

Fault Diagnosis of Motor Bearings Using Machine Learning



Prepared by:

Revashan Soobiah-SBHREV001

Talon Sewnath-SWNTAL001

Prepared for:

EEE4114F

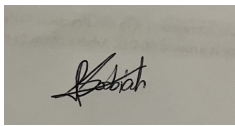
Department of Electrical Engineering

University of Cape Town

May 16, 2024

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.



May 16, 2024

Revashan Soobiah

Date



May 16, 2024

Talon Sewnath

Date

Contents

List of Figures	v
1 Introduction	1
1.1 Background	1
1.2 Objectives	1
1.3 System Requirements	1
1.4 Report Outline	1
2 Literature Review	2
2.1 Introduction	2
2.2 Common Faults	2
2.3 Existing Digital Signal Processing Methods	2
2.3.1 Wavelet transform	2
2.3.2 Fast-Fourier Transform	3
2.4 Existing Machine Learning Methods	3
2.4.1 Deep Neural Networks (DNN)	3
2.4.2 Random Forest	3
2.5 Conclusion	3
3 Methodology	4
3.1 Down Sampling	4
3.2 Filter	4
3.3 Fast Fourier Transform	4
3.4 Machine Learning Algorithm	5
3.4.1 Optimisation	5
3.4.2 Training the Model	5
4 Results	6
4.1 Data Processing	6
4.1.1 Data Capturing	6
4.1.2 Pre-Processing	6
4.2 Learning Algorithm	7
4.2.1 Deep Neural Networks	7
4.2.2 Random Forests	8
5 Conclusion	9
Bibliography	10
A Digital Signal Processing	12
A.1 DSP Code	12

A.2	Unfiltered Vibration Plot	14
B	Machine Learning	15
B.1	ML Code (Deep Neural Network)	15
B.2	Plots of Maximum Voltage vs Frequency indicating states	16
B.3	Plots of Maximum Voltage vs Mean Frequency indicating states	17
B.4	ML Code (Random Forest)	17

List of Figures

3.1	Process used to build the functioning model[1]	4
4.1	Filtered Signal FFT	6
4.2	Final System Design	7
4.3	Random Forest Accuracy	8
A.1	Unfiltered Signal FFT	14
B.1	Maximum voltage during the different states	16
B.2	Maximum voltage during the different states	17

Chapter 1

Introduction

1.1 Background

This report focuses on the health of bearings used in the motor industry. Motors are one of the more widespread pieces of machinery used in industry, which is further indicated since they make up between 60-70% of industrial power consumption. According to [2], 40% of faults are due to bearing failures. Bearings are mechanical support systems that allow for rotational or linear movements in machinery. They prevent contact between two moving parts in order to reduce friction, in turn, reducing the energy consumption of the entire system.. In such a system, bearings are exposed to extreme operating conditions, such as extreme loads, dirt and rough weather conditions causing degradation. Degradation in motors allow for faults to occur, which decreases productivity in industry. Furthermore, bearings are one of the most vulnerable components[3] in a machine system. The continuous operating conditions result in the bearing being worn down to the point of failure if proper maintenance is not adhered to.

1.2 Objectives

The objective of this report is to identify and implement various methods used in Digital Signal Processing and Machine Learning to determine and aid in the health assessment of bearings in the motor industry. To achieve this objective, datasets related to bearing faults were collected from the [4]. The dataset was downsampled, filtered through a bandpass filter and then subjected to a preprocessing algorithm, such as the FFT. Subsequently, it was passed through a developed machine learning algorithm, where it was trained and validated.

1.3 System Requirements

The system requirements involve making sure that the computational requirements of the signal processing system are low, and that the machine learning algorithm has a high accuracy, while being able to determine the difference between faults.

1.4 Report Outline

The report is structured as follows:

Chapter 1 provides insight into the report. Chapter 2 conducts a comprehensive literature review into the existing Digital Signal Processing and Machine Learning algorithms applied in the report. Chapter 3 details the methodology used in order to achieve the objective of this report. Chapter 4 analyses the results of the methodology used to determine the accuracy of the predictive algorithm. Finally, Chapter 5 concludes the report with the findings and recommendations.

Chapter 2

Literature Review

2.1 Introduction

Electric motors make up 60-70% of industrial energy consumption which indicates their ubiquity [5]. Due to the nature of machine operations, mechanical degradation is a common problem which results in a decrease in productivity and profits due to downtime. The advent of learning algorithms has resulted in a nearly impeccable diagnosis of motor faults [6]. Due to the versatility of machine algorithms, several types of faults have been able to be detected with very high accuracy. The integration of machine learning into motor condition monitoring systems also enables early fault identification which provides an opportunity to mitigate the risk of failure and downtime.

2.2 Common Faults

Despite advancements in the modern era, the operation of machinery still allows for faults to prevail. The effect of machine faults on factory productivity is an immense concern for manufacturing industries [7]. Machine faults occur due to machine degradation, which is a result of wear and tear. This allows for the reliability, performance and overall efficiency of the machine to decrease. Furthermore, the ripple effect of the machine's degradation results in high repair costs, safety hazards for workers and a production halt.

