MACHINE ANOMALY DETECTION USING SOUND SPECTROGRAM IMAGES AND NEURAL NETWORKS

by

Hanjun Kim

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Mechanical Engineering

School of Mechanical Engineering

West Lafayette, Indiana

August 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Martin B.G. Jun, Chair
School of Mechanical Engineering
Dr. Patricia Davies
School of Mechanical Engineering
Dr. Peter H. Meckl
School of Mechanical Engineering

Approved by:

Dr. Jay P. Gore
Head of the Graduate Program

To my Parents and Country

ACKNOWLEDGMENTS

First of all, I would like to thank my advisory professor Dr. Martin Jun. Without his sincere instructions and guidance during my master's program, I would have never completed this work.

Special gratitude to my committee members, Dr. Patricia Davies and Dr. Peter Meckl, for providing constructive comments on the thesis. Their professional assistance made this achievement possible.

I would like to thank my family and friends in my motherland, and my friends in LAMM for caring about me at Purdue. My peaceful days without any single accident owe to their help.

I will never be able to forget anything I have faced here at Purdue. Though I'm returning back to my nation, I will remember every single second of my 2 years at Purdue.

Lastly, I would like to take this opportunity to thank my country, the Republic of Korea and Army, for making all of this work possible by their trust and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Author: Kim, Hanjun. MSME
Institution: Purdue University
Degree Received: August 2019
Title: Machine Anomaly Detection using Sound Spectrogram Images and Neural Networks
Committee Chair: Martin B.G. Jun

Sound and vibration analysis is a prominent tool used for scientific investigations in various fields such as structural model identification or dynamic behavior studies. In manufacturing fields, the vibration signals collected through commercial sensors are utilized to monitor machine health, for sustainable and cost-effective manufacturing.

Recently, the development of commercial sensors and computing environments have encouraged researchers to combine gathered data and Machine Learning (ML) techniques, which have been proven to be efficient for categorical classification problems. These discriminative algorithms have been successfully implemented in monitoring problems in factories, by simulating faulty situations. However, it is difficult to identify all the sources of anomalies in a real environment.

In this paper, a Neural Network (NN) application on a KUKA KR6 robot arm is introduced, as a solution for the limitations described above. Specifically, the autoencoder architecture was implemented for anomaly detection, which does not require the predefinition of faulty signals in the training process. In addition, stethoscopes were utilized as alternative sensing tools as they are easy to handle, and they provide a cost-effective monitoring solution. To simulate the normal and abnormal conditions, different load levels were assigned at the end of the robot arm according to the load capacity. Sound signals were recorded from joints of the robot arm, then meaningful features were extracted from spectrograms of the sound signals. The features were utilized to train and test autoencoders. During the autoencoder process, reconstruction errors (REs) between the

autoencoder’s input and output were computed. Since autoencoders were trained only with features corresponding to normal conditions, RE values corresponding to abnormal features tend to be higher than those of normal features. In each autoencoder, distributions of the RE values were compared to set a threshold, which distinguishes abnormal states from the normal states. As a result, it is suggested that the threshold of RE values can be utilized to determine the condition of the robot arm.

CHAPTER 1. INTRODUCTION

1.1 Motivation and objectives

Unexpected halt of factory lines may cause devastating costs, or even worse, dangerous incidents. To avoid such cases, a wide range of studies has focused on timely and precise diagnostics of machine components. To capture the behaviors of the target machine, various sensor data are collected, such as torques [1], [2], vibrations [3]–[5], sound emissions [6], [7], and currents [8], [9]. The gathered signals are investigated to construct physical models, or to extract meaningful features that represent the status of the machines. The former approach is called a model-based method, and the latter is called a data-driven method. The model-based method relates sensor signals to the physical model parameters [3], [9]–[11]. After optimizing the parameters, the models then can be utilized to predict sensor signals in specific conditions. On the other hand, the data-driven method emphasizes feature extraction [12]–[14]. The data-driven method does not aim to estimate the exact sensor signals, rather, it tends to pursue grouping and classifying the features from a variety of conditions.

As the data storage capability of computers increases, studies on data handling techniques have arisen as an independent branch of engineering. The achievements in contemporary computer technologies accelerated the new era of smart manufacturing, which is a branch of industry 4.0. In particular, the enhancement of computing infrastructures such as Artificial Intelligence (AI) permits researchers and scholars to utilize a large amount of manufacturing data for multiple purposes.

The suitability of sensors and methods depends on applications. Firstly, there are differences among sensors in terms of bandwidth and cost. Figure 1.1 illustrates the variations of sensors: microphone, accelerometer and acoustic emission (AE) sensors. Since the aim of this work is

collecting gear signals from robot joints in moderate operation speed, the bandwidth of AE sensors is too high for this application. The response ranges of both the microphone and the accelerometer are both acceptable, but there are some discrepancies. For example, the conditioning cost for the accelerometer is high, since data transfer equipment such as data acquisition (DAQ) is necessary. On the other hand, microphones usually have their direct connection to the PC, hence conditioning equipment is optional.

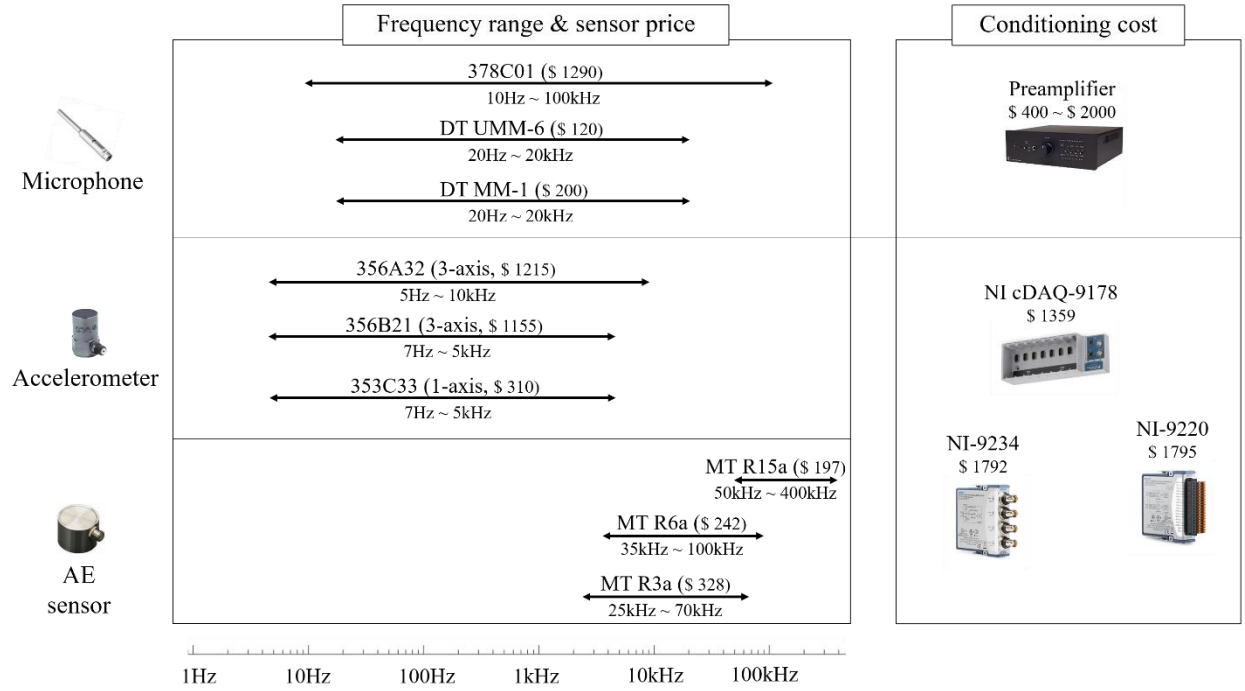


Figure 1.1 Various sensors are compared in multiple aspects, to determine the suitable sensor for our application (based on quotations in May 2019).

However, the sound sensor receives sounds from multiple directions, hence may be susceptible to external noise. To isolate the sensing target from environmental noise, physical sealing can be considered as a solution. Inspired from the health diagnosis on humans, the stethoscope is selected as a sensing tool to diagnose robot health in this work. As portrayed in Figure 1.2, the stethoscope can focus on the target surface and reject noise by rubber sealing. Since the rotational speed of a gear in a robot arm is restricted, the frequency response range of a stethoscope (500Hz) is expected

to be sufficient. Furthermore, the sound signals collected from the stethoscope can be easily delivered to a PC by connecting a USB microphone, which yields cost-effectiveness compared to other sensing methods described in Figure 1.1.

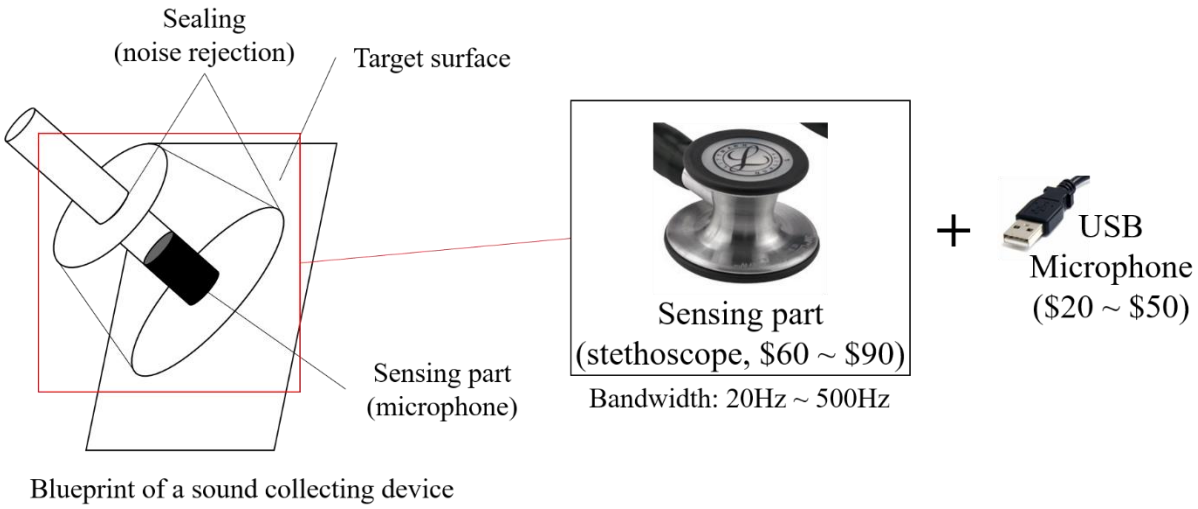


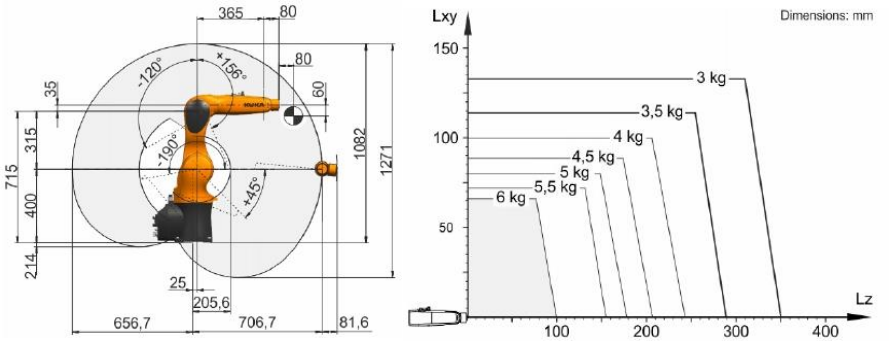
Figure 1.2 The stethoscope is used to shield the low-cost USB microphone from other noise sources and amplify the sound of interest

Next, a proper strategy for signal analysis should be selected between the model-based and data-driven methods. In the model-based approach, it is necessary to clarify the complex relations between the physical and dynamic parameter values and the signal. Therefore, it demands comprehensive knowledge of the attributes of target systems, and takes a long time to verify [5], [11], [15]. In contrast, the data-driven method requires data to be divided into a few groups. Especially in fault detection problems, associating signals to conditions (normal, fault, etc.) is preferred over relating signals to exact parameter values due to its simplicity. To provide a simpler solution than developing physical models, the scope of this work is narrowed down to the data-driven method.

Data-driven machine health monitoring has arisen from the need for classification [12], [13]. In traditional classification problems, the signals from normal and abnormal status are projected into

a feature domain. To collect the abnormal signals, it is required to wait until the machine produces anomalous signals, or to impose some intended defects on the machine. For this reason, the gathering of faulty signals is sometimes limited. Further, running experiments in abnormal conditions may result in severe breakdown that cannot be restored. Due to this, it is more desirable to have reduced requirements for fault simulation data.

In this work, the KUKA KR6 industrial robot arm is chosen as the monitoring target, which is widely used in factories. The robot arm has 6 independent joints and has its own load capacity as depicted in Figure 1.3. Our purpose is to discriminate bad conditions of the robot arm by data-driven method using sound sensors. The autoencoder framework was applied, which is a type of NN architecture. To simulate anomalies, the load conditions were assigned at the end-effector, in compliance with the capacity diagram in Figure 1.3-(b). However, the signals from anomalous status were not utilized in training the autoencoder. They were used only in testing, to verify the NN algorithm can achieve an acceptable anomaly detection performance, without fault data in the training step.



