter
5	Ball number
6	Contact angle
7	Clearance
8	Noise
9	Phase angle
10	Change in amplitude
11	Change in sampling frequency

Table 2.2: Parameters and conditions affecting the output signals

Chapter 3

Work Development

3.1 Signal Processing

The major part of the spectrograms in this study are created using short time Fourier Transform STFT method on MATLAB. The spectrogram(x) function in MATLAB returns the short-time Fourier transform of the input signal, x. if the output of such function is assigned to variable S, each column of S contains an estimate of the short-term, time-localized frequency content of x.

Another necessary input would be number of time windows that the original signal will be divided into.

```
s = spectrogram(x,window,noverlap)
```

Figure 3.1: Matlab spectrogram function

Matlab code above uses “window” to divide the signal into segments and perform windowing and in order to avoid leakage and signal loss due to windowing an overlap is considered by “noverlap” input.

For having a well detailed spectrogram, number of sampling points to calculate the discrete Fourier transform is also defined by “nfft” input, which is the number of the FFT points used for Fast Fourier Transform algorithm.

Trial for correct Spectrograms

As mentioned before the success of STFT is hinged on properly choosing window length and frequency resolution, which often time is difficult. in this study we have tried different window lengths and frequency resolutions in order to achieve a correct spectrogram resolution for both in time and frequency axis.

caption caption

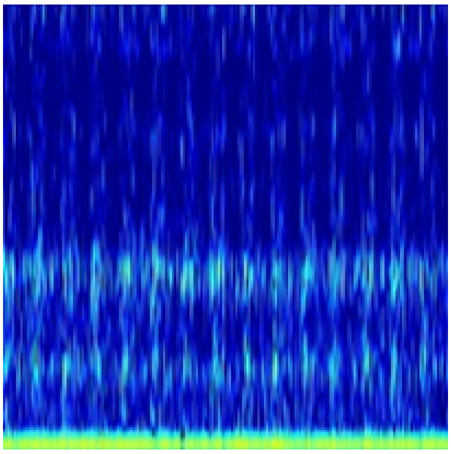

Figure 3.2: time section length:56 overlap:50% Freq resolution:512

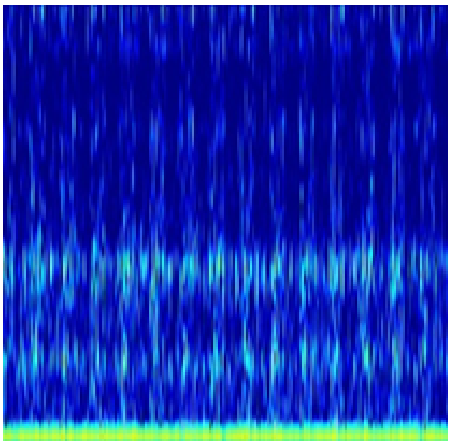

Figure 3.3: time section length:56 overlap:90% Freq resolution:512

It is apparent that in this case (i.e. time section:52) the frequency axis is has not a good resolution (i.e. stretched vertically). In order to improve the result, we shall increase the time chunk length. In addition to that, from the images it is obvious that by increasing the overlap percentage, the resolution improves.

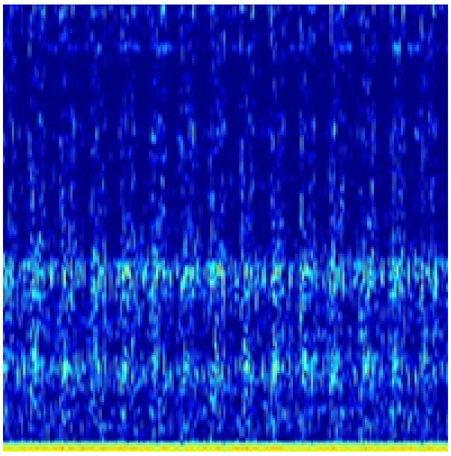

Figure 3.4: time section length:128 overlapping:50% Frequency resolution:512

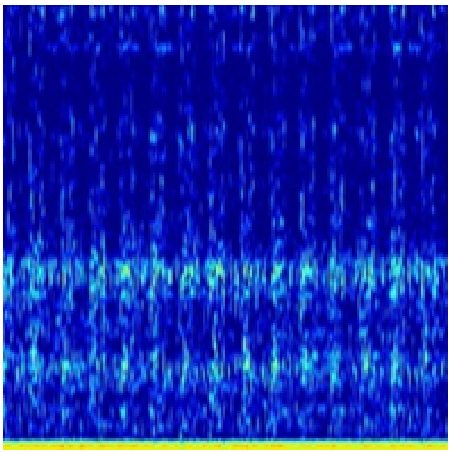

Figure 3.5: time section length:128 overlapping:90% Frequency resolution:512

Clearly now, the frequency axis has a better resolution, therefore we continued on increasing the time section length to compensate for better frequency resolution. Again, by increasing the overlap the results improve.

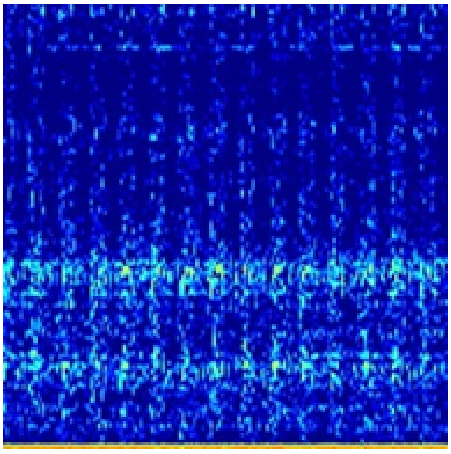

Figure 3.6: time section length:256 overlapping:50% Frequency resolution:512

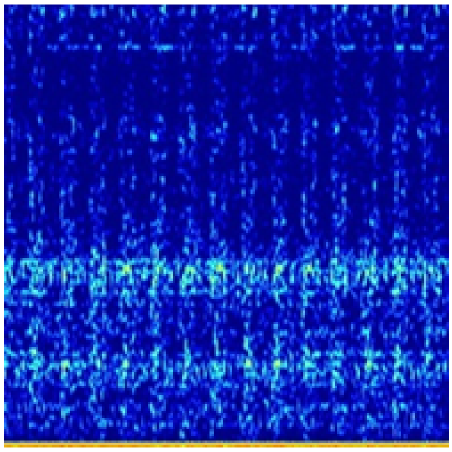

Figure 3.7: time section length:256 overlapping:90% Frequency resolution:512

As it can be seen, by further increasing the time section length, We end up with clearer images. Next step would be further increasing the time section length.

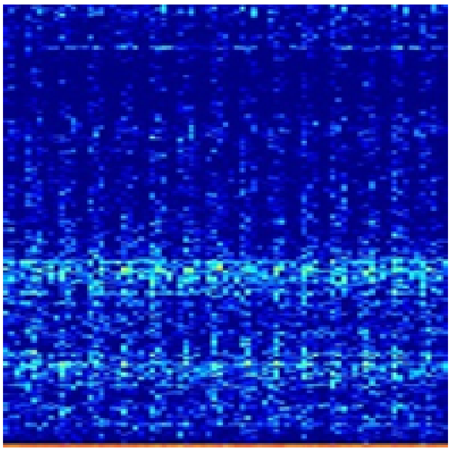

Figure 3.8: time section length:512 overlapping:50% Frequency resolution:512

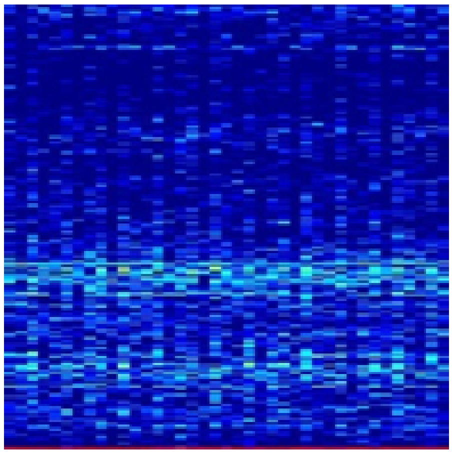

Figure 3.9: time section length:1024 overlapping:50% Frequency resolution:512

Increasing the time length, clearly, negatively affected the time axis leading to a decrease of the time resolution, therefore we fix 256 samples as the time length and 90% as the time overlap. Now let's change the frequency resolution to see how it affects the resolution:

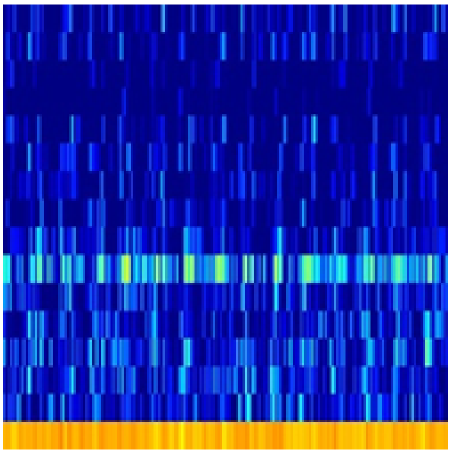

Figure 3.10: time section length:256 overlapping:90% Frequency resolution:30

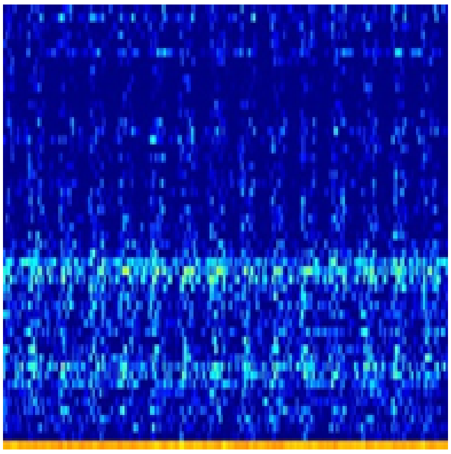

Figure 3.11: time section length:256 overlapping:90% Frequency resolution:100

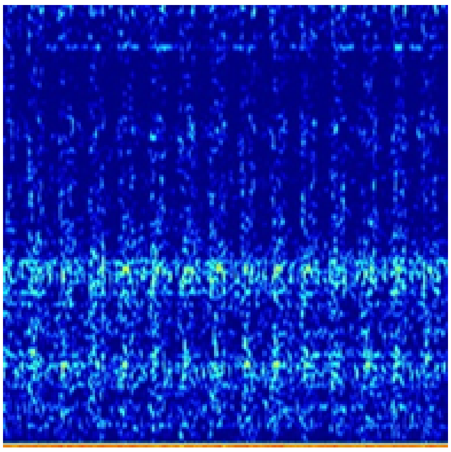

Figure 3.12: time section length:256 overlapping:90% Frequency resolution:1024

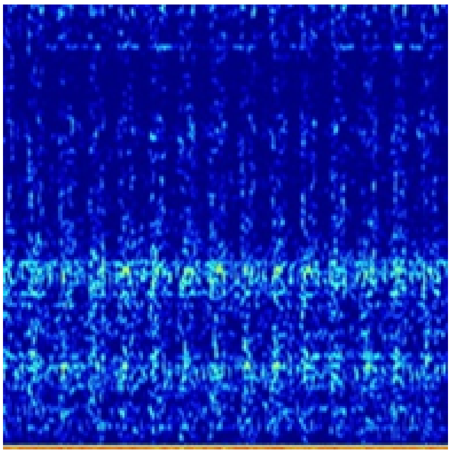

Figure 3.13: time section length:256 overlapping:90% Frequency resolution:20480

It is obvious that by over-decreasing the frequency resolution(30 bands), the image resolution vastly drops and over-increasing it (1024 to 20480) doesn't increase the quality and involves more computational resources.

In conclusion, very small time windows lead to lower frequency resolution and vice versa. Also, increasing the sampling points after a threshold will increase the computation overhead and does not significantly increase the resolution. According to the results we have decided to set the time window sample number to 256 with 98 percent overlap and 512 sampling points to calculate the discrete Fourier transform.

Normalization

In order to have correct trend in amplitude growth, the global minimum and maximum amplitudes must be found and be fixed as the limits for all spectrograms, otherwise amplitudes (i.e. the colorscale) feature in Matlab will be in different ranges in each image and result in wrong trend. The Output P0 (Apendix 0.1) in our code is a matrix containing the amplitudes of all STFT vectors from which the higher and lower limits must be found. In the appendixes A.0.1 you can find the Matlab codes that create the spectrograms.

Amplitude Axis Space

Initially We have produced a full dataset with linear amplitude space. However, due to its extension the growth trend seemed to be very dis-articulated.

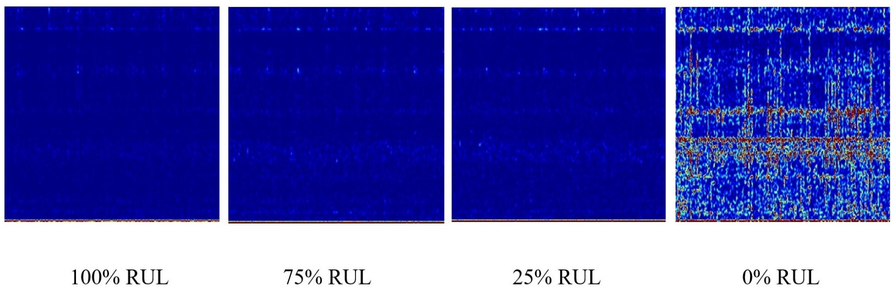

Figure 3.14: Samples of the spectrograms with linear Z-axis

as it can be seen, the color range is almost binary since the harmonics show very different amplitudes. So We have decided to set the amplitude space (i.e. colorscale) to logarithmic. and the spectrograms changed as it can be seen below:

3.2 Transfer Learning Procedure

Available image classification networks like GoogleNet and Alexnet are trained with millions of images and can identify images into hundreds or thousands of categories, such as trees, glasses, pecs, and animals. These networks have mastered feature representations for a wide variety of images in a way that they take an image as input, and then return a label for the object in the image together with

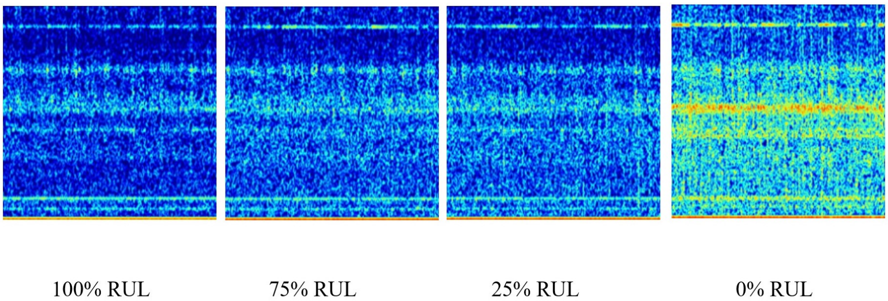

Figure 3.15: Samples of the spectrograms with logarithmic Z-axis

the probabilities for other object categories. Transfer learning is now widely used in DL applications. With any pretrained network you can use as a starting point to make it learn a new task. Furthermore, as noticed before the major focus of this thesis will be on Alexnet as a pretrained network which is originally a natural object classifier trained with millions of images. our objective though, is to estimate the RUL of the bearings. The difference between the two is that the output of the network should be a continuous number rather than a class. This necessitates converting the network from a classifier to a regression network.