Consequently, a systematic monitoring system must be implemented. Numerous implementations of the systematic monitoring system have been used in the production industry. Majority of implementations involve the vibration analysis technique. This technique is reliable and accurate in detecting a defect in the bearing elements of the machine system [8]. Using this analysis, various digital signal processing and machine learning algorithms have been designed.

2.3 Existing Digital Signal Processing Methods

The following sections will delve into the methods used to analyse the health conditions of bearings motor systems.

2.3.1 Wavelet transform

The wavelet transform is a result of analyzing a particular input signal into different frequencies that have different resolutions. This particular transform retains the information of both the time and frequency domains. However, from the numerous advantages, the wavelet transformation has its drawbacks. The computational complexity of a wavelet transform can be intensive for its multi-resolution. Furthermore, different frequency bands are prone to overlap in wavelet transforms, complicating the interpretation of its results.

2.3.2 Fast-Fourier Transform

The fast-Fourier transform is an algorithm that analyses the discrete Fourier transform of a signal significantly faster than a manual approach. According to [9], The FFT is ‘the major signal processing method of vibrations.’ There are many variants of the FFT algorithm, and all exploit the basic redundancy in the computation of the Discrete Fourier Transform. The algorithm uses the Fourier transform that converts the time-domain vibrational signal to its equivalent frequency domain representation[10]. As a result of using the fast Fourier transform, various harmonics of the ball bearings are able to be analysed. In this report, the fast Fourier transform has been selected due to its surplus of information and resources.

2.4 Existing Machine Learning Methods

Supervised models require large datasets which in turn increases the amount of time required for training the algorithm [2]. According to Sobhi et al.(2023), unsupervised models are prone to noise and faulty data. Furthermore, semi-supervised learning algorithms only require data that belongs to one class. As a result, semi-supervised and suitable supervised learning algorithms will be focused on. The following sections will identify the existing algorithms used in condition monitoring systems.

2.4.1 Deep Neural Networks (DNN)

Deep Neural Networks are a form of deep learning which have gained attention in the condition monitoring field. The hierarchical layer-by-layer process allows the algorithm to learn complex representations that are embedded in higher-level concepts related to unique motor fault conditions [1]. Convolutional Neural Networks are a subset of DNN. In industrial use, two parallel CNNs are used, one for fault pattern determination and the other for fault size determination [11]. According to the results from [1], CNN is able to learn multiple features and effectively classify various faults with a 10% increase in accuracy with less dependence on human knowledge. CNN algorithms also allow for early fault detection [12], which enables proactive strategies to avoid failure.

2.4.2 Random Forest

Random Forest is a classification algorithm that is made up of multiple tree decision estimators on random subsets of training data [6]. RF’s ability to handle high-dimensional data, interpret non-linear relationships and avoid overfitting makes it a suitable structure for condition monitoring [13]. In an experiment done by [14], it was found that RF uses less computational resources and produces a more precise model than SVM (Supervised Vector Machine). Random Forests are particularly useful in diagnosing faults which include bearing defects, rotating imbalance and stator winding faults. An important quality of RF is that it is resilient to noise which is crucial in data processing since sensors contain noise [15]

2.5 Conclusion

In conclusion, the health of bearings in the motor industry is a crucial aspect to monitor. Using digital signal processing, the fft will be taken. This ensures an easy analysis into the different faults in a bearing. The use of Machine Learning has proved to ease the algorithms for condition monitoring. The two algorithms will then be explored to determine its individual efficiency for this application.

Chapter 3

Methodology

The following chapter details the methodology used to interpret and analyse the health of bearings in motors. The obtained dataset is processed through a series of analysis tools resulting in an advanced computational algorithm that has been developed, trained and validated to predict a fault in the motor. The processes used are down sampling, fast-fourier transform and machine learning tools as seen in Figure 3.1.

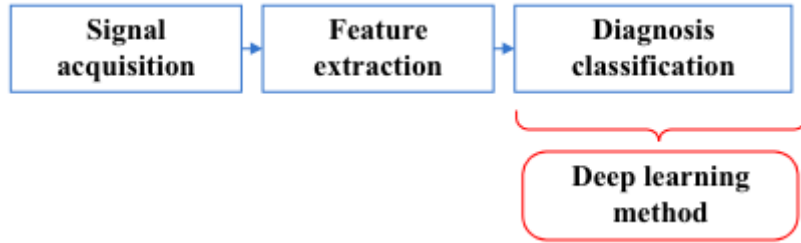


Figure 3.1: Process used to build the functioning model[1]

3.1 Down Sampling

Downsampling is a method used to reduce the size of a dataset by removing certain instances. However, this may result in aliasing. In this project, downsampling has been utilised as a way to reduce the the large datasets acquired from multiple sources.

3.2 Filter

Before the filtering process, zero-padding was used to improve the output of the filter, as it contributes to a better performance. Thereafter, filtering was implemented to reduce noise, captured by the sensors, within the dataset. A band pass filter was implemented with a 20kHz cut-off frequency. This specific frequency was chosen due to the motor's healthy harmonics being at a frequency of 10kHz [16]. The range from 10kHz to 20kHz is used as a precaution for the bearing health analysis, with frequencies beyond this range being filtered out due to noise.