(a) (b)

Figure 1.3 Basic information of KUKA KR6 R700 are illustrated:
(a) geometry of the robot arm, and (b) load capacity diagram.

Accordingly, the objectives of this thesis are: 1) to provide cost-effective data analytics solution using sound sensors, 2) to develop data-driven approach without deriving exact physical model, 3)

to establish anomaly detection algorithms that do not require fault simulation data. To achieve the goals, we employed stethoscopes as sound sensors to focus on sound emissions from joints of the KUKA KR6 industrial robot arm, and to reject external noise. We applied the autoencoder framework to establish the data-driven anomaly detection algorithm, without including fault data sets in training.

1.2 Overview of thesis

The rest of this thesis is outlined as follows. Chapter 2 reviews several research stories relevant to fault detection using data-driven approaches. Contents in Chapter 2 are organized with several ways of feature extraction and their usages for monitoring problems. In addition, the other NN applications for manufacturing are introduced.

Chapter 3 explains some theoretical background required for performing the experiments and anomaly detection algorithm. Initially, a spectrogram-based feature extraction and audio system identification procedure using linear sine chirp are described. These contents are necessary for extracting proper features from raw sound signals. Afterwards, the basic concepts of NN are introduced then expanded to the autoencoder and the Convolutional Neural Network (CNN). The main algorithm of this work is oriented from the autoencoder. The CNN is applied for a preliminary study, in advance to implementing the autoencoder-based framework.

Chapter 4 illustrates the whole process of the application. The sound signals from single-axis operations of the industrial robot arm are transferred by classic stethoscopes and stored into a desktop computer by connecting USB microphones. The features of each load condition are extracted from spectrograms. Extracted features are put into autoencoders, and the reconstruction errors (RE) are examined for anomaly detection. The data processing and NN procedures are performed via two software languages, MATLAB and Python.

Conclusions from the research and some suggestions for future work are given in Chapter 5. Firstly, the benefits and weaknesses of proposed methods are discussed. Chapter 5 also suggests future work to suggest solutions for some limitations. Finally, some collaborative works in progress are introduced.

CHAPTER 2. LITERATURE REVIEW

2.1 Feature extraction and data-driven monitoring applications

The data-driven strategy requires feature extraction steps to represent the conditions of the target machine in compressed form. Since machines in factories are programmed to perform repetitive operations or usually driven by the gear rotations, meaningful features are extractable periodically. One intuitive way to extract features is examining the statistical property of raw sensor signals in every specific period. For example, the root-mean-square (RMS) and kurtosis of time-domain signals have been utilized to explain the status of the target [13], [16]–[18]. Safizadeh and Latifi [13] extracted various statistical features from vibration signals of the rolling element bearings to form high-dimensional feature vectors, and sorted the most meaningful information by principal component analysis (PCA). They pointed out the feature values on the feature map, then partitioned machine health conditions by visible borders. As such, locating features in the same space can provide a visualization of machine health.

When the monitoring target is driven by rotating machine components such as rolling element bearings or gears, the frequency information is also a crucial part of analysis. However, it is difficult to recognize the frequency information by the time-domain signals and features. To capture the behaviors of the machine in the frequency domain, the feature extraction step is executed after domain conversion, such as the Fourier Transform. Lei et al. [19] extended the statistical feature extraction to the frequency domain, then utilized them for grouping various rolling element bearing conditions. Their approach also includes the bearing signal model derived from [3] in calculating frequency domain features. Beyond the basic Fourier Transform, some advanced techniques such as short-term Fourier Transform (STFT) [20], wavelet transform [21]–[23], and Hilbert-Huang transform (HHT) [24]–[26] are applied. Those techniques are also called

time-frequency analysis, which can cope with non-stationary signals and provide better visualization.

Though extracting features helps to compare the signals in a lower dimension, it implies some information is lost since the features are a compact representation of the raw signal. Sometimes features from a single sensor are unable to explain some characteristics present in the raw data, which degrades the accuracy of the algorithm. To enhance the discriminative performance, several studies employed multiple sensors to predict tool state or life during operations [27]–[30], or to estimate bearing health conditions [31]–[33].

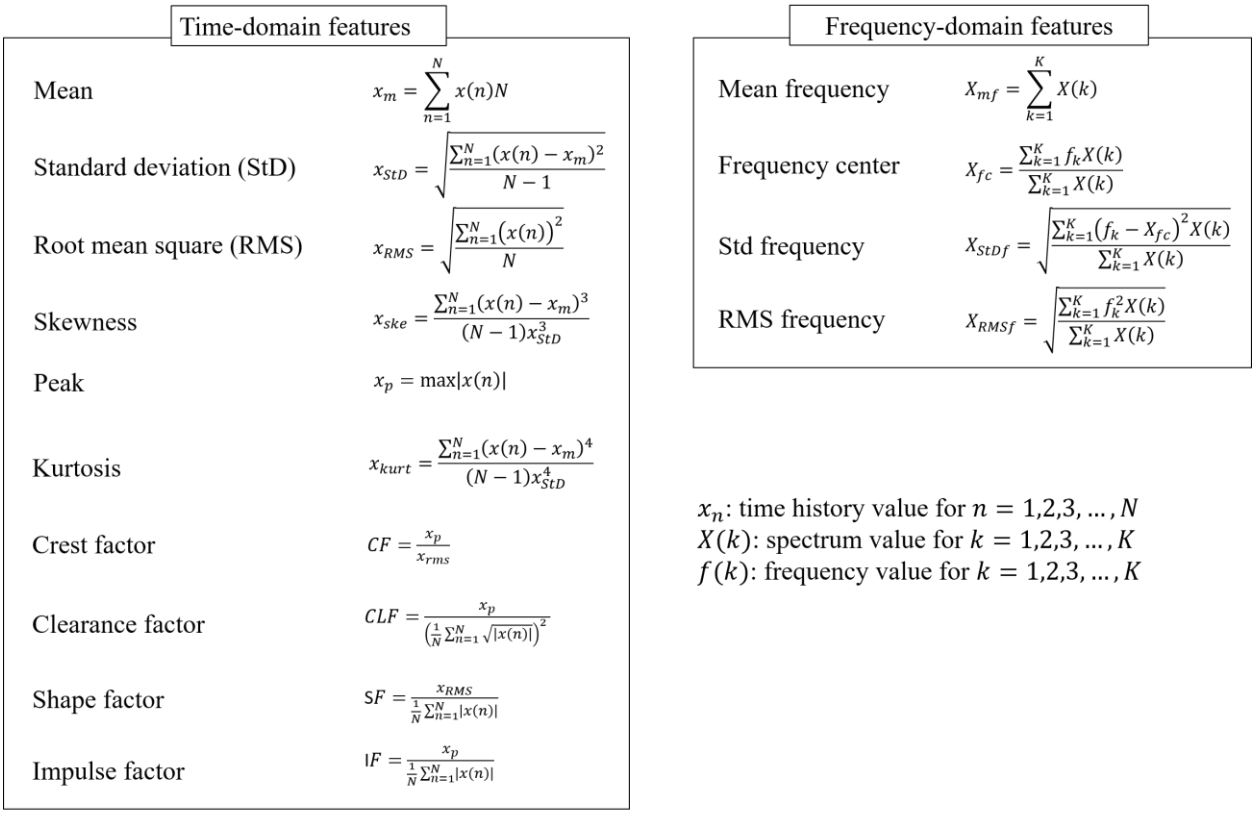


Figure 2.1 To represent characteristics of raw data, features from both time domain and frequency domain are extracted (modified from [19]).

To summarize, the data-driven method derives meaningful features from multiple sensor signals. Figure 2.1 shows several widely-used feature parameters [19] in both time and frequency domains.

The extracted features are projected into a classification map to establish a data-driven fault detection rule. As an example, Figure 2.2 illustrates a 3-class decision map drawn by 2-dimensional features.

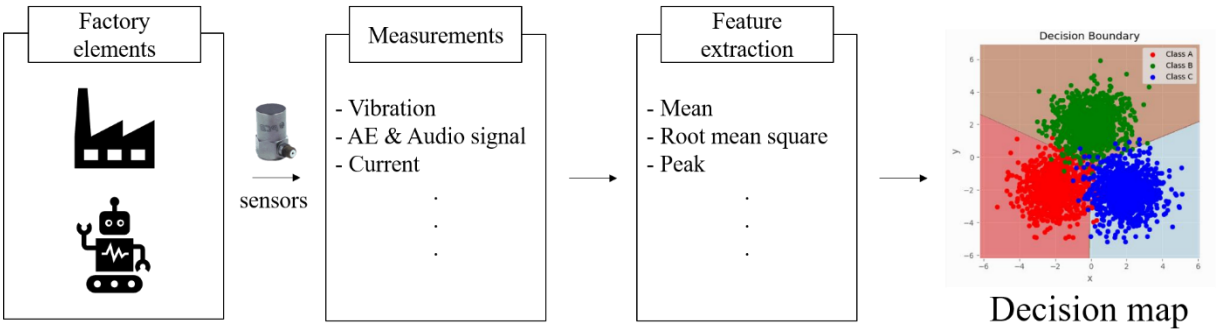


Figure 2.2 By utilizing feature parameters from sensor signals, a decision map that classifies the predefined machine status can be portrayed.

2.2 Neural Network (NN) applications in monitoring problems

Though direct mapping of feature parameters has been successfully implemented for fault classification, the decisions are usually made based on linear classifications. When the decision maps are drawn with nonlinear shapes, it becomes difficult to formulate the rules. Moreover, high-dimensional features such as spectrum vectors or 2D images are intractable to generalize by conventional methods. To assign nonlinearity in monitoring problems, the NN frameworks are applied. By the NN approach, complex and nonlinear relationships can be captured.

In the categorical classification problem, the NN framework also requires normal and abnormal data sets. The conditions to be estimated are digitized as output values, then matched with input signals. Several studies have achieved feasible monitoring solutions for rolling element bearings [34]–[36], gearboxes [37], and gears in the robot manipulator [38]. In those applications, some recognizable defects are imposed to form different condition groups as shown in Figure 2.3.

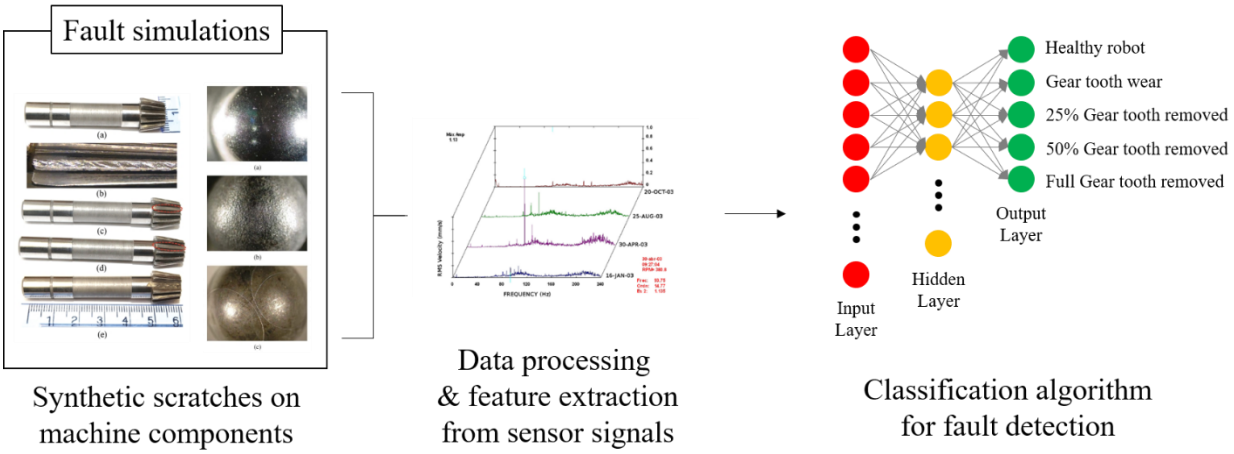


Figure 2.3 Assigning synthetic defects on machine components is a possible approach in classification of machine status using supervised NN. (revised from [38])

As addressed in the previous chapter, the supervised classification approach demands a large amount of abnormal status data for proper training. Again, it is not always possible to predefine existing faults into finite condition classes. To detect the anomaly without categorization, unsupervised clustering can be an alternative solution. Some of the unsupervised NN applications focus on forming clustered groups without prior class information [39], [40], and others on isolating anomalous conditions from normal ones [41], [42].

2.3 Other NN applications in manufacturing area

In NN studies, it is recommended to ensure that the size of the data sets is at least 10 times the number of updatable parameters [43]. However, as the structure of NN gets deeper and more complex, satisfying the recommendation becomes demanding. To resolve this problem, some generative methods are applied to reproduce the data set. Li et al. [44] investigated various direct data augmentation techniques to increase the size of a raw signal by adding random gaussian noise, translating, modulating amplitudes, and time stretching.