1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Grouped Convolution	2 groups of 128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Grouped Convolution	2 groups of 192 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Grouped Convolution	2 groups of 128 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench' and 999 other classes

Figure 3.16: Original Alexnet layers

The convolutional layers of the network extract image features that the last learnable layer and the final classification layer use to classify the input image. Such layers are named as “conv” in the Alexnet. They are called the core layers of the pretrained network and in our case will not be manipulated in transfer learning process. The layers after the final core layer might be deleted or replaced according to the new problem’s need. The fully connected (FC) layer "fc8" and the classifier output carry the information on how to combine the features extracted by the network to achieve the class probabilities, loss value and predicted labels. To retrain our pretrained network for regression, these two layers must be changed with other layers adapted to regression tasks. The original fully connected layer has 1000 responses, we here replace it with another FC layer with a single response and the classification output must be replaced with a regression layer. A regression layer computes the half-mean-squared-error loss for regression tasks.[39]

Softmax layer is a function that takes a vectors of numbers as input and return a probability vector, where the probabilites are normalized according to the relatie scale of each value in the vector. Such a function in our task is use useless since there is only one output.In the Alexnet softmax is the 24th (Counting ReLu and Norm as Layers, otherwise 8th) layer and is called “prob”.[39]

Figure 3.17: Softmax function [40]

The proposed network to begin the training process is shown in fig 3.18 and fig 3.19. The initial network might change as we try to optimize the accuracy of the network.

3.3 Training options

Training options are a set of variables you can set to control network training. Both accuracy and training time are part of a network’s performance. There is no

Figure 3.18: Alexnet Structure Converted to a regression Network

```
layers_1 =
  24×1 Layer array with layers:

     1   'data'              Image Input                   227×227×3 images with 'zerocenter' normalization
     2   'conv1'             Convolution                   96 11×11×3 convolutions with stride [4  4] and padding [0  0  0  0]
     3   'relu1'             ReLU                          ReLU
     4   'norm1'             Cross Channel Normalization   cross channel normalization with 5 channels per element
     5   'pool1'             Max Pooling                   3×3 max pooling with stride [2  2] and padding [0  0  0  0]
     6   'conv2'             Grouped Convolution           2 groups of 128 5×5×48 convolutions with stride [1  1] and padding [2  2  2  2]
     7   'relu2'             ReLU                          ReLU
     8   'norm2'             Cross Channel Normalization   cross channel normalization with 5 channels per element
     9   'pool2'             Max Pooling                   3×3 max pooling with stride [2  2] and padding [0  0  0  0]
    10   'conv3'             Convolution                   384 3×3×256 convolutions with stride [1  1] and padding [1  1  1  1]
    11   'relu3'             ReLU                          ReLU
    12   'conv4'             Grouped Convolution           2 groups of 192 3×3×192 convolutions with stride [1  1] and padding [1  1  1  1
    13   'relu4'             ReLU                          ReLU
    14   'conv5'             Grouped Convolution           2 groups of 128 3×3×192 convolutions with stride [1  1] and padding [1  1  1  1
    15   'relu5'             ReLU                          ReLU
    16   'pool5'             Max Pooling                   3×3 max pooling with stride [2  2] and padding [0  0  0  0]
    17   'fc6'               Fully Connected               4096 fully connected layer
    18   'relu6'             ReLU                          ReLU
    19   'drop6'             Dropout                       50% dropout
    20   'fc7'               Fully Connected               4096 fully connected layer
    21   'relu7'             ReLU                          ReLU
    22   'drop7'             Dropout                       50% dropout
    23   'fc'                Fully Connected               1 fully connected layer
    24   'regressionoutput'  Regression Output             mean-squared-error
```

Figure 3.19: Alexnet layers as converted to a regression Network

single "right" way to set the training options for a deep neural network but there is always a balance between maintaining a high accuracy and decreasing training time. A general guideline to improve the training is shown in fig 3.20

3.3.1 Solver

Selecting the right solver is so important that the choice of optimization algorithm for a task can differ between good results in minutes, hours, and days. The preferred algorithm in this thesis is the "adam" solver. [41] Adam optimization algorithm is an extension of the stochastic gradient descent [42] that has been used broadly for deep learning applications in computer vision and natural language processing recently. Authors that have used Adam Solver list some of the attractive benefits of using Adam on non-convex optimization problems mentioned down below:

1. Straightforward to implement.
2. Computationally efficient.
3. Little memory requirements.
4. Invariant to diagonal re-scale of the gradients.
5. Well suited for problems that are large in terms of data and/or parameters.
6. Appropriate for non-stationary objectives.
7. Appropriate for problems with very noisy/or sparse gradients.
8. Hyper-parameters have intuitive interpretation and typically require little tuning.

Adam combines the best properties of the AdaGrad [43] and RMSProp algorithms [44] to provide an optimization algorithm that can handle sparse gradients on noisy problems.

Adam keeps an element-wise moving mean of the parameter gradients and their squared values:

$$m_\iota = \beta * m_{\iota-1} + (1 - \beta_1)\nabla E(\Theta_l) \tag{3.1}$$

$$\nu_l = \beta_2 \nu_{l-1} + (1 - \beta)[E(\Theta_l)]^2 \tag{3.2}$$

where is the iteration number, >0 is the learning rate, is the parameter vector, and E() is the loss function and 2 is the decay rate of the moving average. You can specify the 1 and 2 decay rates using the 'Gradient Decay Factor' and 'Squared Gradient Decay Factor' name-value pair arguments, respectively. Adam uses the moving averages to update the network parameters as:

$$\Theta_{l+1} = \Theta_l - \frac{\alpha m_l}{\sqrt{\nu} + \epsilon} \tag{3.3}$$

If gradients are similar over many iterations, then to pick up momentum in a certain direction, using a moving average of the gradient enables the parameters to update. If the gradients signal to noise ratio is small, then the moving average of the gradient gets smaller, and as a result the parameter updates become smaller as well.

3.3.2 Learning Rate

Deep learning neural networks are trained using the stochastic gradient descent optimization algorithm. The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.[28]

The graphs shown during the learning process can help us diagnose the problems we might encounter. In case there are large spikes in the loss value, or if they're not being plotted at all. It is probably because the initial learning rate is very high. A general rule is decreasing the initial learning rate by a power of ten until the spikes disappear.

3.3.3 Minibatch and Epoch

In every iteration, a number of the training images, known as a mini-batch, is taken to update the weights. Each iteration uses a unique mini-batch. Once the whole training set has been involved, we have an Epoch. The number of epochs (MaxEpochs) and the number of images in each mini-batche (MiniBatchSize) are the variables you can set in the network training options. Remember that in setting the number of images in each minibatch and epoch number there is no standard. Increasing the number of minibatches in an epoch until a maximum usually improves the training accuracy and epoch numbers should be increased until loss and RMSE values reach a plateau.

The loss and accuracy reported during training are for the mini-batch being used in the current iteration.

3.4 Training procedure

To perform transfer learning we need to prepare 3 things. First we need layers that represent the network which was discussed in the previous section. Secondly we need an image datastore with corresponding labels, and last, a variable holding algorithm settings (i.e. Training options). Once these three factors are introduced the training process begins and the outcome is a the same network but with updated weights. In MATLAB by default we see a text display showing the progress of the training. Accuracy is the percentage of the training images that network classifies correctly. We want to see that increasing during training and loss is a measure of how far we are from a perfect prediction totalled over the set of training images which should decrease toward zero as the training proceeds. [28]

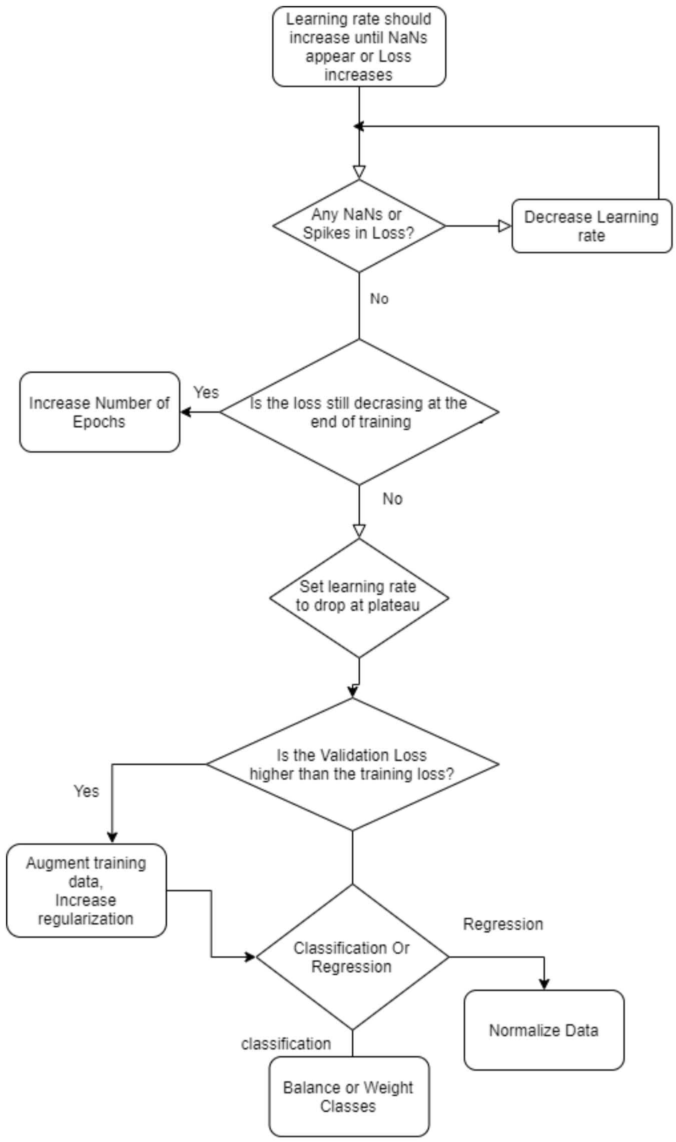

Figure 3.20: Troubleshooting of the training process algorithm

Sensor nomenclature

To avoid repeating sensor names each time I introduce a set of nomenclature for the sensors in n-m format where n is the test number and m is the sensor number e.g. 1-5 is the fifth sensor in the first test.

3.4.1 Labelling

The training images are represented by $dataset = (x, y)_{t=1}^{T}$ where $x^{N*N}$is N×N image at time t. In our case the image size is 227×227 for Alexnet and 224×224 for Resnet. The degradation y is labelled as 1 for a totally healthy bearing decreasing to 0 for failed bearing.

3.5 Results

3.5.1 Initial Results

First Try

As a starting point, we kept the core (hidden) layers of the Alexnet, and started the training process. This network's layers can be find in table 3.3.

Basically, including the bearings that did not develop full failure, wouldn't make sense as long as we can't confirm the actual RUL of them, hence, our dataset is limited to the bearings that did actually fail. On the other hand, the IMS bearing dataset is a real world experiment confirming that a Bearing's life stages appear non-linearly. Despite under going equal RPM and Forces, from the 12 bearings in the experiment only 4 actually failed. Also, the time the defects appeared and the defect type was different in each failed bearing.

Condition	Training Datasets	Testing Dataset
Test 1	1-5 1-6 1-8	1-1 1-2 1-3 1-4 1-7
Test 2	2-1	2-2 2-3 2-4
Test 3	3-3	3-1 3-2 3-4

Table 3.1: Sensors participating in the training and testing the Network

We know that if low in amplitude, informative signals are masked by environment noise,[45] hence, we are expecting that including the spectrograms of the early stages in the training process will not yield accurate results since the early signatures are easily buried by the noise. However, to experiment what will the outcome be, we did include them and the results are presented as shown in table 3.5:

# Epochs	Iterations per Epoch	Learning rate Schedule	Learning rate	Minibatch #	RMSE test	RMSE training
5	1239	Piecewise	1e-07	20	NaN	NaN

Table 3.2: Training Options and Results

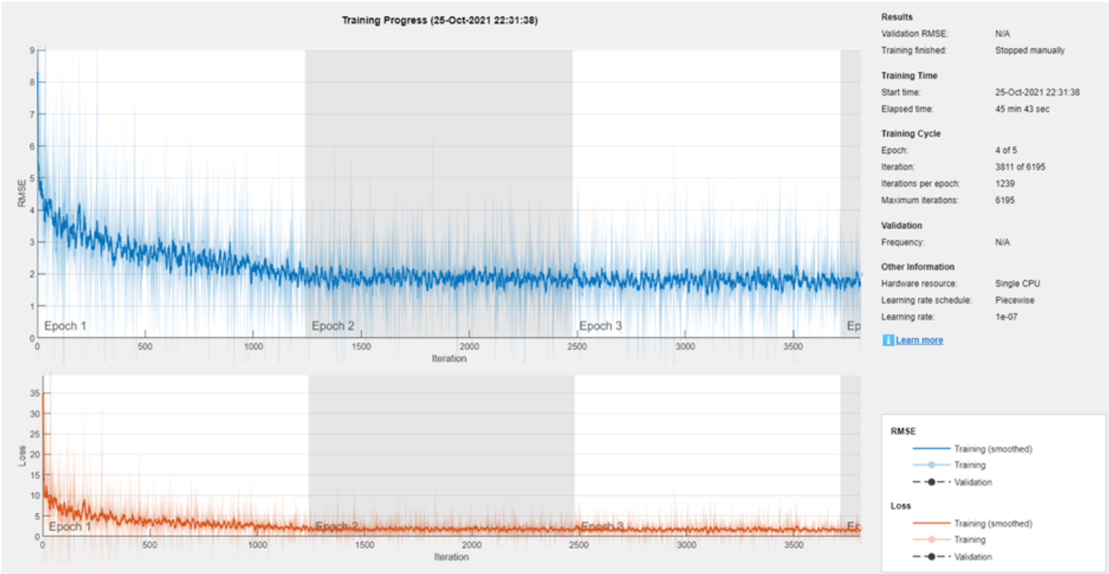

Figure 3.21: Graph of the training progress

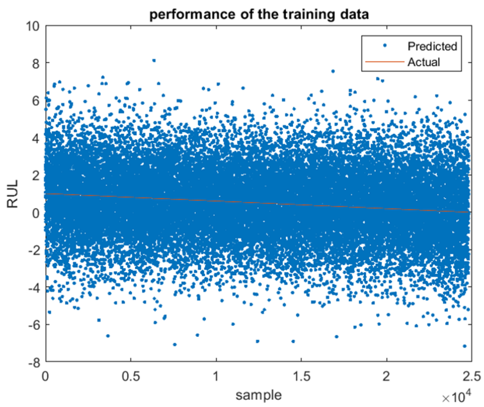

Figure 3.22: Predictions versus the actual RUL of the training images

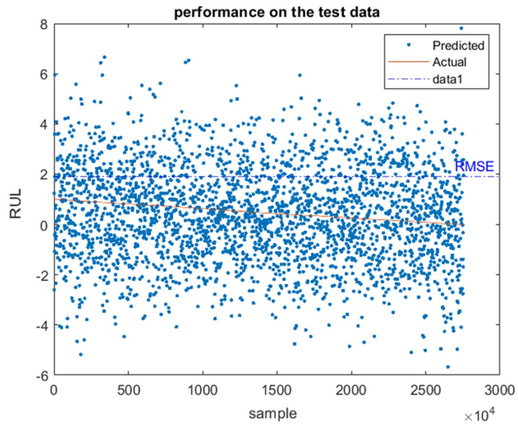

Figure 3.23: Predictions versus the actual RUL of the test images

From the prediction graphs 3.22 3.23 it is evident that the results are very unsatisfactory with such settings so we should try with another settings.

	Name	Type	Description
1	'data'	Image INput	$227\times227\times3$ images with 'zerocenter' normalization
2	'conv1'	Convolution	96 $11\times11\times3$ convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLu	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3×3 max pooling with stride [2 2] amd padding [0 0 0 0]
6	'conv2'	Grouped Convolution	2 groups of 128 $5\times5\times48$ convolutions with stride [1 1] with stride [2 2 2 2]
7	'relu2'	ReLu	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3×3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 $3\times3\times256$ convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLu	ReLU
12	'conv4'	Grouped Convolution	2 groups of 192 $3\times3\times192$ convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLu	ReLU
14	'conv5'	Grouped Convolution	2 groups of 128 $3\times3\times192$ convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	Max Pooling	ReLU
16	'pool5'	Fully Connected	3×3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc'	Regression Output	1 fully connected layer
18	'reg'		mean-squared-error

Table 3.3: The Initial transferred network's layers used in training

2nd Try

Re-editing the network

Initially, We omitted all the layers after the hidden layers of the network. In order to improve the performance of the network, now, We placed the original layers back in place converting the output layer (classifier) to a regression layer, only. the replaced layers of this test can be found on table 3.4.