3.3 Fast Fourier Transform

The Fast Fourier Transform algorithm is used to interpret and analyse the harmonics of the bearings in motors. The downsampled datasets were Fourier transformed and then plotted on an amplitude vs. time/frequency graph. From the analysis of these graphs, the datasets proved to be effectively processed. Thus, they could be implemented to train and validate the machine learning algorithm.

The plots revealed a discernible pattern, in which bearings with no faults exhibited a specific amplitude. Whereas, bearings with faults displayed magnitudes that differed to that of the amplitude with no faults. This difference in magnitude allowed for the identification of faulty bearings.

3.4 Machine Learning Algorithm

The literature review in Chapter [chapter 2](#) states that the Dense Neural Network process allows algorithms to learn complex representations, whereas the random forest is able to avoid over fitting and handle high-dimensional data effectively. Therefore, in order to achieve the objective of determining faults accurately through the use of a predictive machine learning algorithm, both methods were compared. There were various reasons for this choice, including the decision on which algorithm performs more efficiently.

The DNN algorithm uses different layers of neural network that is connected to the previous layer. Using multiple layers will allow for the model to learn the input data and predict with a higher accuracy. As seen in [Appendix B](#), multiple layers of neural networks were used. The first few layers utilised, allowed for the algorithm to learn low-level features. Following into deeper layers, the model improves its accuracy, learns non-linear relationships and has a greater capacity to fit the data given. The random forest algorithm combines the output of multiple decision trees to determine a single result. This algorithm is used for regression and classifications tasks.

3.4.1 Optimisation

Despite its many advantages, the models are prone to over fitting. Thus, the following strategies were implemented to mitigate this:

Dense Neural Network

- **Variation of the Hyper parameters:** This method controls the process of the model and determines the trade-off between under fitting and over fitting.
- **Optimisation of the number of Epochs:** The number of epochs determines the model's generalization ability, potentially resulting in either under fitting or over fitting. In optimising this, the time taken for the model to train can be determined.
- **Increasing the Neural Network Layers:** This method allows for the algorithm to be tailored to specific tasks, however this will increase the complexity of the models architecture.

Random Forest

To optimise the Random Forest algorithm, the random states were tuned to the specific estimators. Estimators are the many trees that output a single result. The random states control the randomness of the sample. initially, there were 100 estimators with 10 random states. In order to optimise the model, the random states were tuned from 10 to 42. This resulted in the model producing an output with more stability.

3.4.2 Training the Model

A crucial step in an advanced computational algorithm is training the model. In the development of this model, a train-test split was utilized. The randomized data was then used to train and test the model. Using the new randomized data, the model was then validated. Analyzing the output, a cost function was implemented to minimize the error.

Chapter 4

Results

4.1 Data Processing

4.1.1 Data Capturing

The data found in the CWRU database[4] was captured using a tachometer on the induction motor's rotor and voltage readings on the rotor terminals. Sampling was done on the front end of the induction motor running at 1720 rpm at 48kHz. The datasets used in this study consisted of a large number of data points with eight features each. There were three unique states in the dataset, interior imbalance, exterior imbalance, ball fault and normal operation. Consequently, the computational resources exceeded the amount available therefore the data was down-sampled using the function found in [Appendix A](#)(line 21). The following table presents the different downsampling test results.

Downsampling		
Scaling (%)	Memory (MB)	Loading Time (s)
10	718.4	No Allocation
1	213.6	36
0.1	51.7	2

Table 4.1: Table showing the results of downsampling

4.1.2 Pre-Processing

The resolution of the frequency domain improves as the number of zero-padded indexes increases. As a general rule of thumb, the number of zero-padded indexes should equal the number of data points. The vibration signals are periodic due to the nature of the induction machine's rotation therefore the vibration properties vary cyclically. As a result, the FFT captures the unique frequencies.

The unfiltered signal contains high-frequency components as seen in [Figure A.1](#). The natural frequencies for bearings are between 2kHz and 10kHz therefore the low pass filter attenuated signals above 20kHz [16]. The additional range was due to the uncertainty of faulty vibration data.

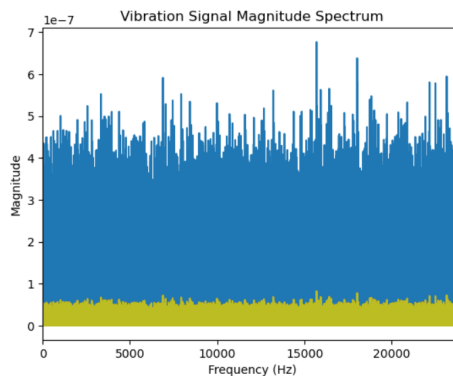


Figure 4.1: Filtered Signal FFT

From Figure 4.1, it can be seen that in the range up to 10kHz, there are no significant spectral components which is in line with the principle that energy is even distributed across all frequencies in normal operation. The spectral components above 10kHz are due to high-frequency Gaussian noise. The FFT algorithm in Appendix A (line 27) was deemed successful and applied to the rest of the data sets.