Different from direct augmentation methods, some generative NN models such as the Generative Adversarial Network (GAN) [45] and the Variational Autoencoder (VAE) [46] perform data augmentation by statistical methods. In NN approaches toward data augmentation, the NN models learn from compressed features, which is called the latent space. In several studies, the generative NN models were applied [47]–[49] to increase manufacturing data size, as well as to extract features from the latent space.

In addition, NN can be used to develop parameter prediction models, such as cutting force [50] and surface roughness prediction [51], [52]. Though these models cannot be expressed in formulated equations, the NN architecture figures out complex relationships between parameters by learning from large amounts of data.

2.4 Summary of review and extensions to our work

The performance of data-driven monitoring depends both on quantity and quality of the data set. In particular, extracting good features accounts for the majority of NN processes. Therefore, understanding the attributes of the monitoring target is required in designing and running experiments. Our target is monitoring the industrial robot arm during its warm-up operation. Some axes have vertical movements, which involve gravity. Therefore, drawing spectrograms in the time-frequency domain may be able to capture variations in the spectrum, instead of taking the basic Fourier Transform. Since the KUKA KR6 robot arm is a more complex system, it is difficult to define every possible defect. Hence, we apply the autoencoder-based anomaly detection algorithm, rather than applying a supervised classification strategy. By implementing the autoencoder framework, we can approach failure without synthetic fault simulations. Starting from Chapter 3, some required prior knowledge is detailed then applied to investigate experimental data.

CHAPTER 3. BACKGROUND

3.1 Feature extraction from spectrograms

To establish a discriminative algorithm without a physical model, proper features should be extracted to represent different conditions of the machine. As aforementioned, spectrograms are widely used in analysis of vibration signals to recognize trend changes of signals in both time and frequency domains [14], [42], [53], [54]. This spectrogram-based approach is implemented again in this work, to detect the evolutions of spectral patterns as different conditions are assigned.

To derive spectrograms, discrete STFT is computed in every k^{th} window as follows:

$$STFT_k(x[n]) = \frac{1}{N}\sum_{n=n_k}^{n_k+N-1} x[n]w[n-(2R_k-1)m]e^{\frac{j2\pi f(n-n_k)}{N}} \tag{3.1}$$

$$PSD_k = \frac{|STFT_k|^2}{(wcomp)} \tag{3.2}$$

where n is the sample number, n_k is the starting location of each signal block ($n_k = 2(k-1)m$), w is the window function, $wcomp$ is the window compensation, $2m$ is the length of the window, and R_k is the number coefficient of the block which starts from 1. If there is no overlap among blocks, then $R_k = 1,2,3,\ldots$; otherwise it has a specific interval other than 1 (e.g. $R_k = 1, 1.5, 2, 2.5, \ldots$ in case of 50% overlap). From equations (3.1) and (3.2) the power spectral densities (PSDs) can be calculated and concatenated horizontally along the time axis as shown in Figure 3.1. In this work, we apply the Hann window ($wcomp = 0.375$) in each application, with 50% of overlap in spectrograms. Figure 3.1 shows an example spectrogram of an impact test, recorded via electronic stethoscope with 4kHz sampling frequency. By drawing spectrograms, we

can sort the impact moments as features if the spectral patterns vary by the experimental conditions. More specific procedures for feature extraction are detailed in later parts of each application.

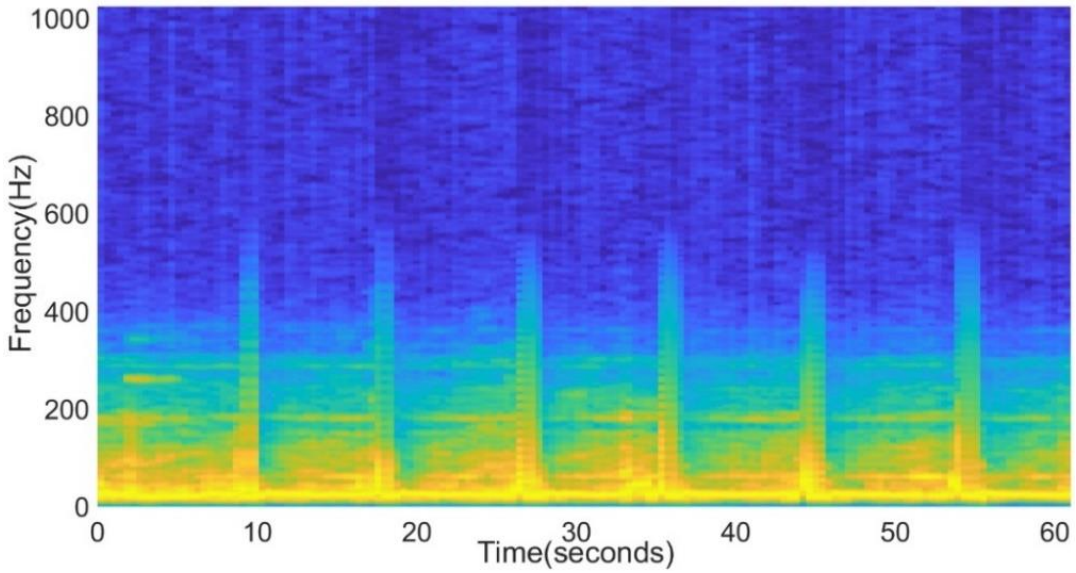


Figure 3.1 A spectrogram is derived from impact test using an electronic stethoscope, to show an example of time-varying spectra.

3.2 Audio system identification using linear chirp

The identification of sound transferring systems is required not only to specify the frequency band for use, but to confirm whether there is any interference of high-frequency noise. To verify both requirements, the linear sine chirp signal is utilized to identify the stethoscope systems.

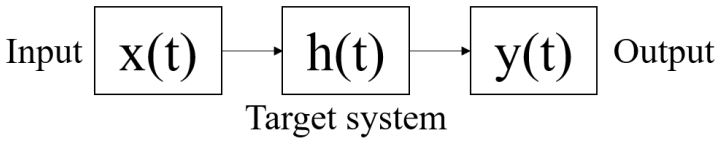


Figure 3.2 The sound transferring system in this work is assumed to be linear and time-invariant.

To begin with, we assume that the systems are approximately linear and time-invariant (LTI) as shown in Figure 3.2. Theoretically, the measured output signal $y(t)$ is acquired by convolving the input signal $x(t)$ and the impulse response of the system $h(t)$:

$$y(t) = x(t) * h(t) \tag{3.3}$$

where the symbol '$*$' indicates the convolution process.

Assume there is any $x'(t)$ such that convolution with $x(t)$ produces Dirac's delta function:

$$x(t) * x'(t) = \delta(t) \tag{3.4}$$

Convolving $x'(t)$ on both sides of (3.3) yields calculation of $h(t)$ and $H(f)$ as follows:

$$\begin{aligned} x'(t) * y(t) &= h(t) \\ H(f) &= \mathcal{F}(h(t)) \end{aligned} \tag{3.5}$$

where $\mathcal{F}$ designates the Fourier Transform. However, it is difficult to simulate the exact direct delta function, since it has infinite length in the frequency domain. Instead, some finite-length rectangular spectra with 0dB magnitude can be designed, which are overlapped to the target bandwidth. The linear chirp signal is tractable to approximate the rectangular spectrum, and thereby is widely used for system identification purposes [55], [56]. In the following subsections, the characteristics of linear chirps are investigated, then the identification process for our audio transfer system are described.

3.2.1 Approximating rectangular frequency response

The linear chirp signal has a linearly increasing instantaneous frequency, and can be written as:

$$s(t) = sin\left(\omega_1 t + \frac{t^2(\omega_2 - \omega_1)}{2T}\right) \tag{3.6}$$

where ω_1 is start frequency, ω_2 is end frequency, and T is total duration. In the ideal case, the magnitude of the linear chirp signal should be flat within the designated bandwidth (ω_1, ω_2) [57], hence a simple amplitude modulation could be done to result in a flat 0dB frequency response.

Geyer [58] and Aldridge [59] investigated the autocorrelation $R_{ss}(\tau)$ of linear chirp signals and approximated it to the sinc function in the time domain, which has a rectangular magnitude in the frequency domain. Theoretically, it also has zero-phase attribute since the autocorrelation $R_{ss}(\tau)$ is given by the convolution of $s(t)$ and $s(-t)$, which yields the product of $S(f)$ and $S^*(f)$ in the frequency domain as follows:

$$\begin{aligned} R_{ss}(\tau) &= \int_{-\infty}^{\infty} \big(s(t)s(\tau+t)\big)dt \\ &= \int_{-\infty}^{\infty} \big(s(-t)s(\tau-t)\big)dt = s(-\tau) * s(\tau) \end{aligned} \tag{3.7}$$

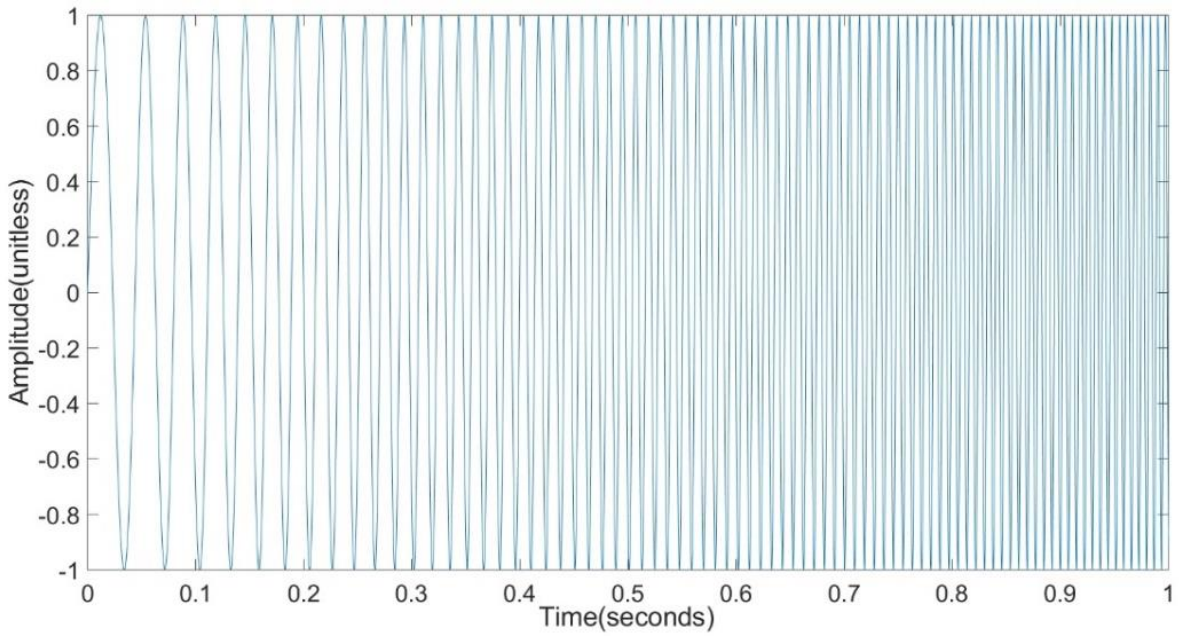
$$\begin{aligned} \mathcal{F}\big(R_{ss}(\tau)\big) &= \mathcal{F}\big(s(\tau) * s(-\tau)\big) = S(f)S^*(f) \\ &= |S(f)|^2 \end{aligned} \tag{3.8}$$

where $\mathcal{F}$ yields the Fourier Transform and $S^*(f)$ yields the complex conjugate of $S(f)$. Equation (3.8) implies that $\mathcal{F}\big(R_{ss}(t)\big)$ is real, and therefore the phase is zero. Following the above process, we can design the proper inverse filter of a linear chirp by reversing and amplifying the original chirp, so that we can measure the impulse response of the target system within specific frequency bandwidth.