	Name	Type	Description
16	'pool5'	Fully Connected	3×3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc'	Regression Output	1 fully connected layer
18	'relu6'	ReLu	ReLu
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLu	ReLu
22	'drop7'	Dropout	50% Dropout
23	'fc'	Fully Connected	1 fully connected layer
24	'regressionoutpu'	Regression Output	mean-squared-error

Table 3.4: Re-edited transfered network's layers used in training

Not surprisingly taking advantage of Alexnet's complete structure improves the results of the second trial which are presented in figures 3.24 3.25. The real performance of the transferred network should be tested on a sensor that didn't participate in the training at all. This is reported in fig 3.26.

Figure 3.24: Predictions versus the actual RUL of the training images

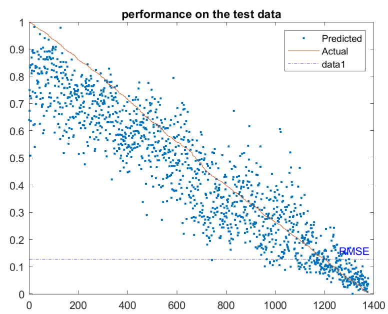

Figure 3.25: Predictions versus the actual RUL of the test images

# Epochs	Iterations per Epoch	Learning rate Schedule	Learning rate	Minibatch #	RMSE test	RMSE training
5	1239	Piecewise	1e-07	20	0.13	0.11

Table 3.5: Training Options and Results

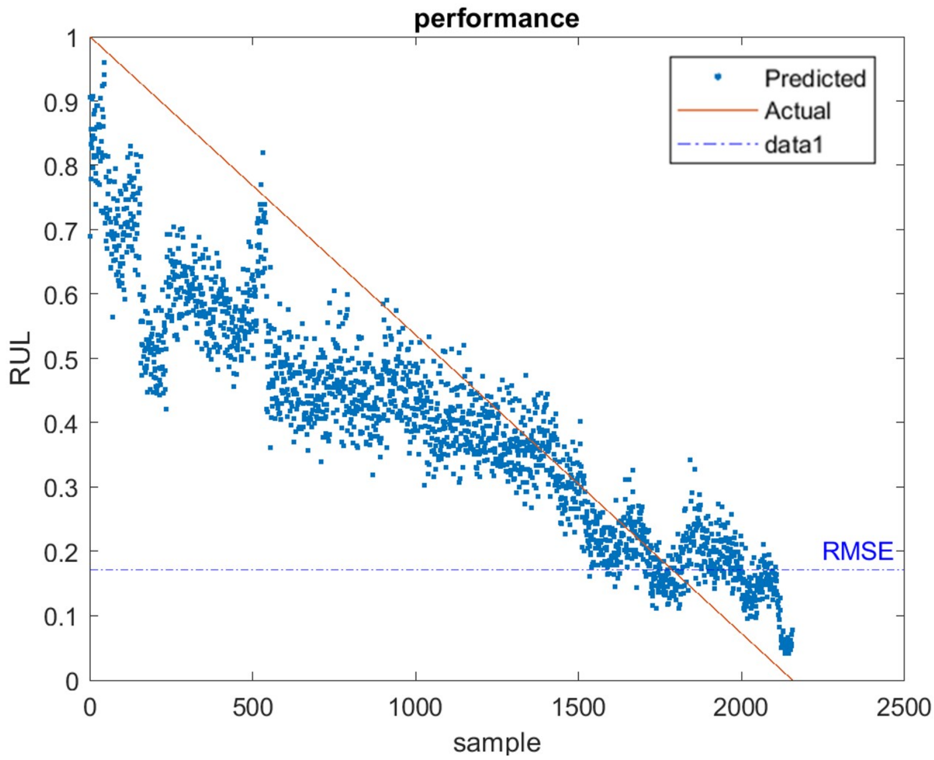

Figure 3.26: Predictions versus actual RUL of sensor 1-7

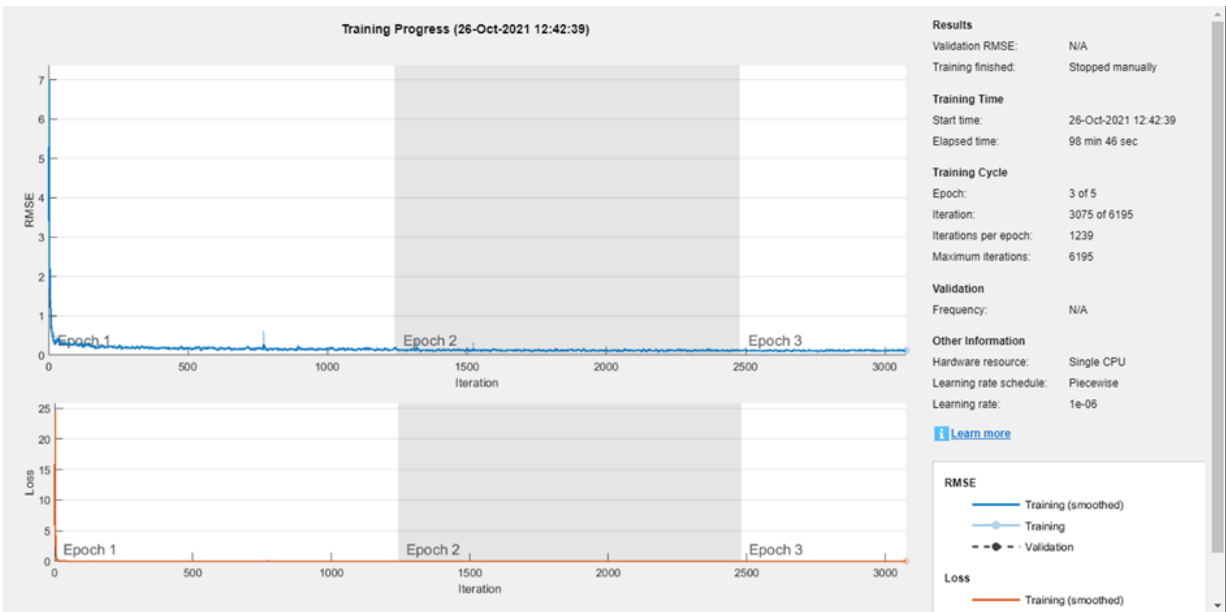

Figure 3.27: Graph of the training progress

The Root Mean Square Error values are acceptable with such setting but more importantly we should check the performance of the network on the a sensor that did not contribute in the training process. Such performance is shown at figure **??**. The RMSE of the network with this sensor is 0.1709 which is not bad. In the next trials we shall follow the troubleshooting Flowchart presented before. 3.20

3.5.2 3rd Try

3.5.3 Data Augmentation

One problem with system prognostics and predictive maintenance is lack of faulty signals as shortly after faulty signals pop out the machine (bearings in this case) fails. To regularize and help lowering the over-fitting of the network, we can augment the dataset images. Data Augmentation is used to increase the quantity of existing data (images) by slightly changing it and adding it to the dataset.

To that end we tried to double the number of spectrograms by halving the acquisition time from 1 second (20e3 samples) to 0.5 second (10e3 samples). Look at fig 3.28

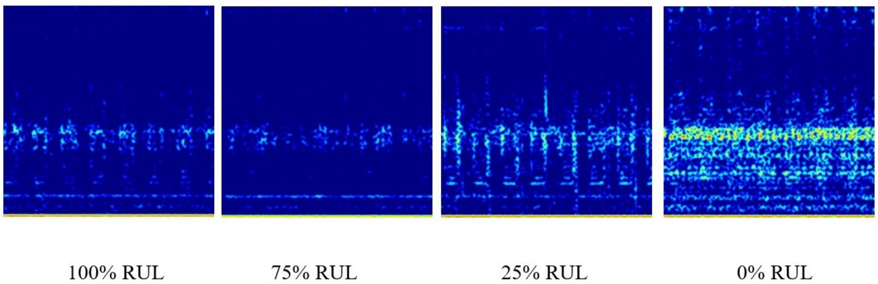

Figure 3.28: The spectrograms with halved time axis

This Data Augmentation had no great effect on training and test dataset (0.1124 and 0.1110) of the sensors who participated in the training process. Data The results are introduced in fugues below: 3.30 3.29

Again the performance of the network on the sensor that did not contribute in the training process(1-7) is shown in fig 3.31. By doubling the images the RMSE value of the prediction on sensor 1-7 decreased by 4 percent to 0.1309.

3.5.4 Sigmoid function as second to the last Layer

We set the RUL value in the (0,1) limit. In predictions though, There are outliers beyond this interval. A sigmoid layer applies a sigmoid function to the input such that the output is bounded in the interval (0,1).

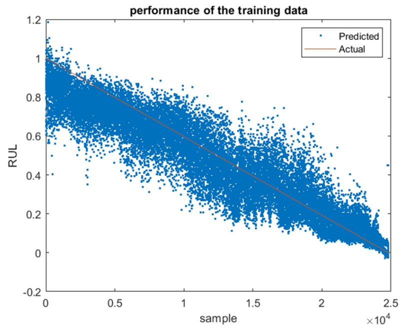

Figure 3.29: Predictions versus the actual RUL of the training images

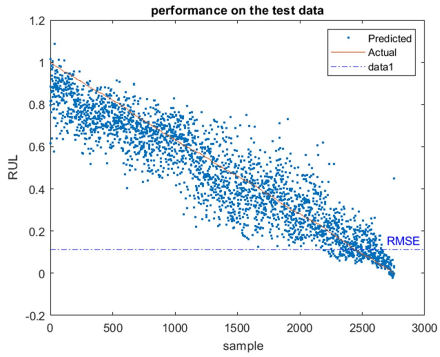

Figure 3.30: Predictions versus the actual RUL of the test images

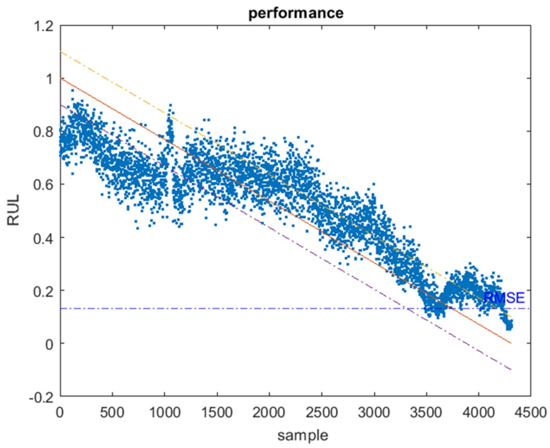

Figure 3.31: Prediction versus actual value of the sensor 1-7

3.5.5 4th Try

Changing the training dataset

We have then changed the training dataset in order to see the effects on the trained network performance. The new setting is shown in fig 3.7. Specifically in the new trial we switched, sensor 1-7 with sensor 1-6.

The results are as shown in figs 3.33 3.32 for trained sensors and fig 3.34 for sensor 1-6 that was not used in the training procedure. The RMSE value of the Training is equal to 0.1016 and that of the Test is 0.10129.

	Name	Type	Description
16	'pool5'	Fully Connected	3×3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc'	Regression Output	1 fully connected layer
18	'relu6'	ReLu	ReLu
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLu	ReLu
22	'drop7'	Dropout	50% Dropout
23	'fc'	Fully Connected	1 fully connected layer
24	'sigmoid'	Sigmoid	Sigmoid
25	'regressionoutpu'	Regression Output	mean-squared-error

Table 3.6: Re-edited transfered network's layers used in training

Condition	Training Datasets	Testing Dataset
Test 1	1-5, 1-7, 1-8	1-1, 1-2, 1-3, 1-4, 1-6
Test 2	2-1	2-2, 2-3, 2-4
Test 3	3-3	3-1, 3-2, 3-4

Table 3.7: Sensors participating in the training and testing the Network

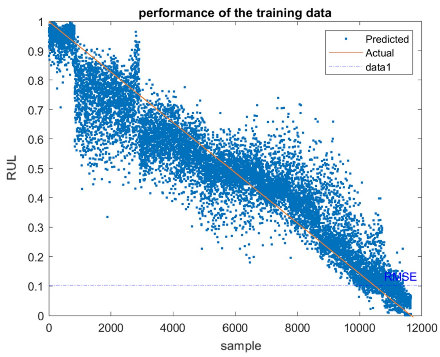

Figure 3.32: Predictions versus the actual RUL of the training images

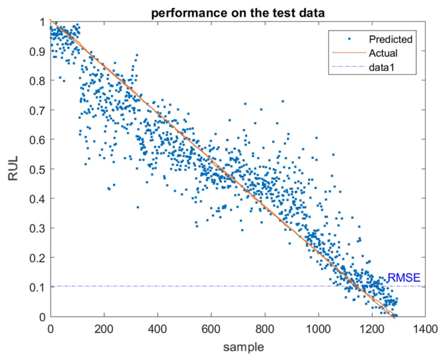

Figure 3.33: Predictions versus the actual RUL of the test images

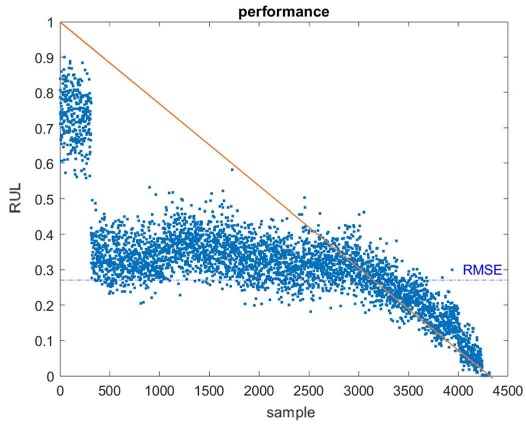

Figure 3.34: Prediction versus actual value of the sensor 1-7

The RMSE value for sensor 1-6 is equal to 0.2703. again, we can observe that in the first half of the bearing life the estimations are not accurate enough. this is because in early stages, the degradation features are buried in noise. [46]

Step shaped Labelling

The idea behind such labelling is based on the face that, at least in the very first periods of bearing's life, the spectrograms' are very similar to each other which might confuse the network. In this section we tried to label the images in batches to see if it improves the accuracy. Such labelling is shown in fig 3.35

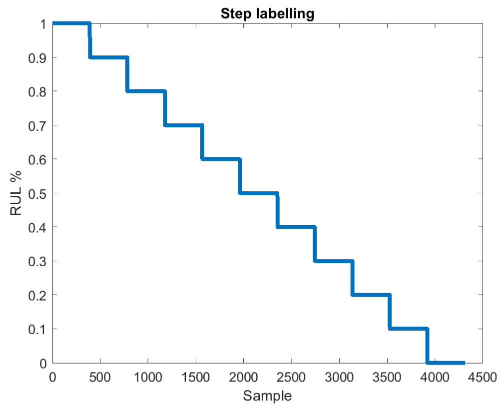

Figure 3.35: Step Shaped Labelling, such labelling might cause less confusion

In this case, we have divided the life into 10 steps decreasing the RUL by 10 percents each time.

$$\forall n \in (1, 10) : RUL = 1 - 0.1 * n$$

where n=0 stands for fully healthy and n=10 for failed bearing. The results of training in this case show an improvement in the performance of the network on the sensors our model has not seen before:

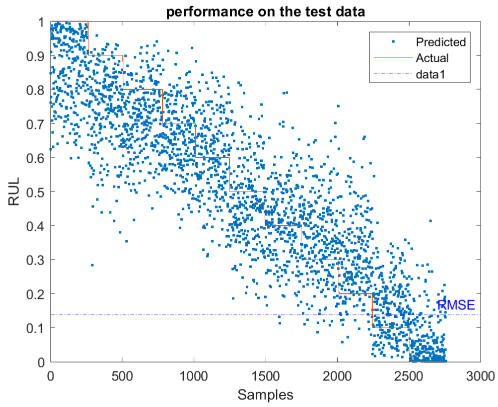

Figure 3.36: Step Shaped Labelling, Predictions versus actual values for test dataset RMSE=0.1374

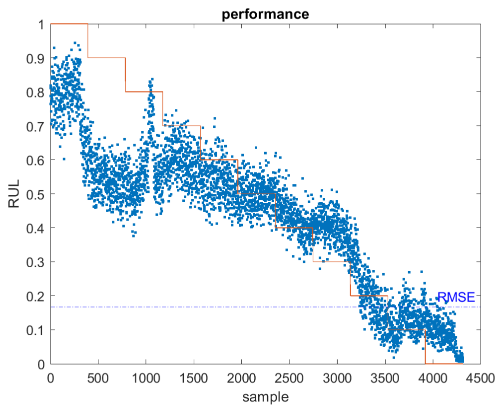

Figure 3.37: Predictions versus actual values for Sensor 1-7 which did not take part in the training process RMSE=16.67

Evidently, overall accuracy has improved. But even though the new model improved the accuracy of prediction of the mid-life samples, predictions of the final samples are less accurate with respect to the previous model. This urges us to train the network based on several models dedicated to different stages using spectrograms containing meaningful trends. in the next chapter will shall extract such trends manually to identify such sections.