4.2 Learning Algorithm

Both of the learning algorithms were built to determine which model is better suited for the motor fault detection application.

4.2.1 Deep Neural Networks

The training and test data were split randomly and the test data size was varied. The following table details the results of model-building using different sizes of test data.

Test Data Variation		
Percentage Test (%)	Test Accuracy (%)	Training Accuracy (%)
10	92	95
25	91	93
50	43	67
80	28	37

Table 4.2: Table showing the results of increasing the amount of test data

At the 50% split, the training algorithm was not able to model sufficiently resulting in a lower testing accuracy. The peak performance to optimize computational resources is to use a 75-25 training-testing split. Further increase in the size of the validation set causes an increase in the testing accuracy at the expense of exponentially higher execution time.

The second test procedure was done by varying the number of epochs at the optimal test split. The following figure shows the accuracy at different epochs. The accuracy stabilizes at 50 epochs however

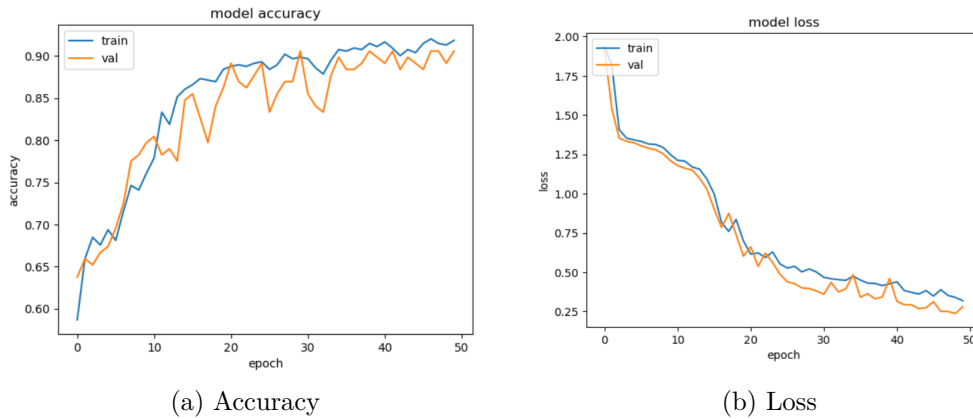


Figure 4.2: Final System Design

the loss function is decreasing beyond that point. During simulation, a higher number of epochs resulted in insignificant differences in accuracy however loss stabilized at 70 epochs. The execution time increased linearly with the increase in epochs therefore 50 epochs resulted in the optimal execution time and model loss.

The last test procedure was done by increasing the amount of layers from two layers to five layers. Since the dataset consists of four states, the model was required to learn complex patterns. Two layers with high density resulted in a model with sub-40% accuracy. As the number of layers increased the accuracy increased logarithmically. This was observed by increasing the layers to six resulting in no significant change of accuracy.

In [section B.2](#), it can be seen that the healthy motor has maximum voltages greater than that of the different faults. It can also be seen that in normal operation the machine is able to generate higher voltages which agrees with the theory that symmetrical rotation ensures a constant air gap flux thereby maximizing the voltage. In [section B.3](#), it can be seen that the Maximum voltage does not vary significantly with a change in mean frequency for the normal model of operation. On the other hand, the data with faults had erratic maximum values when the mean rotation frequency changed. The previous two plots provide a visual identification system for separating healthy data and faulty data.

4.2.2 Random Forests

The model was set to 100 estimators as a step point and the random states were varied. During the iterations, the number of random states that performed optimally was between 37 and 45. The model was experiencing similar issues to the DNN model for estimators above the range found. For the range below, the accuracy was very low until 30 random states since the random tree was not able to model the different motor states well enough to categorize the data into four states.