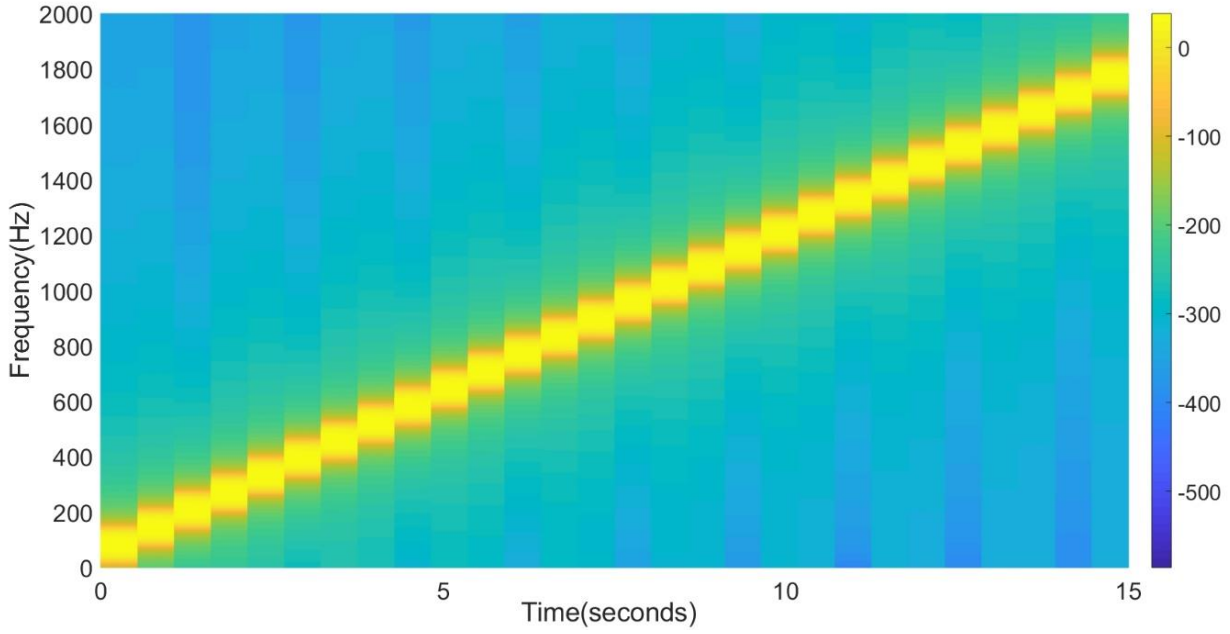
3.2.2 System identification procedure

Our bandwidth of interest for the system identification is up to 2kHz, which is sufficiently broader than the recommended usage of classic stethoscopes (500Hz). We designed the input signal within 20Hz - 1900Hz with 15 seconds emission duration, by applying equation (3.6). Figure 3.3 illustrates the characteristics of the designed linear chirp signal in multiple domains. To derive the

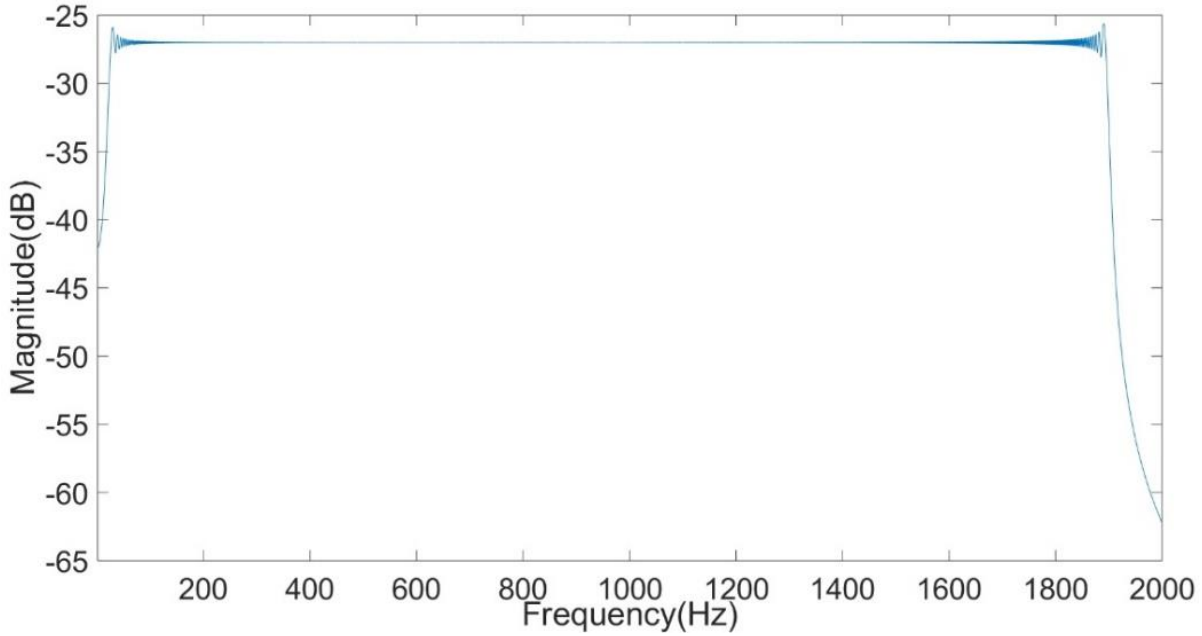
spectrogram in Figure 3.3-(b), a 48000-point (1 second) Hann window with 50% overlap (24000 points) was applied.



(a)



(b)



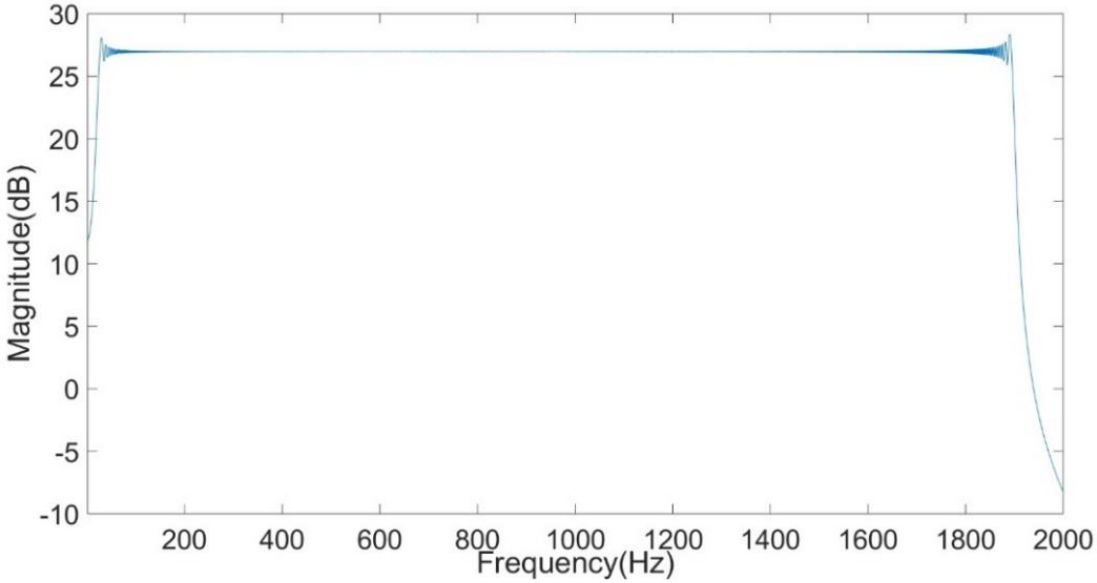
(c)

Figure 3.3 Designed linear sine chirp signal is expressed in different ways:
(a) 1-second time history, (b) spectrogram, and (c) magnitude of the Fourier Transform.

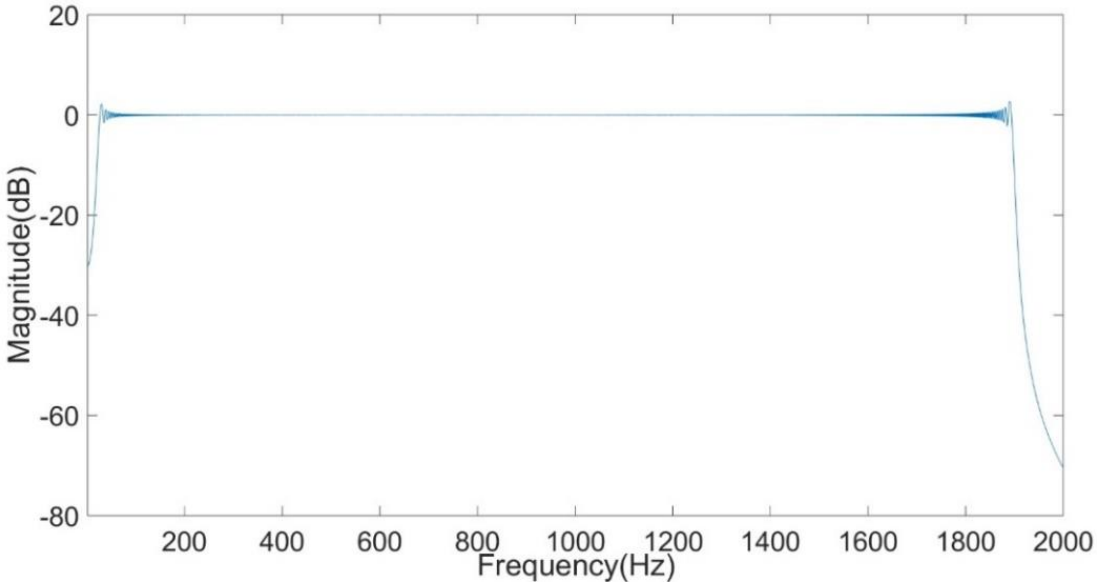
As shown in Figure 3.3-(c), we need amplitude modulation so that $mS(f)S^*(f)$ has flat 0dB magnitude within 20Hz - 1900Hz. We have selected the magnitude value at the middle of the range 960Hz, thereby the inverse filter is designed by:

$$invs(t) = \frac{1}{|S(f_{960Hz})|^2} s(-t) \tag{3.9}$$

where $|S(f_{960Hz})|$ is amplitude of $S(f)$ at 960Hz. Figure 3.4 illustrates the final inverse filter for the designed linear chirp input and the product with $S(f)$.



(a)



(b)

Figure 3.4 Magnitude of the Fourier Transform of:
(a) the inverse signal, and (b) the convolution of the linear sweep with the inverse signal.

By combining the above result and equation (3.5), the frequency response $H(f)$ can be approximated by computing the product of $\mathcal{F}\big(invs(t)\big)$ and $Y(f)$:

$$H(f) = \mathcal{F}\big(invs(t)\big)Y(f) \tag{3.10}$$

The system identification results and verifications for our system are detailed in Chapter 4.

3.3 Neural Network (NN) designs

The NN is a nonlinear data processing architecture, inspired from the perception of human beings through neurons [60]. Studies in the early stage of NN have succeeded in demonstrating the feasibility in several classification and regression problems. However, they were criticized for their inefficiency resulting from the lack of proper computing technologies. For this reason, the NN-based approaches have not become popular until the breakthrough use of graphics processors (GPUs) for machine learning [61].

After the invention of distributive computation enabled by GPUs, the NN strategies have prospered in various fields of studies. In advance to illustrate the proposed algorithm in detail, the basic update algorithm of NN is described in the following contents. Further, the concepts and structures of the autoencoder and Convolutional Neural Network (CNN) are introduced.

3.3.1 Basic learning process

The term “training” or “learning” of the NN comes from minimizing errors between the output of the NN and designated target value. A popular “learning” method for NN is combination of “gradient descent” and “back propagation”. In the following parts below, the back-propagation processes are briefly explained based upon the activation and loss functions.

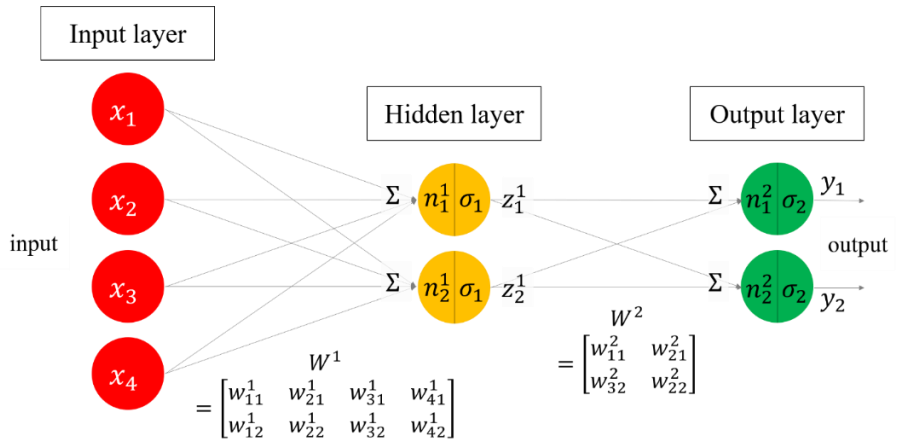


Figure 3.5 A NN consists of several layers, weights, and activation functions.

The structure of NN is composed of an input layer, some hidden layers, and an output layer. Among the layers, weights and activation functions are assigned to deliver input values to output layers. A simple structure of NN is described in Figure 3.5.

In the above graph, $X = [x_1, x_2, x_3, x_4]^T$ is the input, W^1, W^2 are the weights, σ_1, σ_2 are the activation functions, $Z^1 = [z_1^1, z_2^1]^T$ is the intermittent output from hidden layer, and $Y = [y_1, y_2]^T$ is the computed output through NN. Provided that we have k sampled data sets, and intended target values $T = [t_1, t_2]^T$ for every input X, then k input-output pairs of samples exist; $S^i = [X^i, T^i]$, where $i = 1,2,3, \dots, k$. The purpose of training the NN is to minimize the discrepancy between the output (Y) and target value (T) of each data pair, by updating weights (W^1, W^2) to desirable values. The difference between Y and T is also named "loss", hence the NN updates itself to minimize the loss function $E(Y, T)$.