3.5.6 Identification of the fault initiation point

So far we insisted on not dividing the the bearing life into Health Stages (HS). Health Indicators in RUL estimation give a varying trends in degradation as the remaining useful life gets lower and lower. The degradation history of the machinery should be split into HSs based on the trends of the health indicator, whatever it might be[47]. As researches suggest, because there is no information about the degradation trend in the healthy stage, it is difficult and unnecessary to predict the RUL during this stage. The RUL prediction should be triggered once we enter the unhealthy stage, which its beginning is defined as the first predicting time (FPT).[34] Many researches have been working on finding out when exactly the first defect signal pops out. [2] [48] [49] [50] In any case, the task of HS division is to detect the incipient degradation of machinery and provide a suitable FPT for RUL prediction. According to the previous researches a single degradation model is unable to describe the time-varying degradation trends. Therefore, it is necessary to divide the HIs into two or multiple HSs according to the change of their degradation trends and assign different models or missions to each stage.[15]

In fig 3.38 multiple stages are observed in the degradation process of a double row bearing, including a healthy stage, a degradation stage and a critical stage. The RMS values are stable during the healthy stage and then experience an “increase-decrease-increase” trend during the degradation stage. With the aggregation of damage, the RMS values increase rapidly and reach the failure threshold in short time. In such complex cases, a single degradation model is unable to describe the time-varying degradation trends. Therefore, it is necessary to divide the HIs into multiple HSs according to the change of their degradation trends and assign different models or missions to each stage. In the following two subsections, the publications related to the two-stage division and multi-stage division are summarized, respectively.

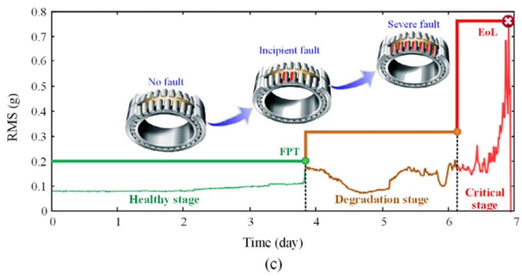

Figure 3.38: Degradation Process with multiple stages [15]

The number of division depending on the trends of the vibrational data might be two or more sections.[50] The easiest strategy would be a two stage division based on a threshold on HI.Wang et al.[2] for example, detected the initial point of a defect (FFP) for bearings when their RMS (root mean square) exceeded a pre-specified threshold.

In ISO-18436 [51] as another example, the following evaluation zones are defined to permit a qualitative assessment of the vibration of a given machine and to provide guidelines on possible actions. Zone A: The vibration of newly commissioned machines would normally fall within this zone. Zone B: Machines with vibration within this zone are normally considered acceptable for unrestricted long-term operation. Zone C: Machines with vibration within this zone are normally considered unsatisfactory for long-term continuous operation. Generally, the machine may be operated for a limited period in this condition until a suitable opportunity arises for remedial action. Zone D: Vibration values within this zone are normally considered to be of sufficient severity to cause damage to the machine, therefore in this research we shall asses the RUL in the C and D zones i.e. after where the first fault shows up.

3.5.7 Root mean square

In time domain studies The RMS (root mean square) value is generally the most useful because it is directly related to the energy content of the vibration data and therefore its destructive capability. [52] [21] On the other hand, RMS takes into account the wave form's time history . As it can be seen during the healthy stage, no fault occurs in the rolling element bearing and the RMS values present random fluctuations and During the unhealthy stage, RMS values increase with the deterioration of the bearing. Here we create and analyse the RMS vectors in order to find such points :

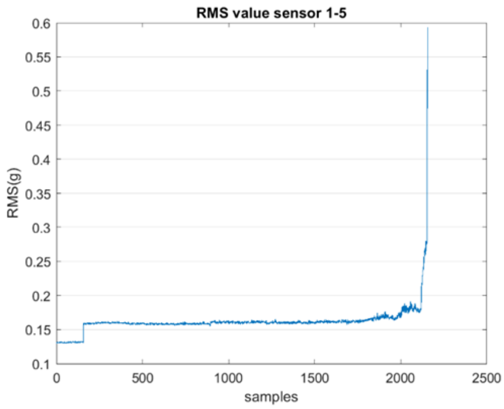

Figure 3.39: Global RMS value 1-5

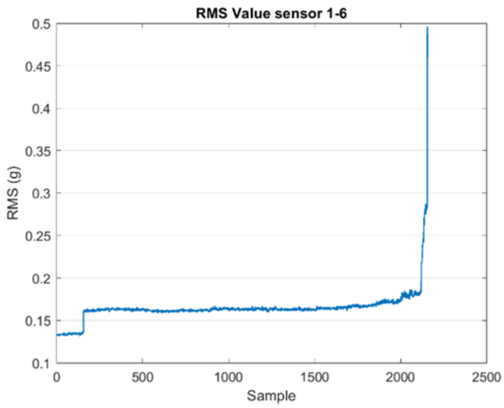

Figure 3.40: Global RMS value 1-6

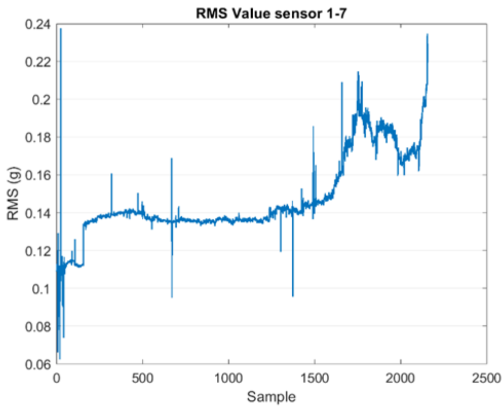

Figure 3.41: Global RMS value 1-7

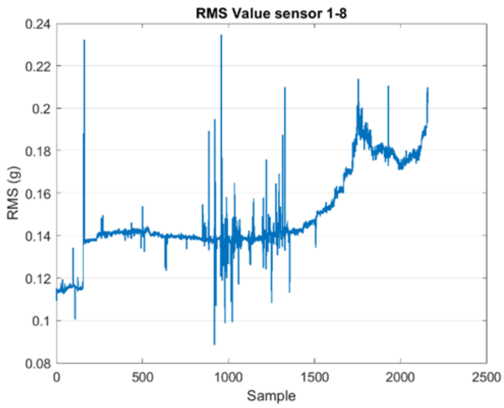

Figure 3.42: Global RMS value 1-8

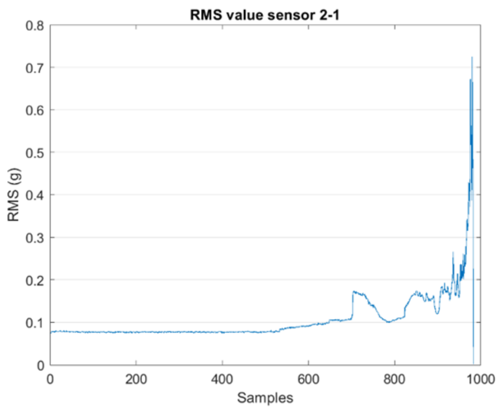

Figure 3.43: Global RMS value 2-1

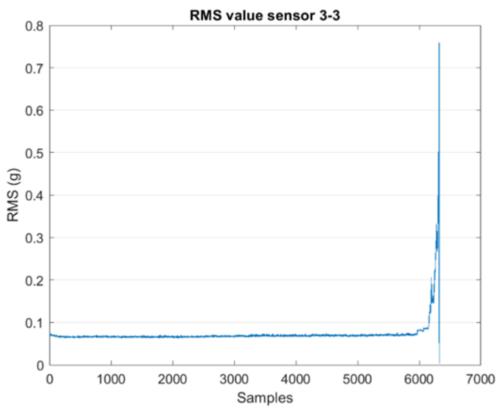

Figure 3.44: Global RMS value 3-3

As it can be seen from the RMS value graphs, the first fault signals are located in the final part of the bearing RUL.

Lets now check another method utilised in the papers to see if we can discover another HS division mode:

3.5.8 Power spectral density (PSD)

James K. Kimotho et al. [28] in addition to the RMS have used Power Spectral Density in their research to define the FFS and estimate the HSs. Likewise, We have used the Power spectral density (PSD) to convert time domain data into frequency domain data. A plot of the frequencies at the peak amplitude of the PSD spectrum against time revealed existence of different health states, where a new set of frequencies appeared or disappeared in the lifetime of the bearing [46] , as shown in Figures below:

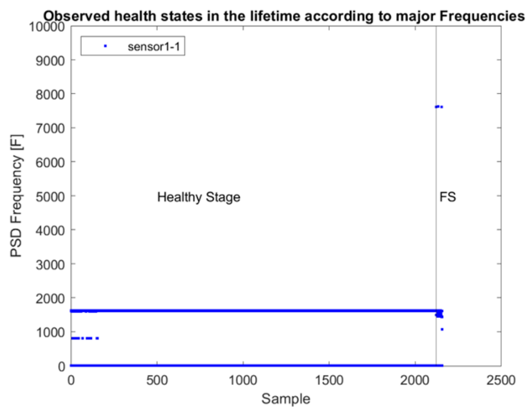

Figure 3.45: Appearance of Important PSD frequencies over time 1-1

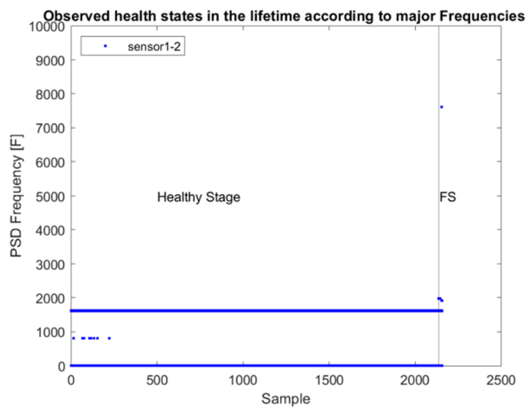

Figure 3.46: Appearance of Important PSD frequencies over time 1-2

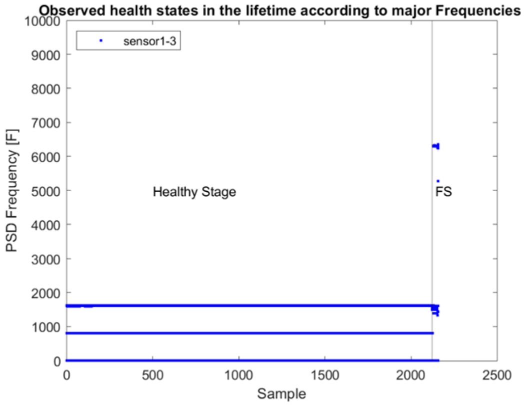

Figure 3.47: Appearance of Important PSD frequencies over time 1-3

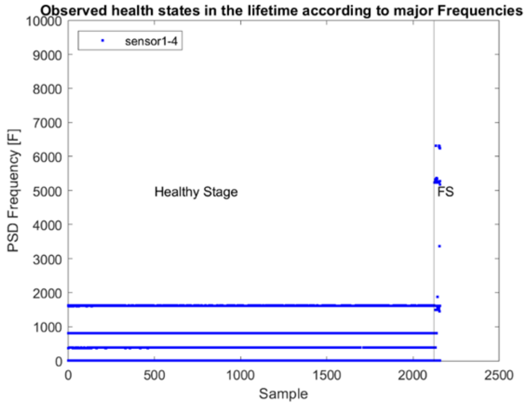

Figure 3.48: Appearance of Important PSD frequencies over time 1-4

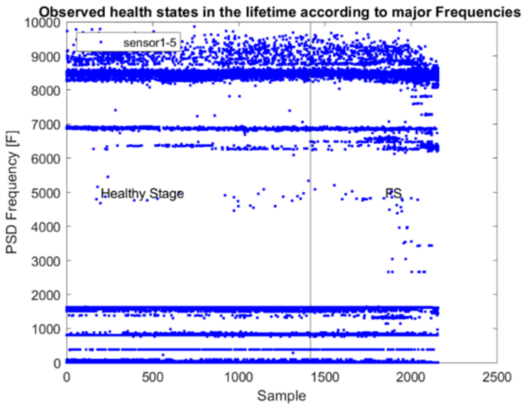

Figure 3.49: Appearance of Important PSD frequencies over time 1-5

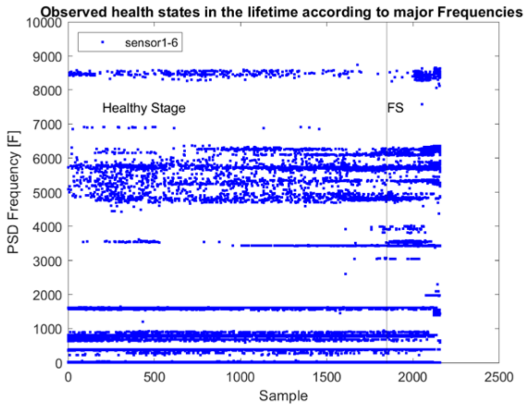

Figure 3.50: Appearance of Important PSD frequencies over time 1-6

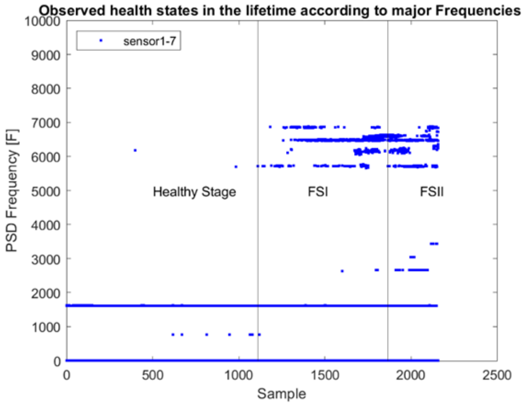

Figure 3.51: Appearance of Important PSD frequencies over time 1-7

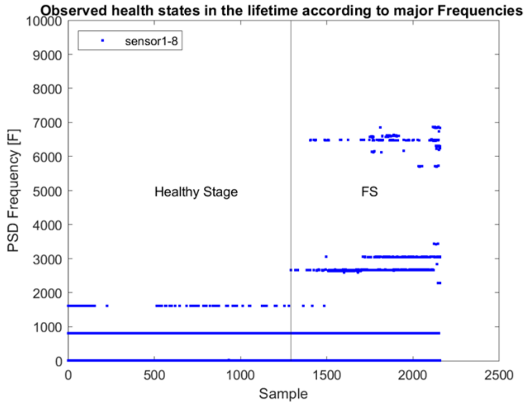

Figure 3.52: Appearance of Important PSD frequencies over time 1-8

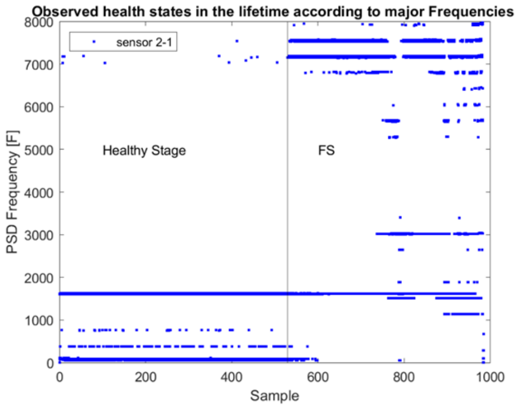

Figure 3.53: Appearance of Important PSD frequencies over time 2-1

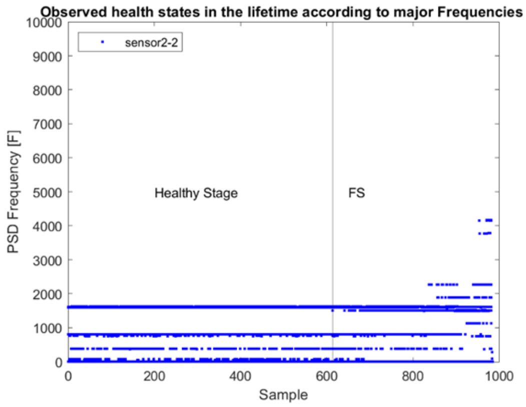

Figure 3.54: Appearance of Important PSD frequencies over time 2-2

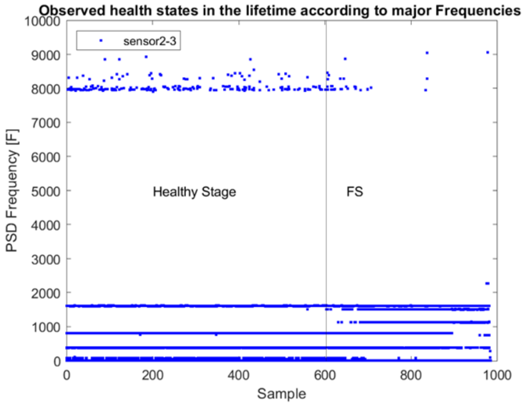

Figure 3.55: Appearance of Important PSD frequencies over time 2-3

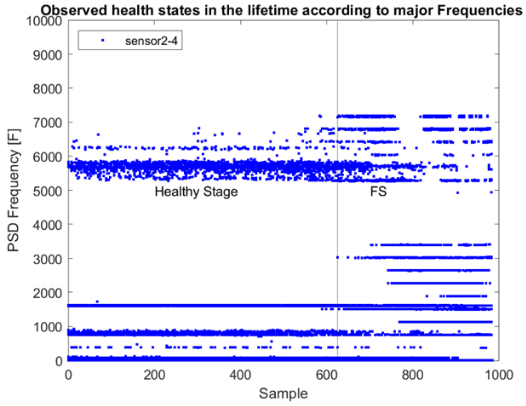

Figure 3.56: Appearance of Important PSD frequencies over time 2-4

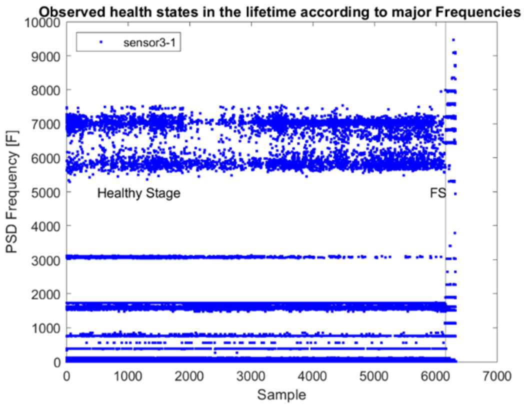

Figure 3.57: Appearance of Important PSD frequencies over time 3-1

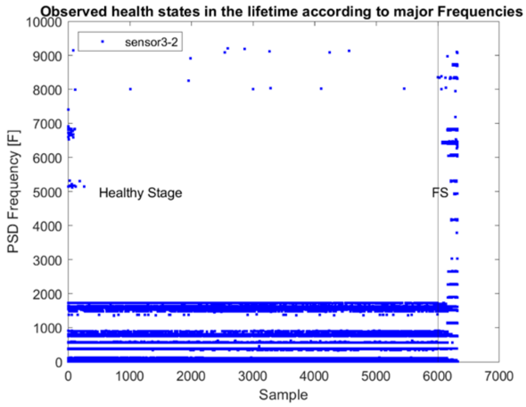

Figure 3.58: Appearance of Important PSD frequencies over time 3-2

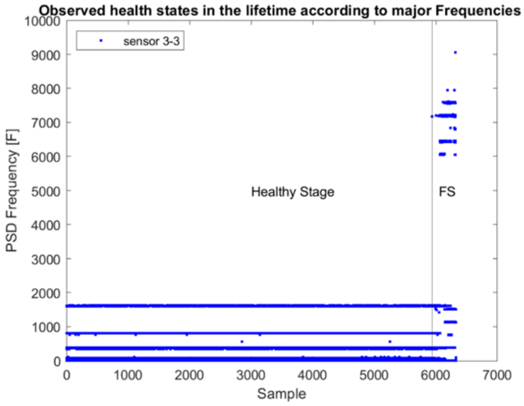

Figure 3.59: Appearance of Important PSD frequencies over time 3-3

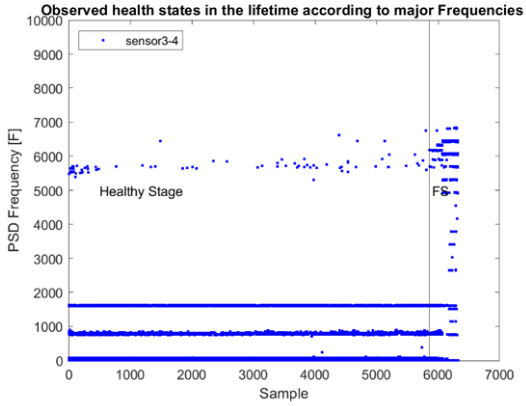

Figure 3.60: Appearance of Important PSD frequencies over time 3-4

3.5.9 High Frequency Crest Factor (HFCF)

Among time domain features, some other papers suggest the High Frequency Crest Factor(HFCF) as the Health indicator in bearing wear indication.[53]

HFCF is a time domain parameter of any alternating signal that demonstrates the ratio between the peak values of each signal to its effective value. The HCFC gives a sense of the extremity of the maximum values of a signal. HFCF equal to 1 means there are no peaks such as square waves or DC current. In the contrary machine vibration data have values often larger than 1. The higher the peaks are the bigger the HFCF value will be.

HFCH is calculated by dividing the absolute value of the maximum amplitude of the signal (in our case each of the time samples (1s)) to its RMS value which is equivalent to its ratio of L norm to the L2 norm of signal's function.

$$HFCF = \frac{|x_{peak}|}{RMS}$$

The results of the HFCF calculations for the failed bearings are shown below:

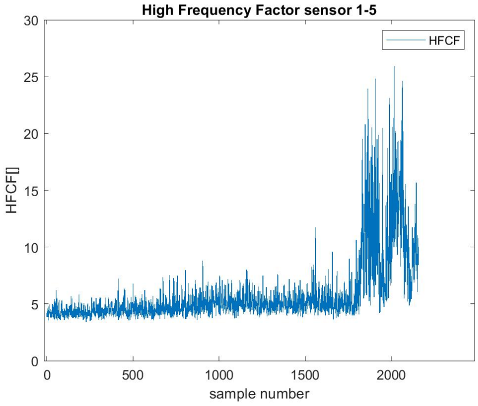

Figure 3.61: global HFCF value 1-5

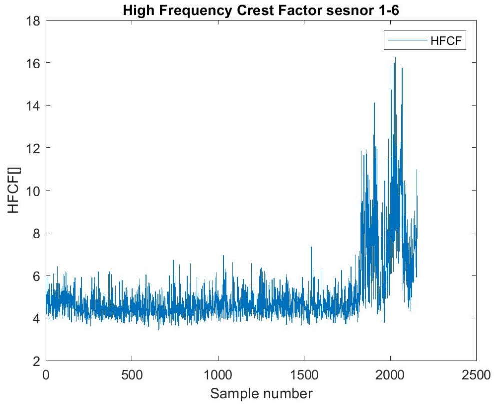

Figure 3.62: global HFCF value 1-6

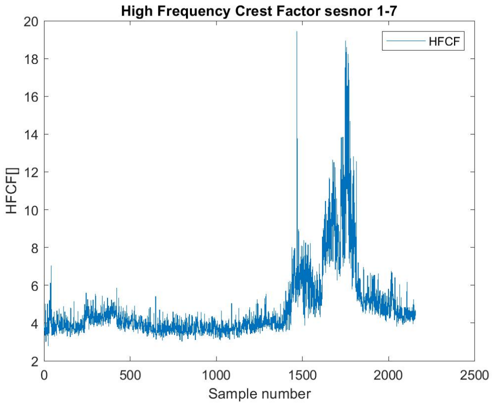

Figure 3.63: global HFCF value 1-7

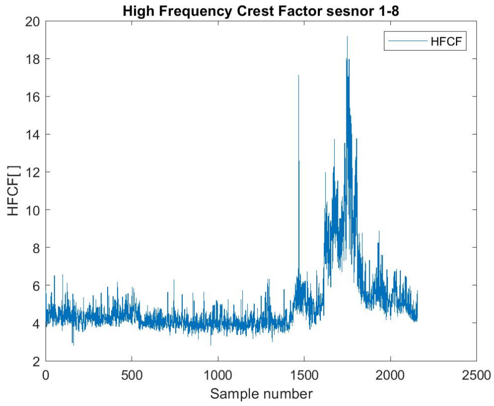

Figure 3.64: global HFCF value 1-8

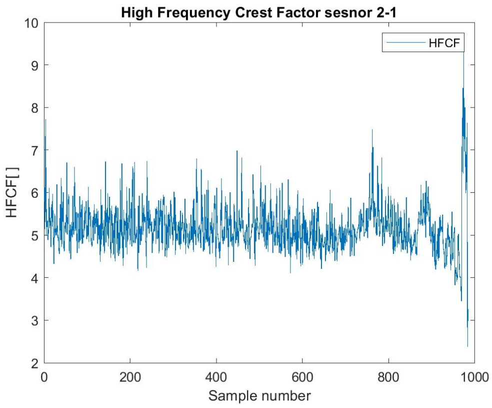

Figure 3.65: global HFCF value 2-1

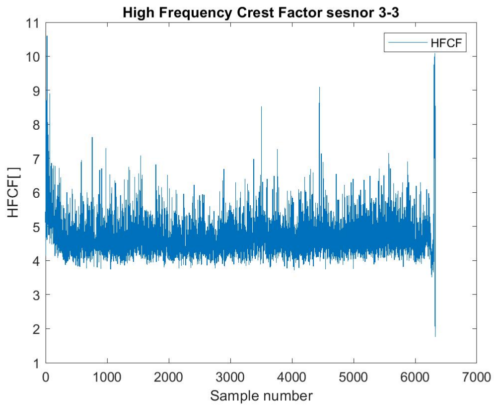

Figure 3.66: global HFCF value 3-3

The healthy and faulty stages for the first test are very well distinguishable, however for the 2nd and 3rd test graphs do not contain a meaningful trend.

In the following table, the first fault signal (FFS) is written for each sensor using two different indicators.

Sensor	1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8
RMS	NA	NA	NA	NA	2001	1994	1612	1612
PSD	2121	2134	2121	2125	1789	1855	1865	1291

Sensor	2-1	2-2	2-3	2-4	3-1	3-2	3-3	3-4
RMS	709	NA	NA	NA	5996	NA	NA	NA
PSD	529	613	603	625	6037	6004	5943	5857

Table 3.8: First Fault initiation points in different bearings using RMS and PSD methods

Training considering the FFS

Ramp labelling

Here we change the labelling mode as it can be seen in fig3.67. According to the FFS we shall consider the state of the bearing as healthy (i.e. RUL=1) before the first fault signal appears and after that the RUL starts decreasing linearly.

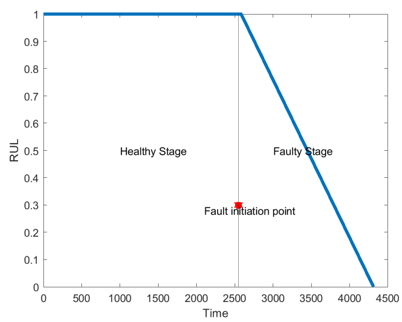

Figure 3.67: Graph of the training progress

The results of such training process is presented down below:

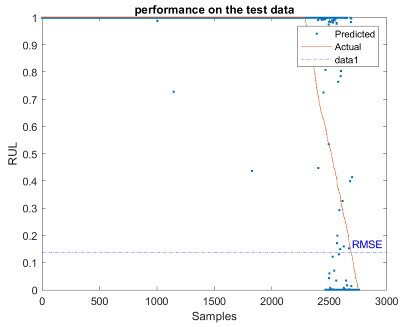

Figure 3.68: Performance of the network on test images

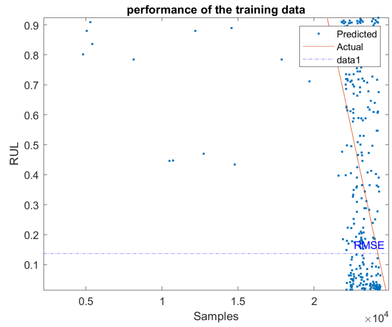

Figure 3.69: Performance of the network on training images

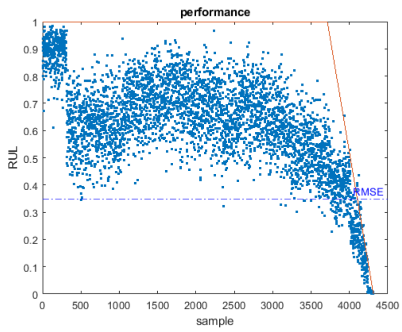

Figure 3.70: Performance of the network on a sensor that didn't take part in training RMSE=0.3471

The outcomes are not acceptable so we shall try another method.

quadratic labelling

Looking at the predictions so far, it is evident that the predictions for the early stages are far lower than actual values, this is because the spectrograms have insignificant gradient of changes in the early life periods. In this section we are going to label the images with their RUL quadratically descending as it can be seen in fig 3.71:

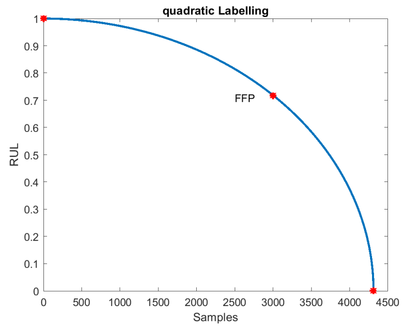

Figure 3.71: Quadratic labelling: the curve is fit to the first fault signal point

in such labeling format the fault initiation point is put at the 45° arc to change the slope of degradation as soon as the fault appears. the results for such labeling format are shown below;

Figure 3.72: Performance on test images, rmse = 0.0905

Figure 3.73: Performance on training images,rmse = 0.0874

From the results it is clear that a quadratic model of degradation fits the trend of changes in the images images quiet well, therefore the RMSE values increase significantly (around 10 percents).