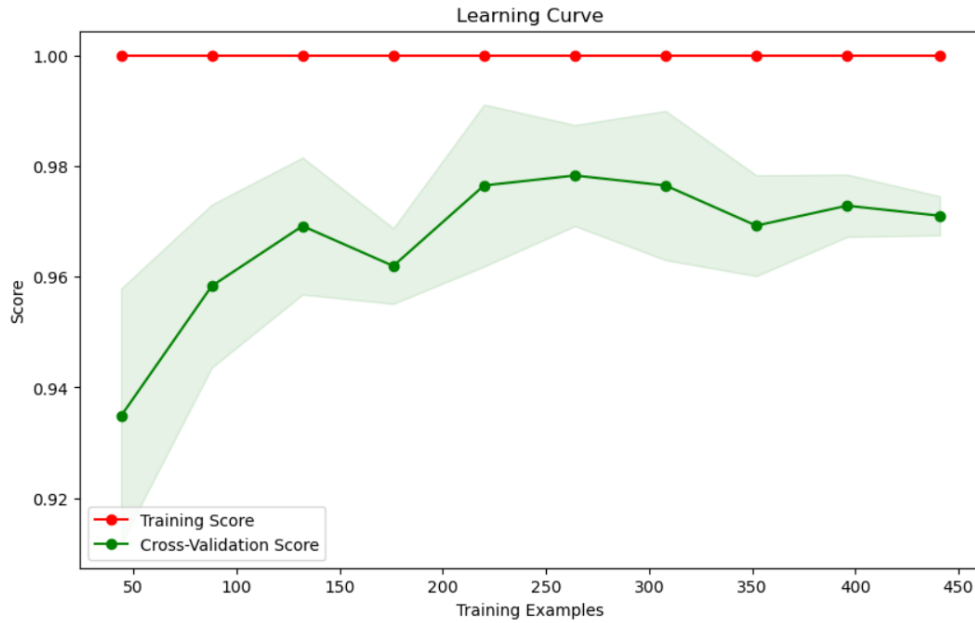


Figure 4.3: Random Forest Accuracy

The train-test split was varied to extreme proportions, namely 20-80 and 95-5. The model was robust at low training data and resulted in an accuracy of 84%. The accuracy increase was insignificant and high training data volumes. This can be observed from [Figure 4.3](#); the variance in accuracy of the test data was 0.04. The training data accuracy is at its optimal at 50 data points which indicates the speed at which random forest can model the data.

Chapter 5

Conclusion

The purpose of this project was to determine the health of bearings in a motor. This was done using tools in digital signal processing and creating a prediction algorithm to determine faults. This report began with an introduction to the theory behind various faults in the motor industry with bearings causing 40% of all faults. The literature review was followed in Chapter 2, giving a comprehensive review of the different methods used. The process of diagnosing faults consisted of sampling data, downsampling the data to a practical volume, performing normalization and preprocessing and finally training a model.

The DNN model resulted in an optimal accuracy of 91% using 50 epochs and 5 layers. Random outperformed DNN's accuracy with an accuracy of 94%. Due to Random Forest's robustness and decreased computational power, it can be concluded that it is the preferred choice for categorizing motor bearing faults. In summary, the project achieved the goals that were set out.

Although the Random Forest algorithm performed more efficiently compared to the Deep Neural Network, the model can be improved with the use of the k-fold Cross-Validation and the use of independent external data to determine the validity of the model.

Bibliography

- [1] L. Jing, M. Zhao, P. Li, and X. Xu, “A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox,” *Measurement*, vol. 111, pp. 1–10, 2017.
- [2] S. Sobhi, M. Reshadi, N. Zarft, A. Terheide, and S. Dick, “Condition monitoring and fault detection in small induction motors using machine learning algorithms,” *Information*, vol. 14, no. 6, p. 329, 2023.
- [3] E. A. Burda, G. V. Zusman, I. S. Kudryavtseva, A. P. Naumenko *et al.*, “An overview of vibration analysis techniques for the fault diagnostics of rolling bearings in machinery,” *Shock and Vibration*, vol. 2022, 2022.
- [4] C. W. R. University, “Bearing data center,” <https://engineering.case.edu/bearingdatacenter>, 2022, accessed: 2023-05-16.
- [5] J.-F. Stumper, A. Dötlinger, and R. Kennel, “Loss minimization of induction machines in dynamic operation,” *IEEE transactions on energy conversion*, vol. 28, no. 3, pp. 726–735, 2013.
- [6] N. Sikder, K. Bhakta, A. Al Nahid, and M. M. M. Islam, “Fault diagnosis of motor bearing using ensemble learning algorithm with fft-based preprocessing,” in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 2019, pp. 564–569.
- [7] J. Lee and B. M. Kramer, “Analysis of machine degradation using a neural network based pattern discrimination model,” *Journal of manufacturing systems*, vol. 12, no. 5, pp. 379–387, 1993.
- [8] H. Saruhan, S. Saridemir, A. Qicek, and I. Uygur, “Vibration analysis of rolling element bearings defects,” *Journal of applied research and technology*, vol. 12, no. 3, pp. 384–395, 2014.
- [9] D. Strömbergsson, P. Marklund, K. Berglund, and P.-E. Larsson, “Bearing monitoring in the wind turbine drivetrain: A comparative study of the fft and wavelet transforms,” *Wind Energy*, vol. 23, no. 6, pp. 1381–1393, 2020.
- [10] M. Khazaei, H. Ahmadi, M. Omid, and A. Moosavian, “An appropriate approach for condition monitoring of planetary gearbox based on fast fourier transform and least-square support vector machine,” *International Journal of Multidisciplinary Sciences and Engineering*, vol. 3, no. 5, pp. 22–26, 2012.
- [11] X. Guo, L. Chen, and C. Shen, “Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis,” *Measurement*, vol. 93, pp. 490–502, 2016.
- [12] X. Wen, G. Lu, J. Liu, and P. Yan, “Graph modeling of singular values for early fault detection and diagnosis of rolling element bearings,” *Mechanical Systems and Signal Processing*, vol. 145, p. 106956, 2020.