Generally, the NN processes can be divided into feed-forwarding and updating. As described in (3.11), the feed-forwarding process computes output values by designated weights and activations:

$$\begin{aligned} N^1 &= [n_1^1, n_2^1]^T = W^1 X \\ Z^1 &= [z_1^1, z_2^1]^T = \sigma_1(N^1) = [\sigma_1(n_1^1), \sigma_1(n_2^1)]^T \\ N^2 &= [n_1^2, n_2^2]^T = W^2 Z^1 \\ Y &= [y_1, y_2]^T = \sigma_2(N^2) = [\sigma_2(n_1^2), \sigma_2(n_2^2)]^T \end{aligned} \tag{3.11}$$

Next, the gradients are calculated to update weights, using the output Y and corresponding target T. Before depicting the back-propagation algorithm for weight update, the loss function is defined as $E(Y, T) = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i)^2$ with $m = 2$ for simplicity. According to [62], the weight gradients ΔW are represented as:

$$\Delta w_{i,j}^{k} = -\mu \frac{\partial E(Y,T)}{\partial w_{i,j}^{k}} \tag{3.12}$$

where μ is the learning rate and $w_{i,j}^{k}$ is the weight value from ith cell of kth layer to jth cell of $(k+1)$th layer (see Figure 3.5 for better understanding).

Equation (3.12) can be expanded by applying the chain rule. First, we calculate $\Delta w_{i,j}^{2}$ which are just before the output layer:

$$\Delta w_{i,j}^{2} = -\mu \frac{\partial E(Y,T)}{\partial w_{i,j}^{2}} = -\mu \frac{\partial E(Y,T)}{\partial n_{j}^{2}} \frac{\partial n_{j}^{2}}{\partial w_{i,j}^{2}} \tag{3.13}$$

$$\frac{\partial n_{j}^{2}}{\partial w_{i,j}^{2}} = z_{i}^{1} \; (\because n_{j}^{2} = w_{1,j}^{2} z_{1}^{1} + w_{2,j}^{2} z_{2}^{1}) \tag{3.14}$$

$$-\delta_{j}^{2} \equiv \frac{\partial E(Y,T)}{\partial n_{j}^{2}} = \frac{\partial E(Y,T)}{\partial y_{j}} \frac{\partial y_{j}}{\partial n_{j}^{2}} \tag{3.15}$$

$$\frac{\partial E(Y,T)}{\partial y_{j}} = -(t_{j} - y_{j}) \tag{3.16}$$

$$\frac{\partial y_{j}}{\partial n_{j}^{2}} = \sigma_{2}^{(1)}(n_{j}^{2}) \tag{3.17}$$

where $\sigma_{2}^{(1)}$ is the first derivative of σ_{2}. Substituting the components by equations (3.14) - (3.17), equation (3.12) can be rewritten as follows:

$$\Delta w_{i,j}^{2} = \mu \cdot (t_{j} - y_{j}) \cdot \sigma_{2}^{(1)}(n_{j}^{2}) \cdot z_{i}^{1} \tag{3.18}$$

Next, $\Delta w_{i,j}^{1}$ are computed in a similar way, but different results are derived:

$$\Delta w_{i,j}^{1} = -\mu \frac{\partial E(Y,T)}{\partial w_{i,j}^{1}} = -\mu \frac{\partial E(Y,T)}{\partial n_{j}^{1}} \frac{\partial n_{j}^{1}}{\partial w_{i,j}^{1}} \tag{3.19}$$

$$\frac{\partial n_{j}^{1}}{\partial w_{i,j}^{1}} = x_i \ (\because n_{j}^{1} = \sum_{i=1}^{4} w_{i,j}^{1} x_{i}^{1}) \tag{3.20}$$

$$-\delta_{j}^{1} \equiv \frac{\partial E(Y,T)}{\partial n_{j}^{1}} = \frac{\partial E(Y,T)}{\partial z_{j}^{1}} \frac{\partial z_{j}^{1}}{\partial n_{j}^{1}} \tag{3.21}$$

$$\frac{\partial E(Y,T)}{\partial z_{j}^{1}} = \sum_{i=1}^{2} -\delta_{i}^{2} w_{i,j}^{2} \tag{3.22}$$

$$\frac{\partial z_{j}^{1}}{\partial n_{j}^{1}} = \sigma_{1}^{(1)}(n_{j}^{1}) \tag{3.23}$$

where δ_{i}^{2} is defined in (3.14). Combining equations (3.19) - (3.23) altogether yields:

$$\Delta w_{i,j}^{1} = \mu \cdot \left(\sum_{i=1}^{2} \delta_{i}^{2} w_{i,j}^{2} \right) \cdot \sigma_{1}^{(1)}\left(n_{j}^{1}\right) \cdot x_i \tag{3.24}$$

We can recognize that the updates of weights are derived in different ways with respect to where the weights are located. Generally, the update derivations in input-hidden and hidden-hidden layer intervals share their computation procedures, while those in hidden-output layer intervals are calculated in a unique way.

By investigating the overall update process through equations (3.11) - (3.24), we can also recall that the updates in later weights are required to renew the earlier weights. In other words, the direction of weight update is opposite to the feed-forward process. For this reason, the name of the learning process of NN is known as “back-propagation”. More generalized representation of back-propagation is described in [62].

Likewise in previous NN research, the convergence analysis using the back-propagation algorithm was successfully performed in [63]. However, it is not guaranteed whether the NN solution leads the losses to the global or local minimum. Furthermore, there is no established method for organizing the optimal NN, hence we need some iterations in selecting the hyperparameters and structures of NN [64]. In general, increasing the depth and dimension of layers enhances the performance on minimizing losses. Even so, indiscreet complication of the NN structure slows down the computations and sometimes yields an "overfitting problem", since more weight parameters are included. Therefore, the users need to find a suitable shape of NN for their own purposes. These efforts toward selecting the hyperparameters of NN are called the tuning process.

In this work, two popular architectures of NN were implemented: the autoencoder and the CNN. The autoencoder was utilized as the main algorithm for anomaly detection, and the CNN was utilized for preliminary study of sound signal analysis. The tuning processes are detailed in later chapters of each application.

3.3.2 Autoencoder

The autoencoder is a NN architecture of semi-supervised learning, which is a popular approach in image reconstruction and denoising. The term "semi-supervised" comes from the aspect that the autoencoder makes use of inputs as targets for reference. The encoding stage finds a compressed interpretation of the original input, and the decoding stage produces an output that mimics the original input.

Assume that there is a sequence composed of n-dimensional vectors $X^{(i)}$'s, $\{X^{(1)}, X^{(2)}, X^{(3)}, \dots\}$, where $X^{(i)} \in R^n$. The autoencoder tries to adapt output $Y^{(i)}$ close to the original input $X^{(i)}$. Figure 3.6 describes the simplest case of a single-layer autoencoder.

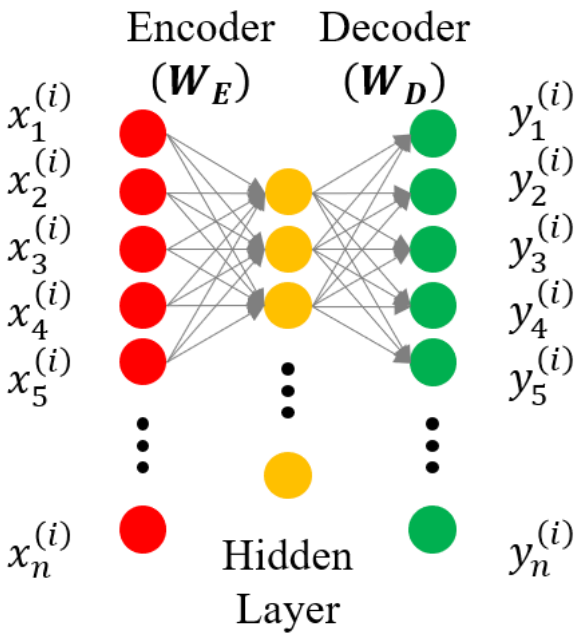


Figure 3.6 An autoencoder receives an input vector to learn the compressed form by encoding, then reconstructs the input by decoding.

The error between $X^{(i)}$ and $Y^{(i)}$ is named the Reconstruction Error (RE), and is also represented as the loss function of the autoencoder. The feedforward process is as follows:

$$\begin{aligned} (Hidden\ Layer) \quad & Z^{(i)} = \sigma\left(W_E X^{(i)}\right) \\ (Output\ Layer) \quad & Y^{(i)} = \sigma\left(W_D Z^{(i)}\right) \end{aligned} \tag{3.25}$$

where W_E and W_D are the weight arrays of encoder and decoder, respectively, σ is the activation function.

The error $Loss\left(x^{(i)}, y^{(i)}\right)$ is computed after the output layer, and the "learning" minimizes the $Loss$ so that reconstructed images get similar to the original inputs. In this work, the activation functions and loss function were designated as follows:

$$(Activation) \quad \sigma_{enc}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$(Activation) \quad \sigma_{dec}(x) = \frac{1}{1 + e^{-x}} \tag{3.26}$$

$$(RE) \quad Loss(x^i, y^i) = \frac{1}{n}\sum_{k=1}^{n}\left(x_k^{(i)} - y_k^{(i)}\right)^2$$

The activation functions $\sigma_{enc}(x)$ and $\sigma_{dec}(x)$ are also known as the Rectified Linear Unit (ReLU) and the sigmoid function, respectively. In this application, $\sigma_{enc}(x)$ was used for encoding (data compression), and $\sigma_{dec}(x)$ was used for decoding (input reconstruction). For the loss function, we assigned a mean-squared error. Herein the RE can be used for the anomaly detection algorithm [42], [65], [66]. To classify anomalous signals from normal signals by RE, the autoencoder should be trained purely with normal signals. After training without abnormal signals, the autoencoder produces larger RE when "unseen" data are fed in as input. Since the collected REs in each operation cycle have their own distributions, we can compare the RE distributions of multiple cycles then decide whether there is any anomalous signal in some cycles.

3.3.3 Convolutional Neural Network (CNN)

The CNN is a NN framework of supervised learning, which is specialized for supervised classification problems. The CNN takes predefined classes as output ("car", "dog", "cat", "airplane", etc.), and aims to predict the classes correctly by feeding forward an input image. In general NN structures, the number of weights between two layers equals the product of the number of cells in neighboring layers. On the other hand, the size of weights in the CNN architecture is flexible. Instead of computing pointwise multiplications, the weight arrays in CNN sweeps the

input of the previous layer, which accounts for the convolutional process. Figure 3.7 portrays a simple configuration of CNN.

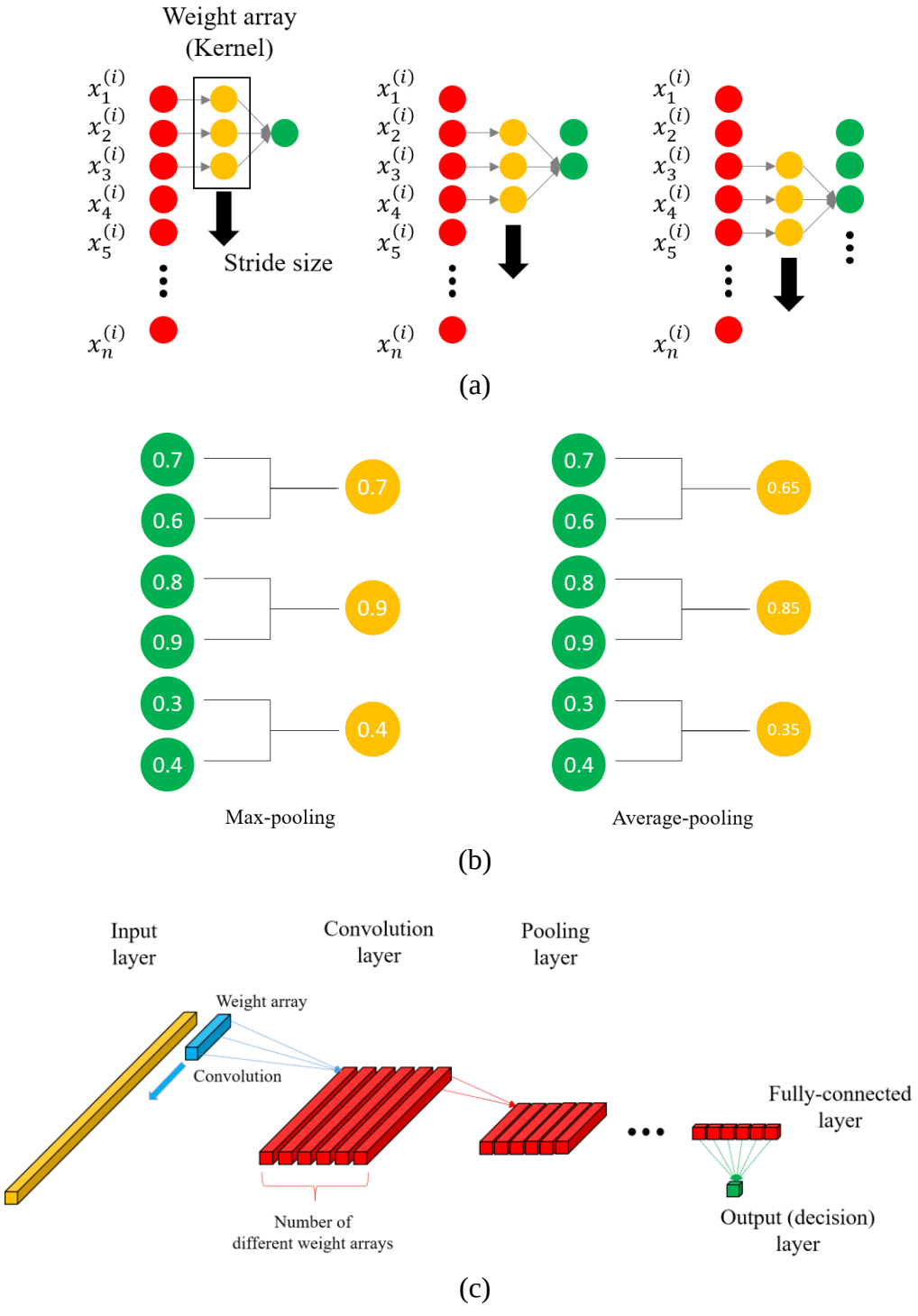


Figure 3.7 CNN includes (a) the convolution layers for more complexity, and (b) the pooling layers for data compression. As a combination of (a) and (b), an example of CNN procedure is described in (c).