Figure 3.74: Performance of the network on the sensor it has never seen (1-6),rmse = 0.1029

However the performance of the network in the final period of the bearings life for the sensors our network has never seen before is not acceptable. this is because the real degradation trend doesn't fully overlap with the proposed model. In the next trial, we will fit a quadratic curve on the bearings' degradation trend based on their RMS values.

Curve fitting on the RMS value

In the literature [54] the RMS value is considered as as a very powerful indicator of machinery health. This made us to base our health degradation profile on the RMS increase profile.

In this part we tried to fit the degradation trend (RUL decrease) on the RMS growth trend to see what the results were. by flipping the values as seen in image 3.75 and fitting a curve by 3 or more points on them we can have a good estimation of the degradation. Such degradation is shown in figure 3.76 . Such labeling might help with the correct RUL decrease profile.

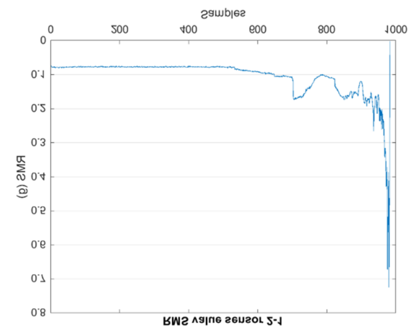

Figure 3.75: Graph of RMS flipped vertically

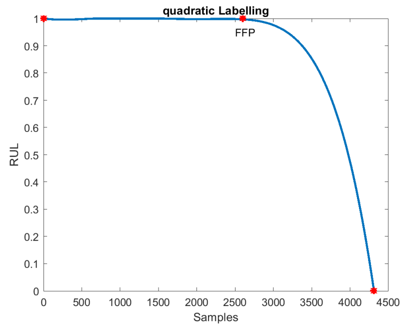

Figure 3.76: Curve fitting on the RMS value sensotr 1-7

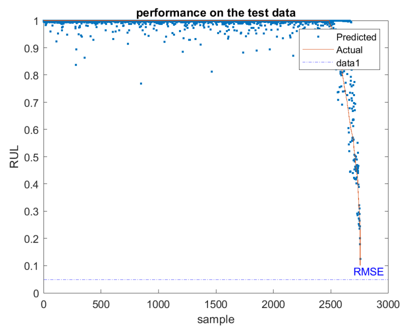

Figure 3.77: Performance on test images, rmse = 0.0484

Figure 3.78: Performance on training images, rmse = 0.0402

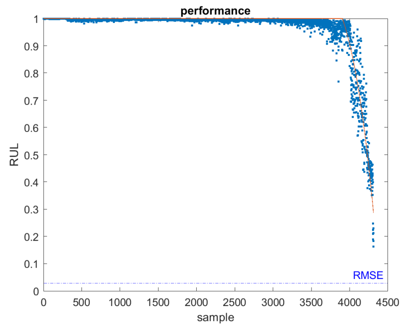

Figure 3.79: Performance of the network on the sensor it has never seen (1-6), rmse = 0.0491

3.5.10 Isolation of the Faulty Signal

As seen, The previous labelling highly overfit the network and diminished the precision so We have decided to develop another method. This time, input images go through a flowchart initially and then based on the outcome the next step is taken. The proposed flowchart can be seen in graph 3.80 [55] HS division performs a service role for RUL estimation. [2] In our case Two-stage division provides a FFP for RUL prediction. Multi-stage division however, may help with the RUL estimation process following these three steps. Firstly, a RUL estimation problem is transformed into a multi-stage classification through the multi-stage division, therefore various multi-class classification techniques can be applied to the area of RUL prediction [55] [56]. Next, different tasks are assigned into different stages after HS division, such as condition monitoring in the normal stage, one-step prediction in the slight degradation stage, RUL prediction in the severe degradation stage and shutdown in the failure stage. Third, due to the variation of the degradation trends in different stages, multi-model prediction is expected to perform better than a single prognostic model.

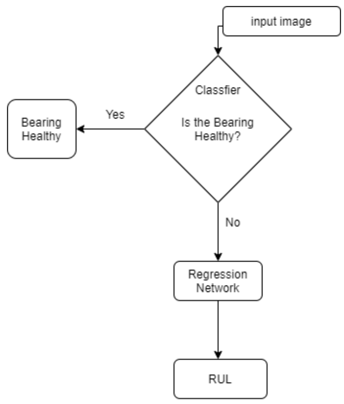

Figure 3.80: proposed workflow for the decision tree model

In this Model, the input image first enters a classifier network. which was trained based on the FFS dividing the dataset into two healthy and faulty sub datasets. this classifier network was again trained on the alexnet. The Trained classifier is 98 percent accurate and the confusion matrix of this classifier is presented below. 3.81

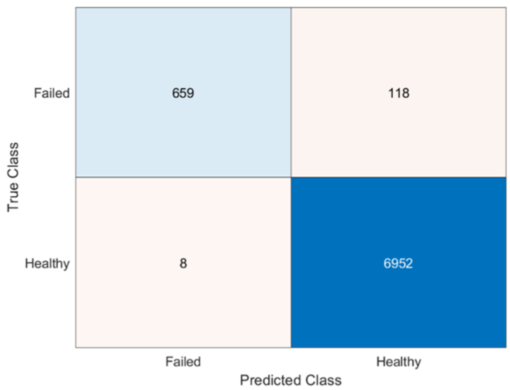

Figure 3.81: The confusion matrix of the proposed binary classifier

Next, if the bearing is faulty the image will enter a regression network to estimate the RUL. This way, only the spectrograms containing the faulty signals might enter the HI indicator yielding accurate results.

3.6 Conclusion

This thesis discovers potentials of Deep Learning in field of Predictive Maintenance and Prognostics with different settings and datasets. The biggest challenge with our dataset is its reality and therefore non-linearity which results in a non-homogeneous trend of degradation and lack of useful signals for training. However, when the network is trained and tested on the same sensors the results are up to 95 percent accurate but when trained and tested on a different sensors unknown to the trained network, the accuracy drastically plummets. This is mainly because the signal including at least one defect type, namely, outer race failure, make a very little percentage of the whole faulty signals (only about 2 percent of whole dataset). By using different methods and settings of creating spectrograms and different normalizations we attempted to extract meaningful trends from the data and in order to decrease the overfitting effect we tried augmentation, separating the failed sensors and training the network only with the failed ones. And finally to increase the accuracy we isolated the faulty signals of the failed sensors for training only. Fortunately, each of these steps positively contributed to the overall accuracy of the network but due to the challenges mentioned before, having a general network for all defect types, seems to be very challenging, using non-supervised training methods.

3.6.1 Discussion

Overall, if the network is trained in a correct manner, this method can acceptably predict the failure in the final stages of bearing life, whether the sensor data related to the tested bearing participated in the training process or not. Predicting the RUL in the early stages of the lifetime however prove to be quiet challenging due to lack of useful data since the signal to noise ratio is quite low in the early stages.

On the other hand, bearings are prone to different types of defects, each of which demonstrating a different degradation trend, fault frequency band and amplitude. This is another challenge to be thought of when we are trying to train a unique model for different fault types. Almost all the literature successfully estimating the RUL have clustered the bearings by defect type when the defect type was known or by similar degradation patterns when the defects were unknown. In this thesis we avoided doing so mainly because of scarcity of useful data for different fault types but creating an algorithm in which first the fault is classified and then the RUL estimator model is activated according to the defect can greatly help the outcome. furthermore, It is a common phenomenon to discover multiple faults in a single bearing in real applications. However, this is often ignored in academic researches for simplification.

In the literature, the HS division accuracy is in general elaborated subjectively and is not verified by the actual process of degradation of the machinery. This is because because it is very challenging to monitor the real degradation stages of the machine. The HSs in are segmented according to the changes of signal characteristics but different methods we utilised in this research yielded different fault initiation points. firstly, such points were not the same when using different methods. for example the fault initiation point for sensor 1-5 using the RMS threshold method,turns out to be the sample number 2001, however when the PSD method is used, such point is sample number 1789 for which I couldn't find a clear explanation. secondly for some of the sensors (i.e. sensors 2-1,3-3) such points didn't show up until the very last moments. this impedes utilisation of a great deal (about 99% in sensor 3-3) of the data. more work should be done on this issue.

In addition, the Particular features of the IMS data set need to be addressed here. Overall the for most of the sensors there are no linear degradation trends and in addition to that the self-healing effect present in the first dataset are great challenges to be tackled. furthermore, In the early stages of the bearing life, sensors have registered an unusual sudden shift in the RMS value in the first test which can be obviously noticed in the spectrograms.

3.7 The App

As a practical outcome of this work, I have proposed the development of a standalone application that could be separately installed on computer devices to estimate the RUL by reading a .CSV file, or connecting to a sensor to receive the time domain data and then creating the spectrogram and finally feeding into the CNN trained during the development of this thesis and follows the final algorithm proposed by me.

3.7.1 instructions of use

once the application has loaded, In the first tab, if the vibration data is saved before by clicking on the brows button user might upload the pre-recorded .CSV file obtained from the accelerometer in case we have a 3-axis accelerometer the desired channel number must be defined and then in the second tab by clicking on the Run button the RUL is estimated.

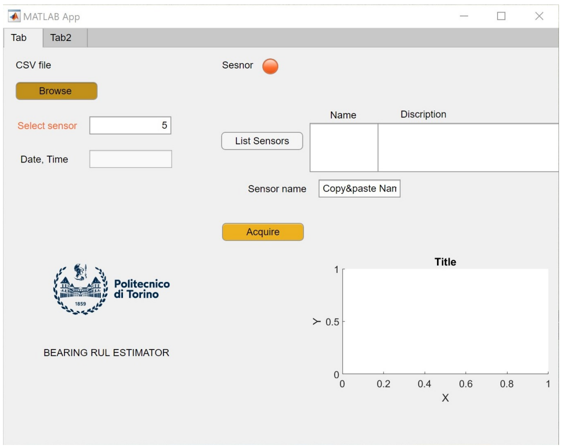

Figure 3.82: Input data port or file selection tab

Another functionality, is connecting the Acquisition interface to the computer. In this case user has to click on list sensors and when the sensor light turns into green the desired sensor might be selected by copying and pasting its name to the sensor name field. (The duration of acquisition(0.5s) and sampling frequency (20480Hz) is preset by us). Once the sensor is selected, the acquisition might be done by clicking on acquire button. once the data is uploaded, the time domain graph is shown and then again in the second tab by clicking on Run button the RUL is calculated.

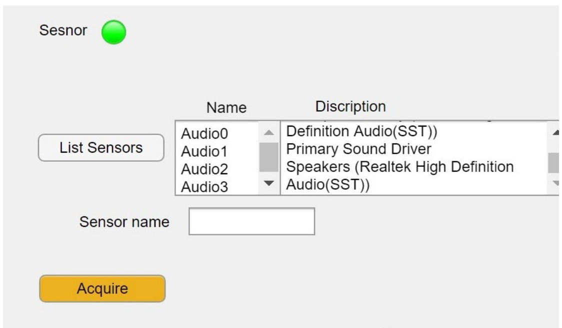

Figure 3.83: Acquisition sensors

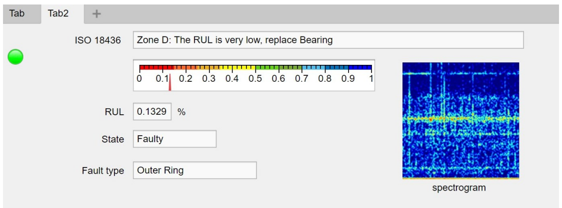

Figure 3.84: RUL calculation tab

3.7.2 Extension of use

Deep learning has a limited ability to be generalized across domains different than the original one. Specifically speaking, to transfer knowledge from the target domain to another domain unseen by the model is a quite difficult problem and is not discussed in this literature. Presenting solutions [57] to this problem could help with many real world issues as well as the RUL estimation.

Because Vibration analysis is the most common monitoring method used in the industry, the developed model has the potential to be utilised in a wider domain under a similar monitoring process. By implementing a cross domain strategy it is possible to generalize the functionality of such an app on other systems with dynamic and mechanical characteristics other than the dataset used in this thesis. The design of the cross domain strategy is outside the scope of this thesis.

Appendix A

MATLAB codes

A.0.1 Spectrogram Creator

```
    clc, clear
opengl('save', 'software')
fs = 20E3;            % unit: Hz
time = linspace(0,1,fs)';
fds = fileDatastore("1st_test\", "ReadFcn", @myCustomReader);
length(fds.Files);
for i=1:length(fds.Files)
H=importdata(fds.Files[1]);
    for j=1:4
h=H(:,j);
[~,fvec,tvec,P0] = spectrogram(h,256,230,512,fs);
fig=figure;
clf;
ah = axes('Units','Normalize','Position',[0 0 1 1]);
imagesc(tvec,fvec,P0);
axis(ah, 'square');
fig.Position(3) = fig.Position(4);  %set width equal to height
axis xy;
lim = caxis;
caxis([0 12e-4]);
axis off;
colormap jet;
colorbar
set(gca,'ColorScale','log');
colorbar
title("");
figname = append('spctrgrm',num2str(j),num2str(i),'.jpg');
saveas(fig,figname);
close all;
```

```
    end
end
```