- [13] I. V. Vamsi, N. Abhinav, A. K. Verma, and S. Radhika, “Random forest based real time fault monitoring system for industries,” in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*. IEEE, 2018, pp. 1–6.
- [14] R. Kizito, P. Scruggs, X. Li, R. Kress, M. Devinney, and T. Berg, “The application of random forest to predictive maintenance,” in *IIE Annual Conference. Proceedings*. Institute of Industrial and Systems Engineers (IISE), 2018, pp. 354–359.
- [15] A. Folleco, T. M. Khoshgoftaar, J. Van Hulse, and L. Bullard, “Software quality modeling: The impact of class noise on the random forest classifier,” in *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 3853–3859.
- [16] S. T. A. . C. KG, “Condition monitoring of rolling bearings,” Schaeffler Technologies AG Co. KG, Tech. Rep., 2017, accessed: 2024-05-16. [Online]. Available: https://www.schaeffler.com/remotemedien/media/_shared_media/08_media_library/01_publications/schaeffler_2/technicalpaper_1/download_1/vibration_analysis_en_en.pdf

Appendix A

Digital Signal Processing

A.1 DSP Code

```
1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
3 import seaborn as sns
4 from pylab import rcParams
5 import matplotlib.pyplot as plt
6 import glob
7 from scipy import signal
8
9 #Read in the data
10 data_time = pd.read_csv(r"C:\Users\Talon\Documents\4th year\EEE4114F\Final\archive\
    ↳ feature_time_48k_2048_load_1.csv")
11 data_time_first_9 = data_time.iloc[:, :9] #Extract the first 9 columns
12
13 #Function for reading the data into an array
14 def dataReader(path_names):
15     data_n = pd.DataFrame()
16     for i in path_names:
17         low_data = pd.read_csv(i,header=None)
18         data_n = pd.concat([data_n,low_data],ignore_index=True)
19     return data_n
20
21 #Down Sample to a smaller data set
22 def downSampler(data, sampling_rate):
23     data_downsampled = pd.DataFrame()
24     for i in range(0, len(data), sampling_rate):
25         data_downsampled = pd.concat([data_downsampled, data.iloc[i:i+sampling_rate, :].mean
    ↳ ().to_frame().T], ignore_index=True)
26     return data_downsampled
27
28 def FFT(data): #FFT of the data
29     autocorr = signal.fftconvolve(data,data[::-1],mode='full')
30     return pd.DataFrame(autocorr)
31
32 data_FFT = FFT(data_time_first_9)
33
34 sampling_rate = 48000
35 duration = 10
```

```

35 Normal = glob.glob(r"C:\Users\Talon\Documents\4th year\EEE4114F\Final\archive\
    ↳ Front_End_Normal.csv")
36 def dataReader(path_names):
37     data_n = pd.DataFrame()
38     for i in path_names:
39         low_data = pd.read_csv(i,header=None)
40         data_n = pd.concat([data_n,low_data],ignore_index=True)
41     return data_n
42
43 Normal_Data = dataReader(Normal)
44
45 time = np.arange(0, duration, 1/sampling_rate)
46 Normal_Signal = Normal_Data[:len(time)]
47
48 plt.plot(time, Normal_Signal)
49 plt.title('Vibration Signal vs Time(Bearing)')
50 plt.xlabel('Time (seconds)')
51 plt.ylabel('Amplitude')
52 plt.show()
53
54 low_freq = 10 # Lower cutoff frequency (Hz)
55 high_freq = 20000 # Upper cutoff frequency (Hz)
56
57 # Calculate the normalized frequency range
58 nyquist_freq = 0.5 * sampling_rate
59 low_norm = low_freq / nyquist_freq
60 high_norm = high_freq / nyquist_freq
61
62 # Bandpass filter
63 order = 2 # Order of the filter
64 normalized_filter = signal.butter(order, [low_norm, high_norm], btype='band', analog=False,
    ↳ output='ba')
65
66 # Pad the signal with zeros before filtering
67 padded_signal = np.pad(Normal_Signal, (100, 100), mode='constant')
68
69 # Apply the filter to the padded signal
70 filtered_signal = signal.filtfilt(*normalized_filter, padded_signal)
71
72 fft_signal = fft(filtered_signal)
73
74 # Calculate the frequency bins
75 N = len(filtered_signal)
76 freqs = fftfreq(N, 1 / sampling_rate)
77

```

```

78 # Calculate the magnitude spectrum and scale it properly
79 magnitude_spectrum = np.abs(fft_signal) / N
80
81 # Plot the magnitude spectrum for positive frequencies only
82 plt.figure()
83 plt.plot(freqs[:N//2], magnitude_spectrum[:N//2])
84 plt.title('Vibration Signal Magnitude Spectrum')
85 plt.xlabel('Frequency (Hz)')
86 plt.ylabel('Magnitude')
87 plt.xlim(0, sampling_rate / 2) # Limit the x-axis to the Nyquist frequency
88 plt.show()