A CNN usually consist of the convolution layer, the pooling layer, and the fully-connected layer (FCL). In the convolution layers, the weight arrays feed the inputs forward to the next layer by computing convolution products. Figure 3.7-(a) depicts a process in the convolution layer, using a weight array size of 3, and stride size 1. When i^{th} input vector is fed into the CNN, the value of j^{th} cell in the next layer is computed by:

$$z_j^{(i)} = \sigma\left(W\left(x_j^{(i)} x_{j+1}^{(i)} x_{j+2}^{(i)}\right)\right) \tag{3.27}$$

where σ is activation function. In this work, the ReLU function described in the previous chapter was implemented as the activation of the CNN.

The pooling layers are responsible for data compression, by averaging or selecting the maximum values. As described in Figure 3.7-(b), the pooling layers reduce the dimension of incoming information. The FCL performs the same operation as the general NN, producing the final output vector which is related with the decision. Combining those layers altogether, the CNN projects high-dimensional input vectors into decision vectors, usually composed of 0, 1 and other digits.

As stated, researchers in engineering fields utilized CNN for categorical fault detection of machine components, by imposing some synthetic faults on the target and gathering faulty signals from them to train the CNN. However, it is difficult to simulate every sort of defect. Furthermore, running experiments on defective machines without replacement may lead to irreversible breakdown. Hence there is a limitation in collecting a large size of faulty data sets. For this reason, the CNN framework was implemented as a preliminary study to verify the possibility of categorization of sound signals from different conditions. More detailed derivation of backpropagation in CNN can be found in [67], [68].

CHAPTER 4. EXPERIMENTAL PROCEDURE

We investigated the sound signals which were collected through the stethoscope-microphone system, while the robot arm was operating single-axis movements with different load conditions. The excessive load conditions were considered as anomaly, which are target outliers to be detected. The features were extracted from the sound spectrograms, then fed into autoencoders for training and testing. After extracting the features, the autoencoder structure was established to compute reconstruction error (RE) in order to detect anomalies. Experiments in both calm and noisy conditions were investigated in order to demonstrate the performance of the proposed method in real factory environments.

4.1 Algorithm scheme

Based upon the preliminary knowledge of the autoencoder, the anomaly detection algorithm can be organized as follows:

1) The signals which are considered as “normal” or “acceptable” are collected to train the autoencoder. The dimension of the input and the depth of the autoencoder are selected by some iterations. Features are extracted from sound spectrograms, and then fed into the autoencoders. The detailed procedure is described in later sections.

2) After training the autoencoder, the signals from both “normal” and “abnormal” conditions are collected for testing. Again, the features are extracted in the same way, then fed forward into the trained autoencoder.

3) The REs are computed and compared in compliance with the load conditions. Some thresholds ε are set up between “normal” and “abnormal” status in each axis.

Figure 4.1 illustrates the scheme of the proposed method.

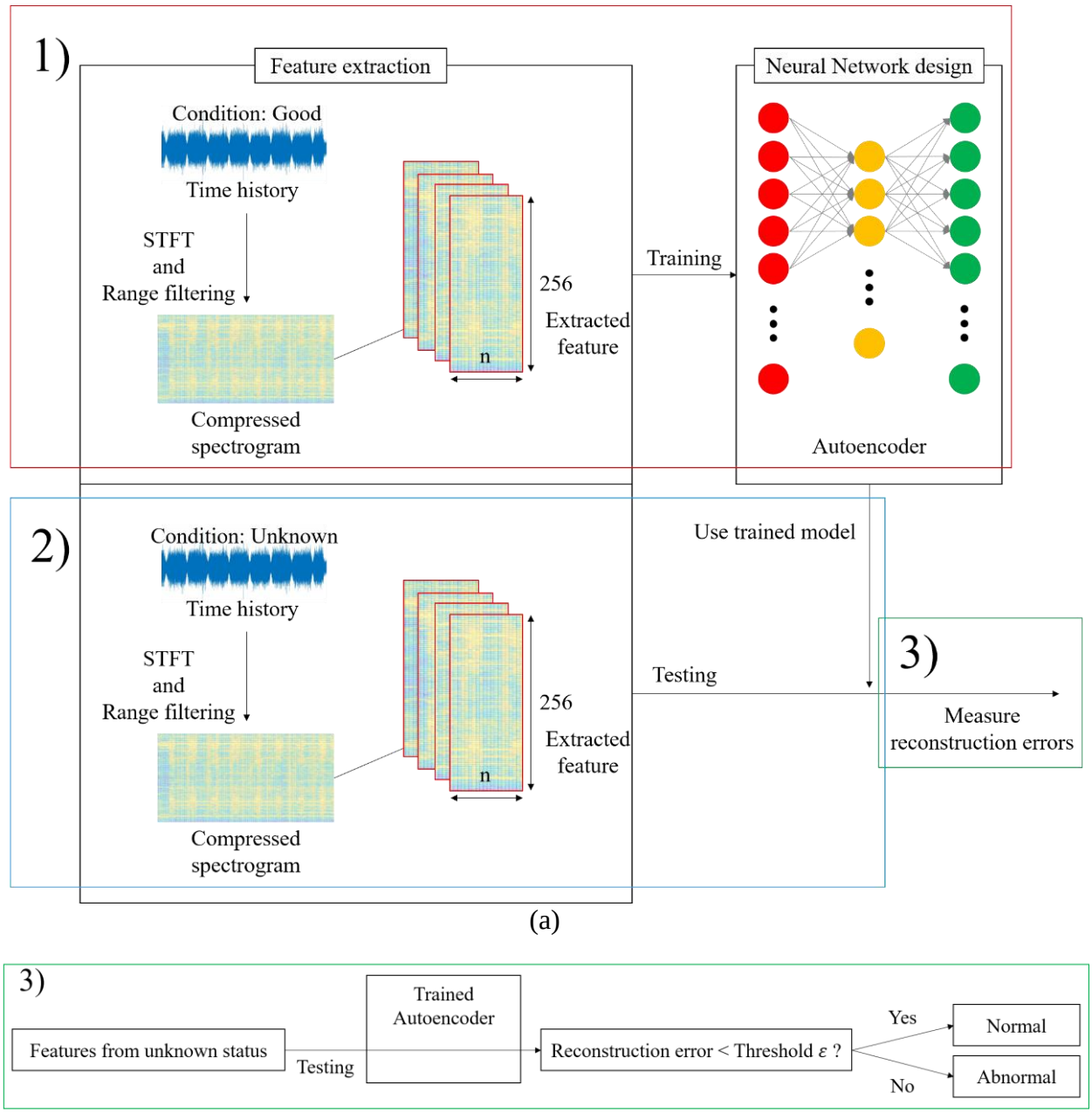
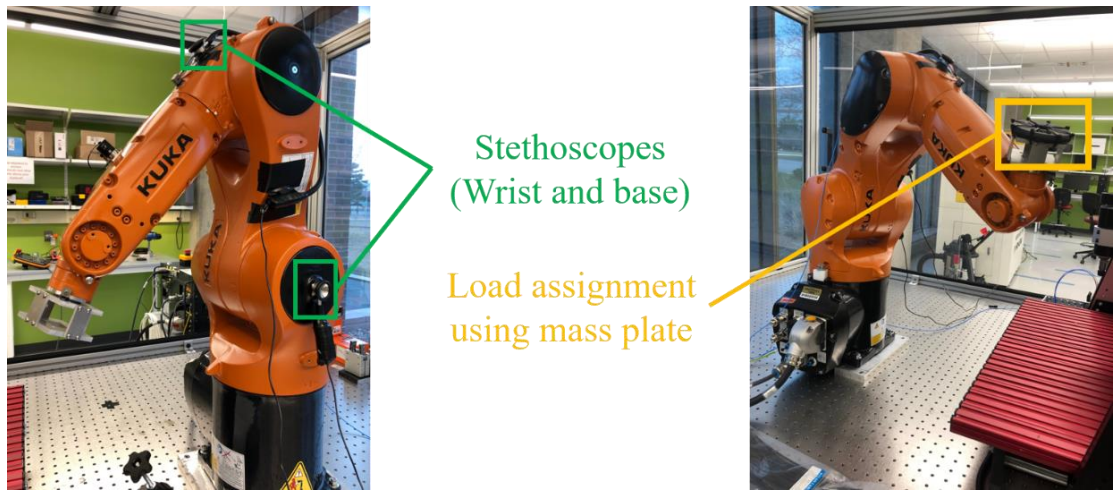


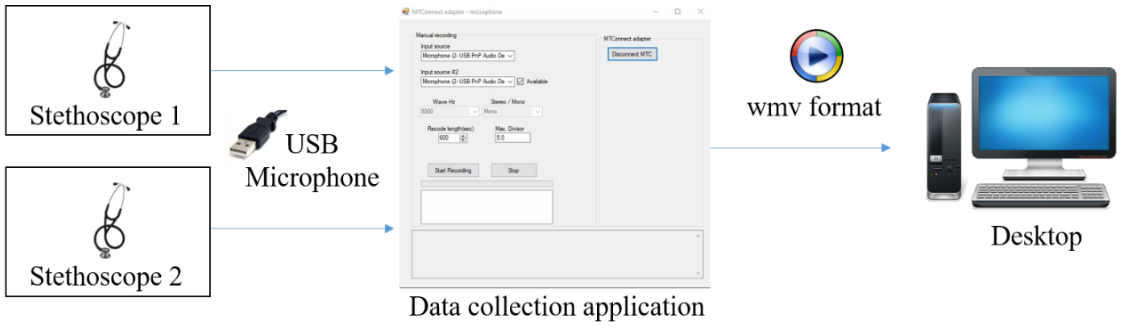
Figure 4.1 In our anomaly detection algorithm, (a) autoencoders are trained by features from good robot conditions. After training, features both from good and bad conditions are fed into the autoencoders to measure RE values. (b) By comparing the distributions of RE, a threshold can be set to distinguish the normal and the abnormal status.

4.2 Experimental setup

The experiments are performed using KUKA KR6 6-DOF industrial robot. The robot is programmed to operate 6 single axial movements. For sound signal acquisition, two 3M Littmann-Classic II stethoscopes were attached, which have frequency range up to 500Hz, on the wrist and the base. The sound signals were transferred to the PC by connecting FIFINE USB microphone. The load conditions were assigned using different mass plates on the end effector, in order to simulate abnormal friction of joints (0, 1.25, 2.5, 5.0, 7.5, 10.0, 12.5lbs). The hardware setup and frequency specifications of sensors are shown in Figure 4.2 and Table 4.1.



(a) (b)



(c)

Figure 4.2 For the experiments, (a) two stethoscopes are attached at the wrist and the base of robot arm, (b) different load conditions are assigned at the end of the manipulator. (c) The captured sound signals are delivered to a desktop through USB microphones.

Table 4.1 Specifications of sensors used in experiments.

Sensor	Model name	Sampling frequency	Response range
Stethoscope	Littmann 3M Classic II	-	20Hz - 500Hz
Microphone	FIFINE K053	48kHz	20Hz - 16000Hz

As shown in Table 4.1, the sound signals were collected with sampling frequency of 48kHz, which is unnecessarily broader than the frequency response range of the stethoscopes. To decide on the appropriate frequency band, the audio system identification is performed in the next part. For noise isolation purpose, the contact locations of robot-stethoscope and stethoscope-microphone are sealed with silicone and 3.2mm Peacemaker Sound Barrier, respectively (Figure 4.3). Note that the sound transfer system identifications were performed before the noise rejecting management.