A.0.2 Codes for training

```
clc, clear
opengl('save', 'software')

1st test
%to perform labeling in a regression problem we need to have table-
    type data directory
%I chose the 1st 3rd 4th 5th 7th 8th sensors for training and 2nd 6th
    for
%cross test. of course I also considered 10% of each sensor data for
    self testing.
% fds11 = fileDatastore("1-1 logscale\" ,"ReadFcn", @myCustomReader);
% Spectrogram11=natsortfiles(fds11.Files); %order data in natural
    order instead of default 1 10 100 2 20 200 etc...
% fds13=fileDatastore('1-3 logscale\',"ReadFcn", @myCustomReader);
% Spectrogram13=natsortfiles(fds13.Files);
% fds14=fileDatastore('1-4 logscale\',"ReadFcn", @myCustomReader);
% Spectrogram14=natsortfiles(fds14.Files);
fds15=fileDatastore('1-5\',"ReadFcn", @myCustomReader);
Spectrogram15=natsortfiles(fds15.Files);
fds17=fileDatastore('1-7\',"ReadFcn", @myCustomReader);
Spectrogram16=natsortfiles(fds17.Files);
fds18=fileDatastore('1-8\',"ReadFcn", @myCustomReader);
Spectrogram18=natsortfiles(fds18.Files);
%sensor 1-5

load("rul15fitted.mat");
Rul15_1=rul15
%sensor 1-7
load('rul17fitted.mat');
Rul17_1=rul17
%sensor 1-8
load('rul18fitted.mat');
Rul18_1=rul18

2nd test
fds21 = fileDatastore("2-1\" ,"ReadFcn", @myCustomReader);
Spectrogram21=natsortfiles(fds21.Files);
% fds22 = fileDatastore("2-2 logscale\" ,"ReadFcn", @myCustomReader);
% Spectrogram22=natsortfiles(fds22.Files);
% fds24 = fileDatastore("2-4 logscale\" ,"ReadFcn", @myCustomReader);
% Spectrogram24=natsortfiles(fds24.Files);
```

```
load('rul21fitted.mat');
Rul21_1=rul21

3rd test
fds33 = fileDatastore("3-3\" ,"ReadFcn", @myCustomReader);
Spectrogram33=natsortfiles(fds33.Files);
% fds32 = fileDatastore("3-2 logscale\" ,"ReadFcn", @myCustomReader);
% Spectrogram32=natsortfiles(fds32.Files);
% fds34 = fileDatastore("3-4 logscale\" ,"ReadFcn", @myCustomReader);
% Spectrogram34=natsortfiles(fds34.Files);
load('rul33fitted.mat');
Rul33_1=rul33

Spectrogram=[Spectrogram15;Spectrogram16;Spectrogram18;Spectrogram21;
    Spectrogram33];
Rul=[Rul15_1;Rul17_1;Rul18_1;Rul21_1;Rul33_1];
Ttes=table(Spectrogram,Rul);
% %create the training files and test files
numfiles=length(Rul);
my_indices = randperm(numfiles);
testset = Ttes(my_indices(1:floor(0.1*length(Rul))),:);
trainset = Ttes(my_indices(ceil(0.1*length(Rul)):end),:);

load('coollayer.mat')
layers_2(1:16)=freezeWeights(layers_2(1:16));
net=layers_2



%training options
options = trainingOptions("adam", ...
    "Plots","training-progress", 'InitialLearnRate',0.00001, ...
    "LearnRateSchedule","piecewise",'Shuffle','every-epoch',...
    "LearnRateDropPeriod",1,'LearnRateDropFactor',0.1...
    ,"MaxEpochs",5,'MiniBatchSize',20);

Faultyonly = trainNetwork(trainset,net,options)

testds = augmentedImageDatastore([227 227],testset);
predstest = predict(Faultyonly,testds);

testvals = testset.Rul; %actual test Values
error = testvals - predstest;

squares = error.^2;
rmse = sqrt(mean(squares))


%test the network on test set
```

```
figure(2)
Ttest=table(testset.Spectrogram,testset.Rul,predstest);
TTe = sortrows(Ttest,'Var2','descend');
pred_sort = TTe.Var3;
% smooth = smoothdata(pred_sort);
test_val_sort = TTe.Var2;
% error =abs(test_val_sort -pred_sort);
% errsmooth=smoothdata(error);
plot(pred_sort,".",'DisplayName','Predicted');
hold on
plot(test_val_sort,'DisplayName','Actual');
% plot(error,'DisplayName','error');
titolo=append('performance on the test data');
title (titolo);
yline(rmse,'-.b','RMSE');
legend();
ylabel('RUL')
xlabel('Samples')
hold off

%test network on training set
trainds=augmentedImageDatastore([227 227],trainset)
predstrain= predict(Faultyonly,trainds)
Ttrain=table(trainset.Spectrogram,trainset.Rul,predstrain);
TTr = sortrows(Ttrain,'Var2','descend');
pred_sort_train = TTr.Var3;
train_val_sort = TTr.Var2;
error_train =abs(train_val_sort -pred_sort_train);

trainvals = trainset.Rul; %actual test Values
errort = trainvals - predstrain;

squarest = errort.^2;
rmset = sqrt(mean(squarest))

figure(5)
plot(pred_sort_train,".",'DisplayName','Predicted');
hold on
plot(train_val_sort,'DisplayName','Actual');
% plot(error_train,'DisplayName','error');
titolotr=append('performance of the training data');
title (titolotr);
yline(rmse,'-.b','RMSE');
legend();
ylabel('RUL')
xlabel('Samples')
```

A.0.3 Codes for the APP

```
classdef FINALE < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                    matlab.ui.Figure
        TabGroup                    matlab.ui.container.TabGroup
        Tab                         matlab.ui.container.Tab
        BrowseButton                matlab.ui.control.StateButton
        SelectsensorEditFieldLabel  matlab.ui.control.Label
        SelectsensorEditField       matlab.ui.control.
    NumericEditField
        DateTimeEditFieldLabel      matlab.ui.control.Label
        DateTimeEditField           matlab.ui.control.EditField
        CSVfileLabel                matlab.ui.control.Label
        SesnorLabel                 matlab.ui.control.Label
        ListSensorsButton           matlab.ui.control.Button
        ListTextArea                matlab.ui.control.TextArea
        SensornameEditFieldLabel    matlab.ui.control.Label
        SensornameEditField         matlab.ui.control.EditField
        Lamp_2                      matlab.ui.control.Lamp
        AcquireButton               matlab.ui.control.Button
        modelTextArea               matlab.ui.control.TextArea
        NameLabel                   matlab.ui.control.Label
        DiscriptionLabel            matlab.ui.control.Label
        UIAxes2                     matlab.ui.control.UIAxes
        Tab2                        matlab.ui.container.Tab
        Image                       matlab.ui.control.Image
        spectrogramLabel            matlab.ui.control.Label
        Lamp                        matlab.ui.control.Lamp
        RULGauge                    matlab.ui.control.LinearGauge
        Label                       matlab.ui.control.Label
        StateEditFieldLabel         matlab.ui.control.Label
        StateEditField              matlab.ui.control.EditField
        ISO18436EditFieldLabel      matlab.ui.control.Label
        ISO18436EditField           matlab.ui.control.EditField
        RULEditFieldLabel           matlab.ui.control.Label
        RULEditField                matlab.ui.control.EditField
        FaulttypeEditFieldLabel     matlab.ui.control.Label
        FaulttypeEditField          matlab.ui.control.EditField
        UIAxes                      matlab.ui.control.UIAxes
    end

    % Callbacks that handle component events
    methods (Access = private)