```

A.2 Unfiltered Vibration Plot

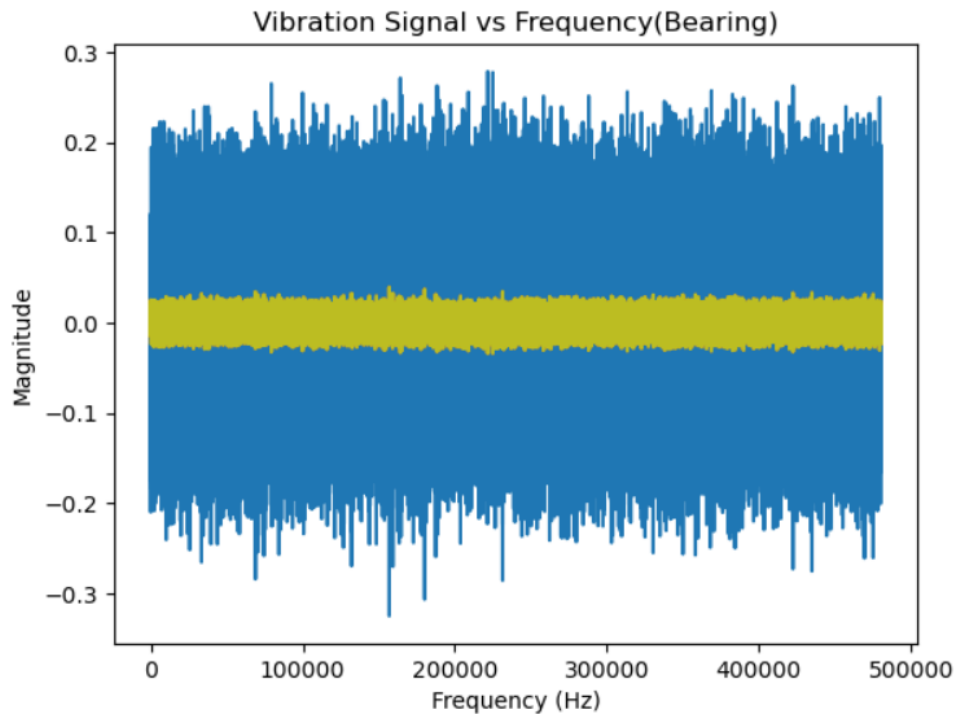


Figure A.1: Unfiltered Signal FFT

Appendix B

Machine Learning

B.1 ML Code (Deep Neural Network)

```
1 # Generate arrays of classification for each data point
2 ones = np.ones(230)
3 twos = np.full(230, 2)
4 threes = np.full(230, 3)
5 zeros = np.zeros(230)
6 y = np.concatenate((ones, twos, threes, zeros))
7
8 # Concatenate the arrays to form y
9 d1 = data_time_first_9.iloc[:691].copy()
10 d2 = data_time_first_9.iloc[2071:2301].copy()
11 data_f = pd.concat([d1,d2], ignore_index=True)
12
13 from sklearn.model_selection import train_test_split
14 X_train, X_test, y_train, y_test = train_test_split(data_f, y, test_size=0.25, shuffle=True)
15
16 from tensorflow.keras.models import Sequential
17 from tensorflow.keras.layers import Dense
18 from tensorflow.keras.layers import Dropout
19 from tensorflow.keras.callbacks import EarlyStopping
20
21 early_stop = EarlyStopping(monitor='loss', patience=2)
22 model = Sequential()
23
24 #Sets hyperparameters, input shape and layer configuration
25 model.add(Dense(32, activation='relu', input_shape=(9,), kernel_initializer='random_uniform')
26     ↪ )
27 model.add(Dense(64, activation='relu', kernel_initializer='random_uniform'))
28 model.add(Dense(128, activation='relu', kernel_initializer='random_uniform'))
29 model.add(Dense(64, activation='relu', kernel_initializer='random_uniform'))
30 model.add(Dense(32, activation='relu', kernel_initializer='random_uniform'))
31 model.add(Dense(7, activation='softmax', kernel_initializer='random_uniform'))
32
33 model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy']
34     ↪ )
35 model.summary()
```

```

35 from sklearn.preprocessing import LabelEncoder
36 y = LabelEncoder().fit_transform(y)
37
38 hist = model.fit(X_train , y_train , epochs=50, validation_split=0.2)
39
40 import matplotlib.pyplot as plt
41 plt.plot(hist.history['accuracy'])
42 plt.plot(hist.history['val_accuracy'])
43 plt.title('model accuracy')
44 plt.ylabel('accuracy')
45 plt.xlabel('epoch')
46 plt.legend(['train', 'val'], loc='upper left')
47 plt.show()
48
49 from sklearn.preprocessing import OrdinalEncoder
50
51 ord_enc = OrdinalEncoder()
52 data_time["fault_code"] = ord_enc.fit_transform(data_time[["fault"]])
53 data_time[["fault", "fault_code"]]
54
55 import matplotlib.pyplot as plt
56 plt.plot(hist.history['loss'])
57 plt.plot(hist.history['val_loss'])
58 plt.title('model loss')
59 plt.ylabel('loss')
60 plt.xlabel('epoch')
61 plt.legend(['train', 'val'], loc='upper left')
62 plt.show()