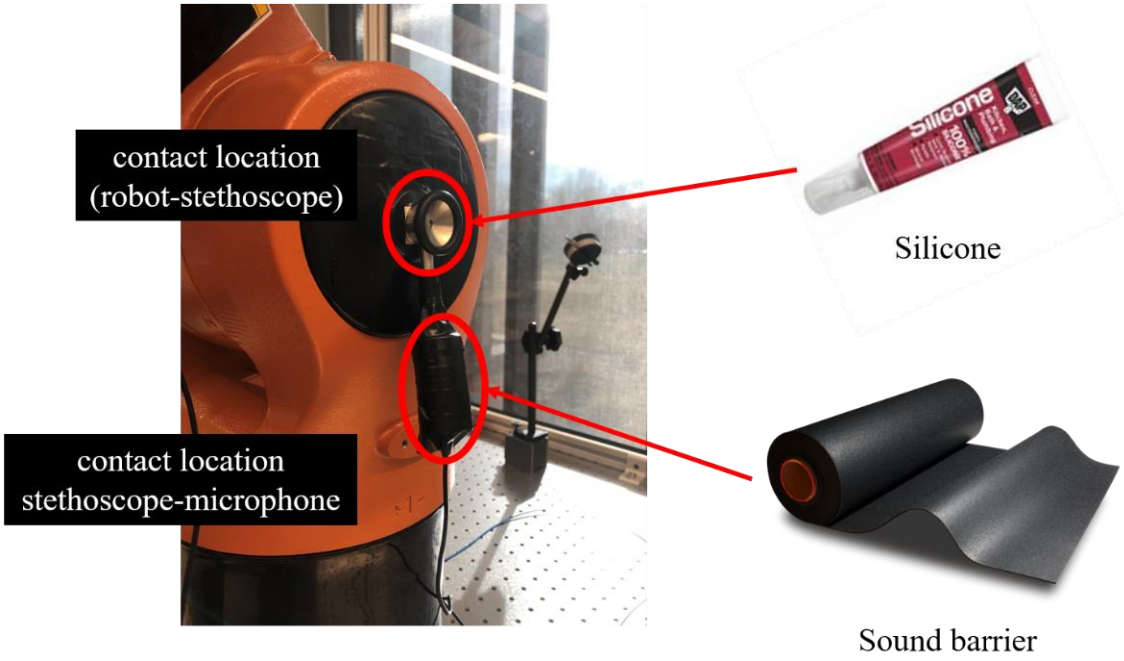
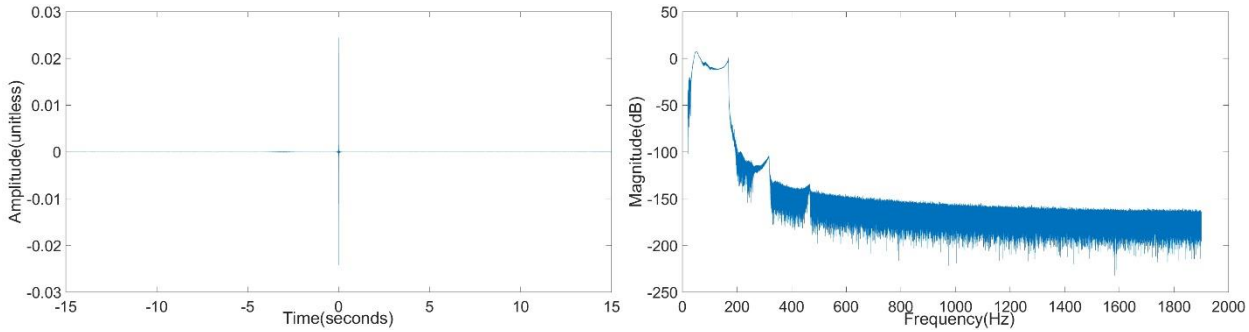


Figure 4.3 Silicone and sound barrier sealing are applied in order to reject external noise.

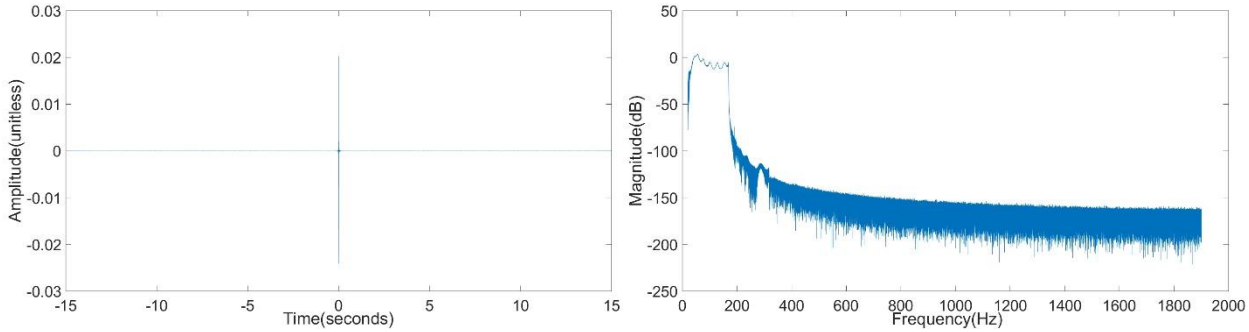
4.3 Audio system identification

To measure the output signal in the PC, the input signal was designed by following Chapter 3.2, then was transferred through incorporated stethoscope-microphone systems. For sound emission,

ETEKCITY ROVERBEATS T3 speaker was utilized to deliver the input signal designed from MATLAB. Received output signals were inverted to impulse responses $h(t)$ by taking the inverse Fourier Transform (IFFT) of $H(f)$, derived from equation (3.9).



(a)

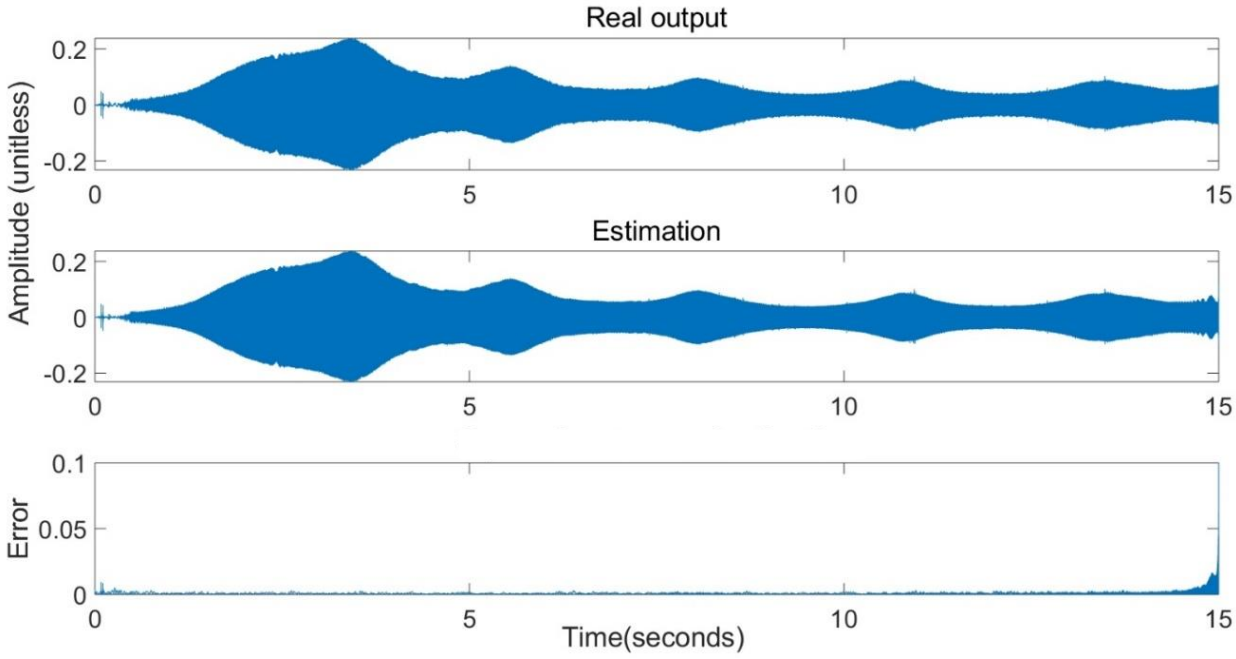


(b)

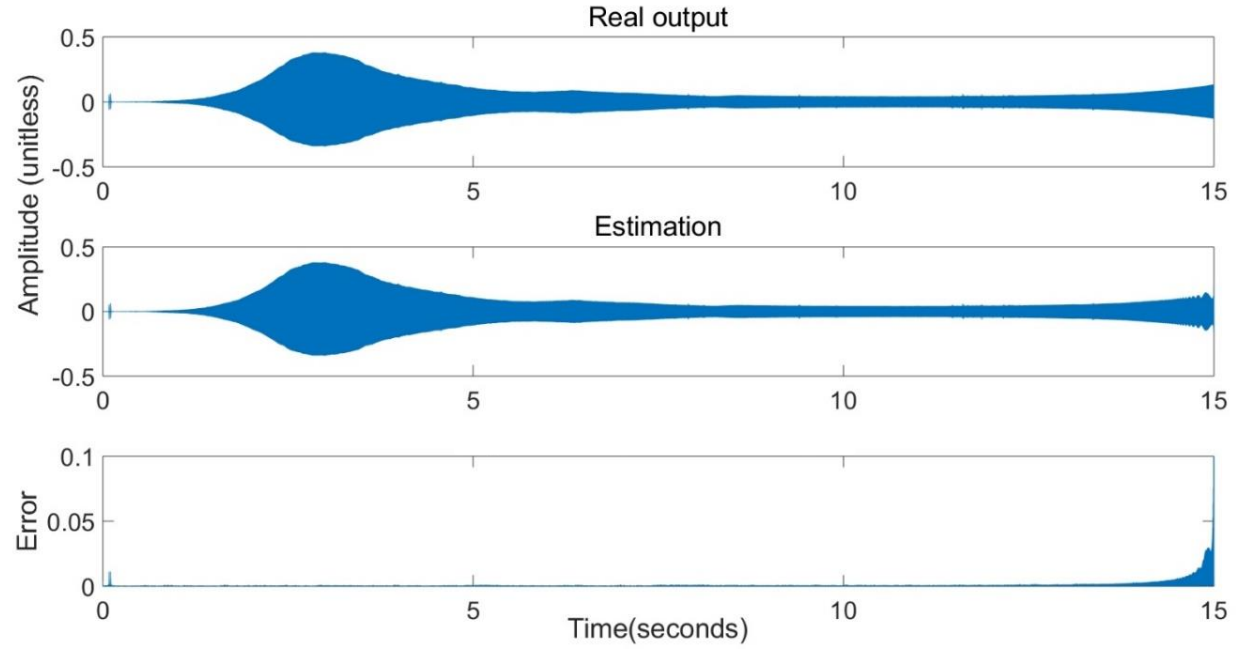
Figure 4.4 Approximated impulse response *h(t)* and frequency response *20log10|H(f)|* of the stethoscope at: (a) wrist, and (b) base.

Figure 4.4 shows approximated impulse responses $h(t)$ and frequency responses $H(f)$. For verification, the real outputs $y_{real}(t)$ and estimations $y_{est}(t) = s(t) * h(t)$ are compared in Figure 4.5 with errors $e(t) = |y_{real}(t) - y_{est}(t)|$.

From the system identification process, we narrow down the range of frequency for data analysis. Since we set the data compression ratio as 50% in the later autoencoder processes, we restricted the frequency response range up to power of 2 ($2^{\mathrm{k}} - 1$) with 1Hz frequency resolution. As shown in Figure 4.4, it is sufficient to select 0Hz - 255Hz to get meaningful information, whereas 2^7(up to 127Hz) is lossy, and 2^9 (up to 511Hz) is redundant.



(a)



(b)

Figure 4.5 Real output $y(t)$ and estimation $y'(t)$ are compared with errors between them at: (a) wrist, and (b) base.

4.4 Data acquisition and feature extraction

The KUKA KR6 robot arm executes axial operations along every single joint (axis 1 – axis 6). The robot repeats the axial operation 5 times in each cycle, and the sound signals were captured by stethoscopes then delivered to PC through microphones (sampling frequency: 48kHz). To test the feasibility of the algorithm in real factory environment, experiments were fulfilled in both calm and noisy conditions. The noisy environments were simulated by turning on the audio records from real factories nearby the robot arm.

Acquired signals were converted into spectrogram every second in accordance with equation (3.1), with a Hann window and 50% overlap applied ($m = 24000$ and $R_k = 1.0, 1.5, 2.0, 2.5$ ...). The features were extracted using the following procedure:

1) The PSDs were filtered up to 255Hz. In this step, selecting the first 256 points in each PSD vector accounts for selecting the bandwidth up to 255Hz, since the frequency resolution of the raw spectrogram is 1Hz. After this, every single PSD has length of 256.

2) The filtered PSDs were normalized to have values within 0 and 1. Equation (4.1) maps filtered PSDs into normalized vectors. This step is required since the scale affects performance of the autoencoder.

$$PSD_{norm}^{k}(i) = \frac{PSD^{k}(i) - min(PSD^{k})}{max(PSD^{k}) - min(PSD^{k})} \tag{4.1}$$

where $PSD^{k}(i)$ is i^{th} component in k^{th} PSD of spectrogram.