```

```
        % Value changed function: BrowseButton
        function BrowseButtonValueChanged(app, event)
            value = app.BrowseButton.Value;
            [FileName,PathName,FilterIndex] = uigetfile('');
            app.DateTimeEditField.Value=FileName;
            H=importdata(FileName);
            sensornum=app.SelectsensorEditField.Value;
            h=H(:,sensornum);
            time=1:1:length(h);
            plot(app.UIAxes,time,h);
            ylabel('m/s2')
            xlabel('t[s]')
            xlim([0 20480])
            fs=20480;
            [~,fvec,tvec,P0] = spectrogram(h(1:10240),256,230,512,fs)
    ;
            % fig=figure('Menu','none','ToolBar','none');
            figu=figure;
            clf;
            % fh = figure('Menu','none','ToolBar','none');
            ah = axes('Units','Normalize','Position',[0 0 1 1]);
            imagesc(tvec,fvec,P0);
            axis(ah, 'square');
            figu.Position(3) = figu.Position(4);  %set width equal to
     height
            axis xy;
            lim = caxis;
            caxis([1e-6 1e-3]);
            axis off;
            colormap jet;
            set(gca,'ColorScale','log');
            colorbar off;
            title("");
            F = getframe(gcf);
            [X, Map] = frame2im(F);
            fig=imresize(X,[227 227]);
            app.Image.ImageSource=fig;
            load ('Faridclassifier.mat');
            cl=classify(Faridclassifier,fig);
            cl=char(cl(1));
            app.StateEditField.Value=cl;
            if strcmp(cl,'Failed')
                load("typeclassifier.mat")
                fl=classify(Faridtypeclassifier,fig);
                fl=char(fl(1));
                app.FaulttypeEditField.Value=fl;
                load('Faultyu.mat');
                 pred=predict(Faultyonly,fig);
                 if pred>=0.9
```

```
                    app.Lamp.Color=[0.4660 0.6740 0.1880];
                    app.ISO18436EditField.Value='Zone has been finished';
                    elseif pred<0.9 && pred>=0.5
                            app.Lamp.Color='yellow';
                             app.ISO18436EditField.Value='Zone B: The RUL is is high enough';
                    elseif pred<0.5 && pred>0.1
                        app.Lamp.Color=[0.8500 0.3250 0.0980];
                        app.ISO18436EditField.Value='Zone C: The RUL is less than half';
                    else
                          app.Lamp.Color='r';
                         app.ISO18436EditField.Value='Zone D: The RUL is very low, replace Bearing';
                    end
               app.RULGauge.Value=pred*100;
               app.RULEditField.Value=num2str(pred);

               elseif strcmp(cl,'Healthy')
                   app.ISO18436EditField.Value='Zone A: No action needed';
                   app.Lamp.Color='g';
                   app.RULGauge.Value=1;
                   app.RULEditField.Value='100';
                   app.FaulttypeEditField.Value='None';
               end
           end

           % Button pushed function: ListSensorsButton
           function ListSensorsButtonPushed(app, event)
            sensor=daq.getDevices
            x={sensor(1,:).ID};
            app.ListTextArea.Value=x;
            app.Lamp_2.Color='g'
            m={sensor(1,:).Model}
            app.modelTextArea.Value=m;
           end

           % Button pushed function: AcquireButton
           function AcquireButtonPushed(app, event)

name=app.SensornameEditField.Value

%% Create a data acquisition session
daqSession = daq.createSession('directsound');

%% Add channels specified by subsystem type and device
daqSession.addAudioInputChannel(name,'1');
```

```
daqSession.addAudioInputChannel('Audio1','2');

%% Configure properties
daqSession.Rate = 20480;
daqSession.Rate = 20000;


%% Configure properties
daqSession.DurationInSeconds = 1;

%% Run the data acquisition session
[data,time] = daqSession.startForeground();

plot(app.UIAxes2,time,data)

%% Disconnect from the device
daqSession.release();
delete(daqSession);
clear daqSession;

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are
    created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 679 509];
            app.UIFigure.Name = 'MATLAB App';

            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.Position = [1 -22 699 532];

            % Create Tab
            app.Tab = uitab(app.TabGroup);
            app.Tab.Title = 'Tab';

            % Create BrowseButton
            app.BrowseButton = uibutton(app.Tab, 'state');
            app.BrowseButton.ValueChangedFcn = createCallbackFcn(app,
     @BrowseButtonValueChanged, true);
            app.BrowseButton.Text = 'Browse ';
```

```
            app.BrowseButton.BackgroundColor = [0.9294 0.6941 0.1255];
            app.BrowseButton.Position = [16 444 100 22];

            % Create SelectsensorEditFieldLabel
            app.SelectsensorEditFieldLabel = uilabel(app.Tab);
            app.SelectsensorEditFieldLabel.HorizontalAlignment = 'right';
            app.SelectsensorEditFieldLabel.FontColor = [1 0.4118 0.1608];
            app.SelectsensorEditFieldLabel.Position = [13 402 78 22];
            app.SelectsensorEditFieldLabel.Text = 'Select sensor';

            % Create SelectsensorEditField
            app.SelectsensorEditField = uieditfield(app.Tab, 'numeric');
            app.SelectsensorEditField.Position = [106 402 100 22];
            app.SelectsensorEditField.Value = 5;

            % Create DateTimeEditFieldLabel
            app.DateTimeEditFieldLabel = uilabel(app.Tab);
            app.DateTimeEditFieldLabel.HorizontalAlignment = 'right';
            app.DateTimeEditFieldLabel.Position = [16 361 64 22];
            app.DateTimeEditFieldLabel.Text = 'Date, Time';

            % Create DateTimeEditField
            app.DateTimeEditField = uieditfield(app.Tab, 'text');
            app.DateTimeEditField.Editable = 'off';
            app.DateTimeEditField.Position = [106 361 101 22];

            % Create CSVfileLabel
            app.CSVfileLabel = uilabel(app.Tab);
            app.CSVfileLabel.Position = [16 476 49 22];
            app.CSVfileLabel.Text = 'CSV file';

            % Create SesnorLabel
            app.SesnorLabel = uilabel(app.Tab);
            app.SesnorLabel.Position = [268 476 47 22];
            app.SesnorLabel.Text = 'Sesnor ';

            % Create ListSensorsButton
            app.ListSensorsButton = uibutton(app.Tab, 'push');
            app.ListSensorsButton.ButtonPushedFcn = createCallbackFcn(app, @ListSensorsButtonPushed, true);
            app.ListSensorsButton.Position = [267 383 100 22];
            app.ListSensorsButton.Text = 'List Sensors';

            % Create ListTextArea
            app.ListTextArea = uitextarea(app.Tab);
```

```
            app.ListTextArea.Position = [375 356 84 60];

            % Create SensornameEditFieldLabel
            app.SensornameEditFieldLabel = uilabel(app.Tab);
            app.SensornameEditFieldLabel.HorizontalAlignment = 'right
';
            app.SensornameEditFieldLabel.Position = [294 325 77 22];
            app.SensornameEditFieldLabel.Text = 'Sensor name';

            % Create SensornameEditField
            app.SensornameEditField = uieditfield(app.Tab, 'text');
            app.SensornameEditField.Position = [386 325 100 22];
            app.SensornameEditField.Value = 'Copy&paste Name';

            % Create Lamp_2
            app.Lamp_2 = uilamp(app.Tab);
            app.Lamp_2.Position = [317 475 20 20];
            app.Lamp_2.Color = [1 0.4118 0.1608];

            % Create AcquireButton
            app.AcquireButton = uibutton(app.Tab, 'push');
            app.AcquireButton.ButtonPushedFcn = createCallbackFcn(app
, @AcquireButtonPushed, true);
            app.AcquireButton.BackgroundColor = [0.9294 0.6941
0.1255];
            app.AcquireButton.Position = [268 272 100 22];
            app.AcquireButton.Text = 'Acquire';

            % Create modelTextArea
            app.modelTextArea = uitextarea(app.Tab);
            app.modelTextArea.Position = [458 356 229 60];

            % Create NameLabel
            app.NameLabel = uilabel(app.Tab);
            app.NameLabel.Position = [400 415 38 22];
            app.NameLabel.Text = 'Name';

            % Create DiscriptionLabel
            app.DiscriptionLabel = uilabel(app.Tab);
            app.DiscriptionLabel.Position = [486 416 62 22];
            app.DiscriptionLabel.Text = 'Discription';

            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.Tab);
            title(app.UIAxes2, 'Title')
            xlabel(app.UIAxes2, 'X')
            ylabel(app.UIAxes2, 'Y')
            zlabel(app.UIAxes2, 'Z')
            app.UIAxes2.Position = [375 73 300 185];
```

```

            % Create Tab2
            app.Tab2 = uitab(app.TabGroup);
            app.Tab2.Title = 'Tab2';

            % Create Image
            app.Image = uiimage(app.Tab2);
            app.Image.Position = [493 314 157 145];

            % Create spectrogramLabel
            app.spectrogramLabel = uilabel(app.Tab2);
            app.spectrogramLabel.Position = [536 293 72 22];
            app.spectrogramLabel.Text = 'spectrogram';

            % Create Lamp
            app.Lamp = uilamp(app.Tab2);
            app.Lamp.Position = [8 456 20 20];

            % Create RULGauge
            app.RULGauge = uigauge(app.Tab2, 'linear');
            app.RULGauge.Limits = [0 1];
            app.RULGauge.ScaleColors = [0 0 1;0 0.4471 0.7412;0.302
    0.7451 0.9333;0.4667 0.6745 0.1882;0.3922 0.8314 0.0745;1 1
    0;0.9294 0.6941 0.1255;1 0.4118 0.1608;1 0 0];
            app.RULGauge.ScaleColorLimits = [0.9 1;0.8 0.9;0.7
    0.8;0.6 0.7;0.5 0.6;0.35 0.5;0.25 0.35;0.15 0.25;0 0.15];
            app.RULGauge.Position = [164 423 301 40];

            % Create Label
            app.Label = uilabel(app.Tab2);
            app.Label.Position = [219 387 25 22];
            app.Label.Text = '%';

            % Create StateEditFieldLabel
            app.StateEditFieldLabel = uilabel(app.Tab2);
            app.StateEditFieldLabel.HorizontalAlignment = 'right';
            app.StateEditFieldLabel.Position = [119 353 34 22];
            app.StateEditFieldLabel.Text = 'State';

            % Create StateEditField
            app.StateEditField = uieditfield(app.Tab2, 'text');
            app.StateEditField.Editable = 'off';
            app.StateEditField.Position = [164 353 104 22];

            % Create ISO18436EditFieldLabel
            app.ISO18436EditFieldLabel = uilabel(app.Tab2);
            app.ISO18436EditFieldLabel.HorizontalAlignment = 'right';
            app.ISO18436EditFieldLabel.Position = [86 476 63 22];
            app.ISO18436EditFieldLabel.Text = 'ISO 18436';
```

```

            % Create ISO18436EditField
            app.ISO18436EditField = uieditfield(app.Tab2, 'text');
            app.ISO18436EditField.Editable = 'off';
            app.ISO18436EditField.Position = [164 476 486 22];

            % Create RULEditFieldLabel
            app.RULEditFieldLabel = uilabel(app.Tab2);
            app.RULEditFieldLabel.HorizontalAlignment = 'right';
            app.RULEditFieldLabel.Position = [123 387 30 22];
            app.RULEditFieldLabel.Text = 'RUL';

            % Create RULEditField
            app.RULEditField = uieditfield(app.Tab2, 'text');
            app.RULEditField.Editable = 'off';
            app.RULEditField.Position = [164 387 48 22];

            % Create FaulttypeEditFieldLabel
            app.FaulttypeEditFieldLabel = uilabel(app.Tab2);
            app.FaulttypeEditFieldLabel.HorizontalAlignment = 'right'
    ;
            app.FaulttypeEditFieldLabel.Position = [95 314 58 22];
            app.FaulttypeEditFieldLabel.Text = 'Fault type';

            % Create FaulttypeEditField
            app.FaulttypeEditField = uieditfield(app.Tab2, 'text');
            app.FaulttypeEditField.Editable = 'off';
            app.FaulttypeEditField.Position = [164 314 154 22];

            % Create UIAxes
            app.UIAxes = uiaxes(app.Tab2);
            title(app.UIAxes, 'Time Domain')
            xlabel(app.UIAxes, 'Time [s]')
            ylabel(app.UIAxes, 'a [m/s2]')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [364 73 300 206];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = FINALE

            % Create UIFigure and components
```

```
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Bibliography

[1] Jürgen Schmidhuber. «Deep learning in neural networks: An overview». In: *Neural networks* 61 (2015), pp. 85–117 (cit. on p. iii).

[2] Wenbin Wang. «A model to predict the residual life of rolling element bearings given monitored condition information to date». In: *IMA Journal of management mathematics* 13.1 (2002), pp. 3–16 (cit. on pp. 1, 2, 4, 47, 48, 61).

[3] Weizhong Yan, Hai Qiu, and Naresh Iyer. *Feature Extraction for Bearing Prognostics and Health Management (PHM)-A Survey (Preprint).* Tech. rep. AIR FORCE RESEARCH LAB WRIGHT-PATTERSON AFB OH MATERIALS and MANUFACTURING ..., 2008 (cit. on pp. 1, 11, 12).

[4] Levent Eren. «Bearing fault detection by one-dimensional convolutional neural networks». In: *Mathematical Problems in Engineering* 2017 (2017) (cit. on pp. 1, 5).

[5] M. Mitchell Waldrop. «What are the limits of deep learning?» In: *Proceedings of the National Academy of Sciences of the United States of America* (Jan. 2019). URL: `https://www.pnas.org/content/116/4/1074` (cit. on p. 2).

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105 (cit. on pp. 3, 5, 6).

[7] Jerry Wei. *Alexnet: The architecture that challenged CNNS*. Sept. 2020. URL: `https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951` (cit. on pp. 3, 15).

[8] Dengsheng Lu and Qihao Weng. «A survey of image classification methods and techniques for improving classification performance». In: *International journal of Remote sensing* 28.5 (2007), pp. 823–870 (cit. on p. 3).

[9] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. «A review of feature selection methods on synthetic data». In: *Knowledge and information systems* 34.3 (2013), pp. 483–519 (cit. on p. 3).

[10] Yaguo Lei and Ming J Zuo. «Gear crack level identification based on weighted K nearest neighbor classification algorithm». In: *Mechanical Systems and Signal Processing* 23.5 (2009), pp. 1535–1547 (cit. on p. 3).

[11] *Predictive maintenance*. Sept. 2021. URL: `https://en.wikipedia.org/wiki/Predictive_maintenance` (cit. on p. 4).

[12] Yaguo Lei, Bin Yang, Xinwei Jiang, Feng Jia, Naipeng Li, and Asoke K Nandi. «Applications of machine learning to machine fault diagnosis: A review and roadmap». In: *Mechanical Systems and Signal Processing* 138 (2020), p. 106587 (cit. on p. 4).

[13] Xiang Li, Qian Ding, and Jian-Qiao Sun. «Remaining useful life estimation in prognostics using deep convolution neural networks». In: *Reliability Engineering & System Safety* 172 (2018), pp. 1–11 (cit. on pp. 5, 12).

[14] Robert X Gao and Ruqiang Yan. «Non-stationary signal processing for bearing health monitoring». In: *International journal of manufacturing research* 1.1 (2006), pp. 18–40 (cit. on p. 5).

[15] Yaguo Lei, Naipeng Li, Liang Guo, Ningbo Li, Tao Yan, and Jing Lin. «Machinery health prognostics: A systematic review from data acquisition to RUL prediction». In: *Mechanical systems and signal processing* 104 (2018), pp. 799–834 (cit. on pp. 5, 6, 13, 47).

[16] Bin Zhang, Lijun Zhang, and Jinwu Xu. «Degradation feature selection for remaining useful life prediction of rolling element bearings». In: *Quality and Reliability Engineering International* 32.2 (2016), pp. 547–554 (cit. on p. 5).

[17] Minghang Zhao, Baoping Tang, and Qian Tan. «Bearing remaining useful life estimation based on time–frequency representation and supervised dimensionality reduction». In: *Measurement* 86 (2016), pp. 41–55 (cit. on p. 5).

[18] Richmond Alake. *What alexnet brought to the world of Deep learning*. July 2020. URL: `https://towardsdatascience.com/what-alexnet-brought-to-the-world-of-deep-learning-46c7974b46fc` (cit. on p. 5).

[19] *Alexnet*. Oct. 2021. URL: `https://en.wikipedia.org/wiki/AlexNet` (cit. on p. 5).

[20] Sunil Tyagi and SK Panigrahi. «A simple continuous wavelet transform method for detection of rolling element bearing faults and its comparison with envelope detection». In: *Int. J. Sci. Res* 6 (2017), pp. 1033–1040 (cit. on p. 5).

[21] Yaguo Lei, Naipeng Li, and Jing Lin. «A new method based on stochastic process models for machine remaining useful life prediction». In: *IEEE Transactions on Instrumentation and Measurement* 65.12 (2016), pp. 2671–2684 (cit. on pp. 6, 48).

[22] Patrick Nectoux, Rafael Gouriveau, Kamal Medjaher, Emmanuel Ramasso, Brigitte Chebel-Morello, Noureddine Zerhouni, and Christophe Varnier. «PRONOSTIA: An experimental platform for bearings accelerated degradation tests.» In: *IEEE International Conference on Prognostics and Health Management, PHM'12*. IEEE Catalog Number: CPF12PHM-CDR. 2012, pp. 1–8 (cit. on p. 6).

[23] Vimala Mathew, Tom Toby, Vikram Singh, B Maheswara Rao, and M Goutham Kumar. «Prediction of Remaining Useful Lifetime (RUL) of turbofan engine using machine learning». In: *2017 IEEE International Conference on Circuits and Systems (ICCS)*. IEEE. 2017, pp. 306–311 (cit. on p. 6).

[24] Fei Shen, Chao Chen, Ruqiang Yan, and Robert X Gao. «Bearing fault diagnosis based on SVD feature extraction and transfer learning classification». In: *2015 Prognostics and System Health Management Conference (PHM)*. IEEE. 2015, pp. 1–6 (cit. on p. 6).

[25] Youngji Yoo and Jun-Geol Baek. «A novel image feature for the remaining useful lifetime prediction of bearings based on continuous wavelet transform and convolutional neural network». In: *Applied Sciences* 8.7 (2018), p. 1102 (cit. on p. 6).

[26] Siyu Shao, Stephen McAleer, Ruqiang Yan, and Pierre Baldi. «Highly accurate machine fault diagnosis using deep transfer learning». In: *IEEE Transactions on Industrial Informatics* 15.4 (2018), pp. 2446–2455 (cit. on p. 6).

[27] Yoav Freund and Robert E Schapire. «Large margin classification using the perceptron algorithm». In: *Machine learning* 37.3 (1999), pp. 277–296 (cit. on p. 13).

[28] Jason Brownlee. *Softmax activation function with python*. June 2020. URL: `https://machinelearningmastery.com/softmax-activation-function-with-python` (cit. on pp. 14, 35).

[29] Muhamad Yani et al. «Application of transfer learning using convolutional neural network method for early detection of terry's nail». In: *Journal of Physics: Conference Series*. Vol. 1201. 1. IOP Publishing. 2019, p. 012052 (cit. on p. 15).

[30] Xiaobing Han, Yanfei Zhong, Liqin Cao, and Liangpei Zhang. «Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification». In: *Remote Sensing* 9.8 (2017), p. 848 (cit. on p. 16).

[31] Pavan Kumar Kankar, Satish C Sharma, and Suraj Prakash Harsha. «Fault diagnosis of ball bearings using continuous wavelet transform». In: *Applied Soft Computing* 11.2 (2011), pp. 2300–2312 (cit. on p. 17).

[32] Hsiung-Cheng Lin, Yu-Chen Ye, Bo-Jyun Huang, and Jia-Lun Su. «Bearing vibration detection and analysis using enhanced fast Fourier transform algorithm». In: *Advances in Mechanical Engineering* 8.10 (2016), p. 1687814016675080 (cit. on p. 18).

[33] Jaouher Ben Ali, Nader Fnaiech, Lotfi Saidi, Brigitte Chebel-Morello, and Farhat Fnaiech. «Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals». In: *Applied Acoustics* 89 (2015), pp. 16–27 (cit. on p. 19).

[34] J Lee, H Qiu, G Yu, J Lin, et al. «Rexnord technical services: Bearing data set». In: *Moffett Field, CA: IMS, Univ. Cincinnati. NASA Ames Prognostics Data Repository, NASA Ames* (2007) (cit. on pp. 20, 47).

[35] T Williams, X Ribadeneira, S Billington, and T Kurfess. «Rolling element bearing diagnostics in run-to-failure lifetime testing». In: *Mechanical systems and signal processing* 15.5 (2001), pp. 979–993 (cit. on p. 20).

[36] Jason R Stack, Thomas G Habetler, and Ronald G Harley. «Fault-signature modeling and detection of inner-race bearing faults». In: *IEEE transactions on Industry Applications* 42.1 (2006), pp. 61–68 (cit. on p. 21).

[37] Registered Member. *Typical BPFO failure at 80cpm conveyor pulley bearing.* URL: `https://www.maintenance.org/topic/typical-bpfo-failure-at-60-cpm-conveyor-pulley-bearing` (cit. on p. 22).

[38] Shiza Mushtaq, MM Islam, and Muhammad Sohaib. «Deep Learning Aided Data-Driven Fault Diagnosis of Rotatory Machine: A Comprehensive Review». In: *Energies* 14.16 (2021), p. 5150 (cit. on p. 24).

[39] *Regressionlayer.* URL: `https://ch.mathworks.com/help/deeplearning/ug/convert-classification-network-into-regression-network.html` (cit. on p. 32).

[40] Leixian Shen, Qingyun Zhang, Guoxu Cao, and He Xu. «Fall detection system based on deep learning and image processing in cloud environment». In: *Conference on Complex, Intelligent, and Software Intensive Systems.* Springer. 2018, pp. 590–598 (cit. on p. 32).

[41] Zijun Zhang. «Improved adam optimizer for deep neural networks». In: *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS).* IEEE. 2018, pp. 1–2 (cit. on p. 34).

[42] Léon Bottou et al. «Stochastic gradient learning in neural networks». In: *Proceedings of Neuro-Nımes* 91.8 (1991), p. 12 (cit. on p. 34).

[43] Agnes Lydia and Sagayaraj Francis. «Adagrad—an optimizer for stochastic gradient descent». In: *Int. J. Inf. Comput. Sci* 6.5 (2019), pp. 566–568 (cit. on p. 34).

[44] Sebastian Ruder. «An overview of gradient descent optimization algorithms». In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 34).

[45] Guangming Dong and Jin Chen. «Noise resistant time frequency analysis and application in fault diagnosis of rolling element bearings». In: *Mechanical Systems and Signal Processing* 33 (2012), pp. 212–236 (cit. on p. 37).

[46] Edwin Sutrisno, Hyunseok Oh, Arvind Sai Sarathi Vasan, and Michael Pecht. «Estimation of remaining useful life of ball bearings using data driven methodologies». In: *2012 ieee conference on prognostics and health management*. IEEE. 2012, pp. 1–7 (cit. on pp. 45, 50).

[47] Joan Pellegrino, Mauricio Justiniano, Anand Raghunathan, Brian A Weiss, et al. «Measurement science roadmap for prognostics and health management for smart manufacturing systems». In: (2016) (cit. on p. 47).

[48] Antonio Ginart, Irtaza Barlas, Jonathan Goldin, and Jordan Lewis Dorrity. «Automated feature selection for embeddable prognostic and health monitoring (PHM) architectures». In: *2006 IEEE Autotestcon*. IEEE. 2006, pp. 195–201 (cit. on p. 47).

[49] Yu Wang, Yizhen Peng, Yanyang Zi, Xiaohang Jin, and Kwok-Leung Tsui. «A two-stage data-driven-based prognostic approach for bearing degradation problem». In: *IEEE Transactions on industrial informatics* 12.3 (2016), pp. 924–932 (cit. on p. 47).

[50] James Kuria Kimotho, Christopher Sondermann-Wölke, Tobias Meyer, and Walter Sextro. «Machinery Prognostic Method Based on Multi-Class Support Vector Machines and Hybrid Differential Evolution–Particle Swarm Optimization». In: *Chemical Engineering Transactions* 33 (2013) (cit. on pp. 47, 48).

[51] *ISO 18436-2:2014*. Sept. 2019. URL: `https://www.iso.org/standard/50447.html` (cit. on p. 48).

[52] Naipeng Li, Yaguo Lei, Jing Lin, and Steven X Ding. «An improved exponential model for predicting remaining useful life of rolling element bearings». In: *IEEE Transactions on Industrial Electronics* 62.12 (2015), pp. 7762–7773 (cit. on p. 48).

[53] N Tandon. «A comparison of some vibration parameters for the condition monitoring of rolling element bearings». In: *Measurement* 12.3 (1994), pp. 285–289 (cit. on p. 54).

[54] Feng Yang, Mohamed Salahuddin Habibullah, Tianyou Zhang, Zhao Xu, Pin Lim, and Sivakumar Nadarajan. «Health index-based prognostics for remaining useful life predictions in electrical machines». In: *IEEE Transactions on Industrial Electronics* 63.4 (2016), pp. 2633–2644 (cit. on p. 59).

[55] Patricia Scanlon, Darren F Kavanagh, and Francis M Boland. «Residual life prediction of rotating machines using acoustic noise signals». In: *IEEE Transactions on Instrumentation and Measurement* 62.1 (2012), pp. 95–108 (cit. on p. 61).

[56] Hack-Eun Kim, Andy CC Tan, Joseph Mathew, and Byeong-Keun Choi. «Bearing fault prognosis based on health state probability estimation». In: *Expert Systems with Applications* 39.5 (2012), pp. 5200–5213 (cit. on p. 61).

[57] Qingjie Meng, Daniel Rueckert, and Bernhard Kainz. «Learning cross-domain generalizable features by representation disentanglement». In: *arXiv preprint arXiv:2003.00321* (2020) (cit. on p. 66).