```

B.2 Plots of Maximum Voltage vs Frequency indicating states

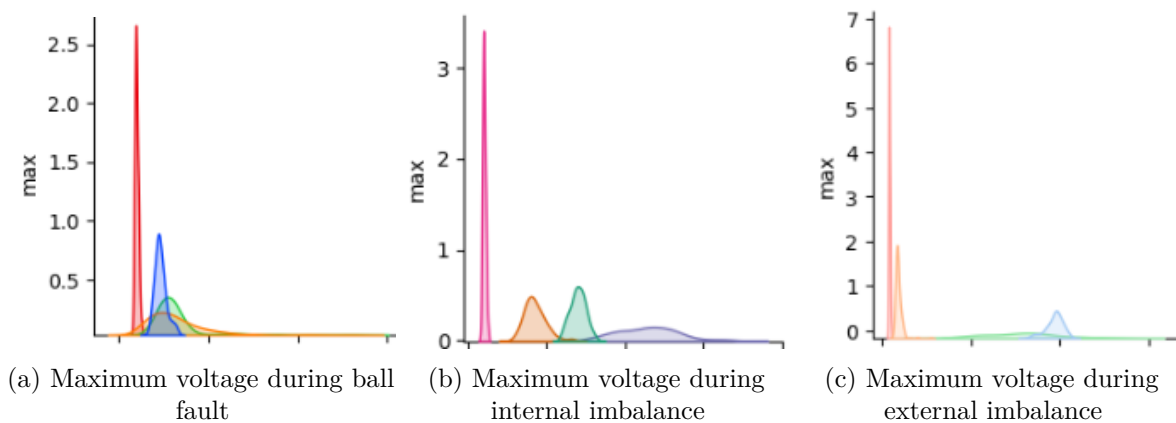


Figure B.1: Maximum voltage during the different states

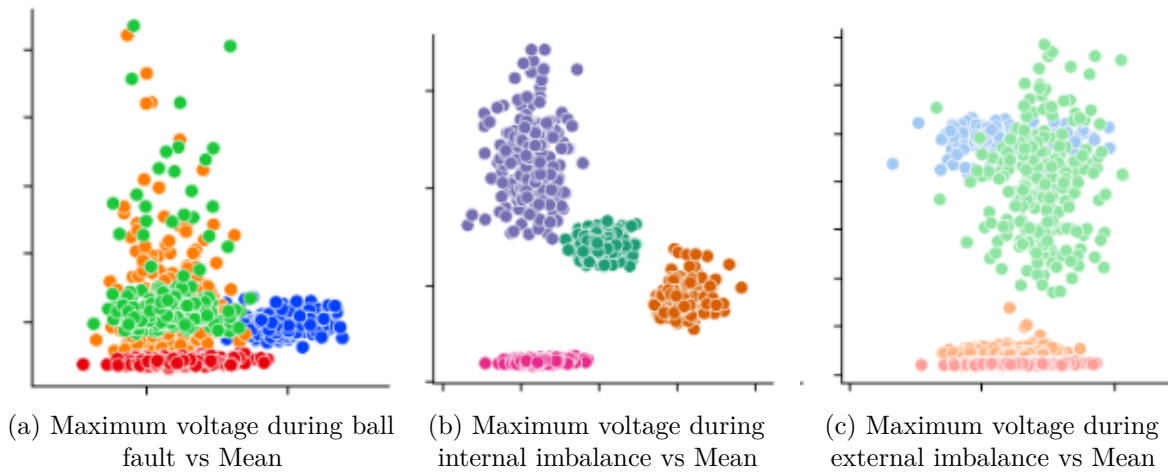


Figure B.2: Maximum voltage during the different states

B.3 Plots of Maximum Voltage vs Mean Frequency indicating states

The normal motor operations are represented in red for all 3 plots. There are 3 different sub-faults per fault. Eg 3 IR faults and one normal operation in the second plot.

B.4 ML Code (Random Forest)

```

1 # Import necessary libraries
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 X_train, X_test, y_train, y_test = train_test_split(data_f, y, test_size=0.4, shuffle=True)
7
8 rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
9
10 rf_model.fit(X_train, y_train)
11
12 y_pred = rf_model.predict(X_test)
13
14 accuracy = accuracy_score(y_test, y_pred)
15 print(f"Accuracy: {accuracy:.2f}")
16
17 from sklearn.model_selection import learning_curve
18
19 # Calculate the learning curve
20 train_sizes, train_scores, test_scores = learning_curve(
21     estimator=rf_model,
22     X=X_train,
23     y=y_train,
24     cv=5,
25     scoring='accuracy',

```

```

26     n_jobs=-1,
27     train_sizes=np.linspace(0.1, 1.0, 10)
28 )
29
30 # Calculate the mean and standard deviation for training and test scores
31 train_mean = np.mean(train_scores, axis=1)
32 train_std = np.std(train_scores, axis=1)
33 test_mean = np.mean(test_scores, axis=1)
34 test_std = np.std(test_scores, axis=1)
35
36 # Plot the learning curve
37 plt.figure(figsize=(10, 6))
38 plt.title("Learning Curve")
39 plt.xlabel("Training Examples")
40 plt.ylabel("Score")
41 plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.1,
42                  color="r")
43 plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.1, color="g")
44 plt.plot(train_sizes, train_mean, 'o-', color="r", label="Training Score")
45 plt.plot(train_sizes, test_mean, 'o-', color="g", label="Cross-Validation Score")
46 plt.legend(loc="best")
47 plt.show()

```