3) After normalization, successive PSDs were concatenated along the time axis. Since the collected signals were not stationary, training the autoencoder with 1-dimensional PSDs may lead to confusion between normal and abnormal conditions. Rather, multiple PSDs are combined into 2-dimensional PSD sequences in order to construct a 2-dimensional input for the autoencoder. The

tuning process for the time-length n of spectrogram patch is described in later sections. The concatenation of 1-dimensional PSDs results in 256-by-n-2D images. Therefore, k^{th} 2-dimensional feature $F(k)$ is configured through equation (4.2):

$$F(k) = [PSD_{norm}^{k} \mid PSD_{norm}^{k+1} \mid \dots \mid PSD_{norm}^{k+m}],$$
$$k = 1,2,3, \dots (m-n) \tag{4.2}$$

where m is the place of the first 1-dimensional PSD vector and n is the time-length of the 2-dimensional feature. The k^{th} feature starts from k^{th} PSD, not $(1+8(k-1))^{th}$, in order to make features overlap to each other. This is needed because our sound recording system is not synchronized with the robot arm operation. Since the starting points are not the same, the features should be extracted with overlapped parts to include possible patterns. The whole procedure of feature extraction is described in Figure 4.6.

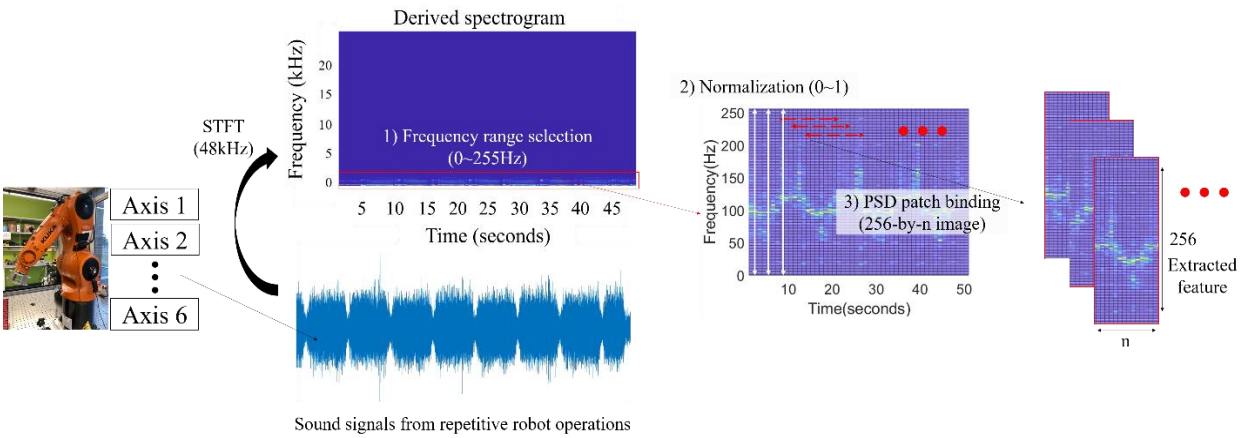


Figure 4.6 Features are extracted by 1) bandpass filtering, 2) normalizing, and 3) binding the sound spectrogram images.

To verify the noise rejection of our incorporated sound sensing system, Figure 4.7 compares processed signals (before PSD patch binding in Figure 4.6) from 6 axes, in both noiseless and noisy conditions. We noticed that there is little discrepancy between the external noise conditions.

Thus, data sets from both environments were fed into the autoencoders, then the results were compared for verification.

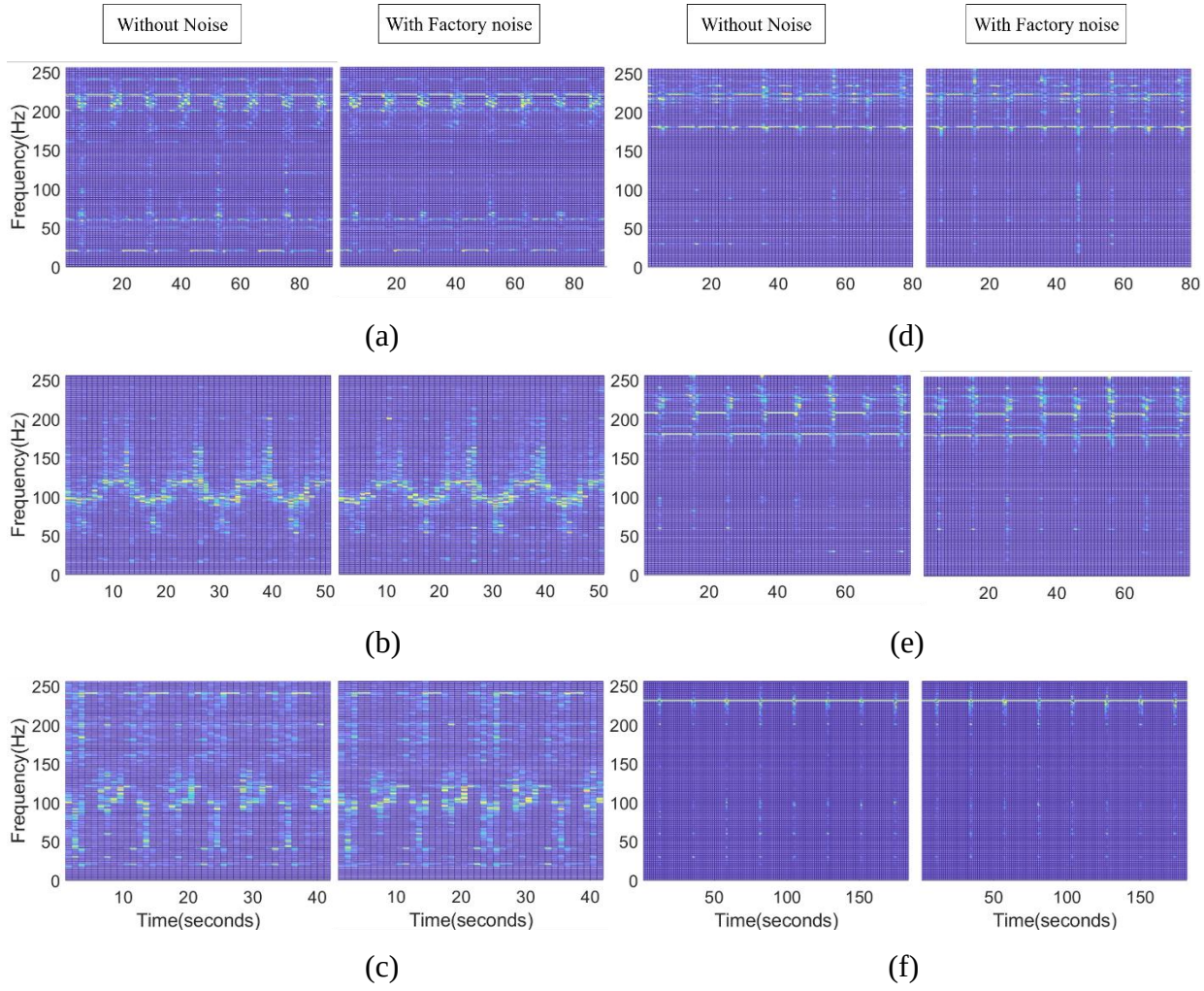


Figure 4.7 Sound spectrogram images both in the calm and the noisy environments are compared from the joint at: (a) axis 1, (b) axis 2, (c) axis 3, (d) axis 4, (e) axis 5, (f) axis 6.

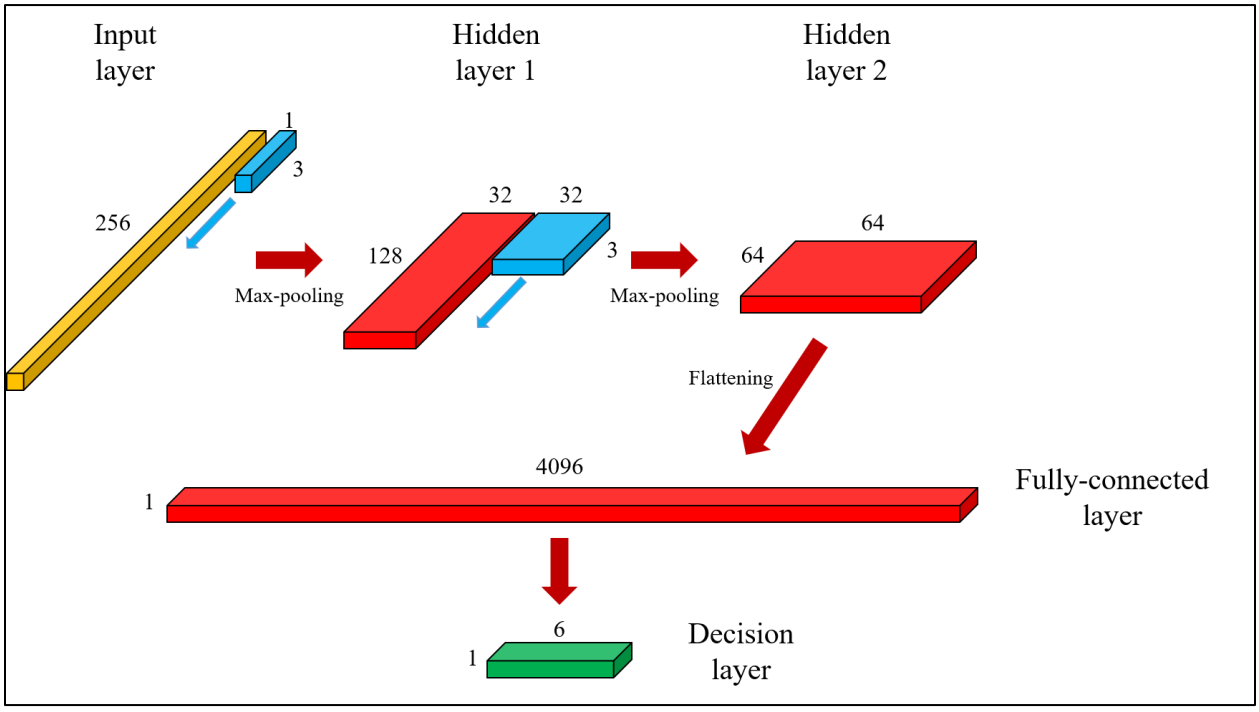
4.5 Preliminary study using the Convolutional Neural Network (CNN)

In Figure 4.7, several identical patterns of spectrograms can be recognized. However, it is difficult to perceive the trend of pattern change, since the baseline is not a physical model. Instead, the NN framework takes over in discerning the trends of spectral patterns. For this purpose, the CNN architecture was applied to verify whether the NNs may learn the features of the spectra.

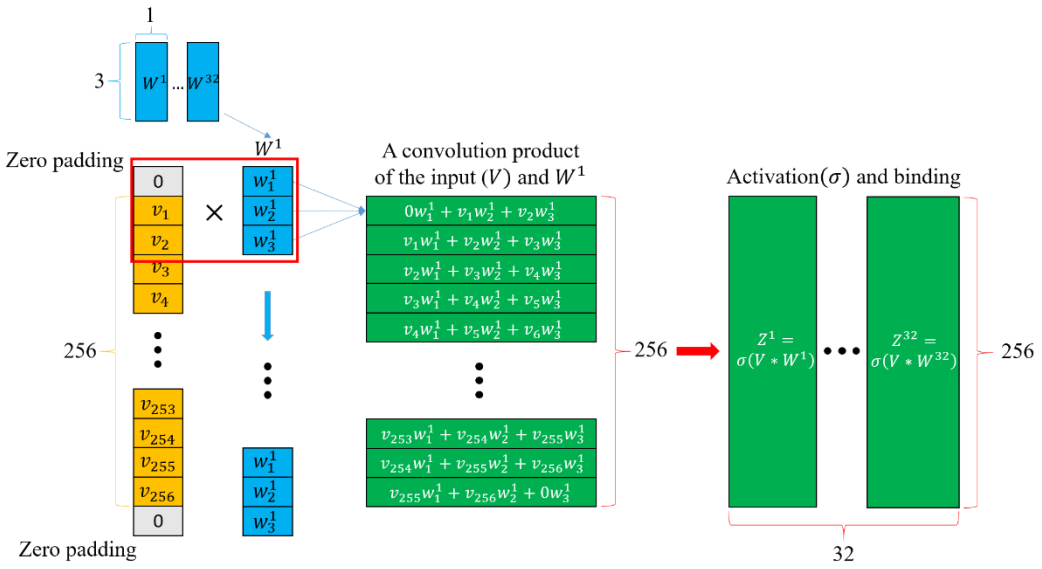
In this preliminary study, the CNN architecture was implemented to predict axis number by feeding forward the given input PSD, which is filtered up to 255Hz. The load condition for CNN application was restricted to 0lb. Based on the feature extraction process in Chapter 4.4, the 1-dimensional CNN with 2 hidden layers was designed, starting from input size of 256. The target value of CNN was designated as the axis number in operation (axis 1 - axis 6). The structural attributes of designed CNN are described in Table 4.2 and Figure 4.8.

Table 4.2 Hyperparameters of designed 2-hidden layer CNN; a hidden layer is composed of a convolution layer and a pooling layer. The max-pooling was applied in pooling layer.

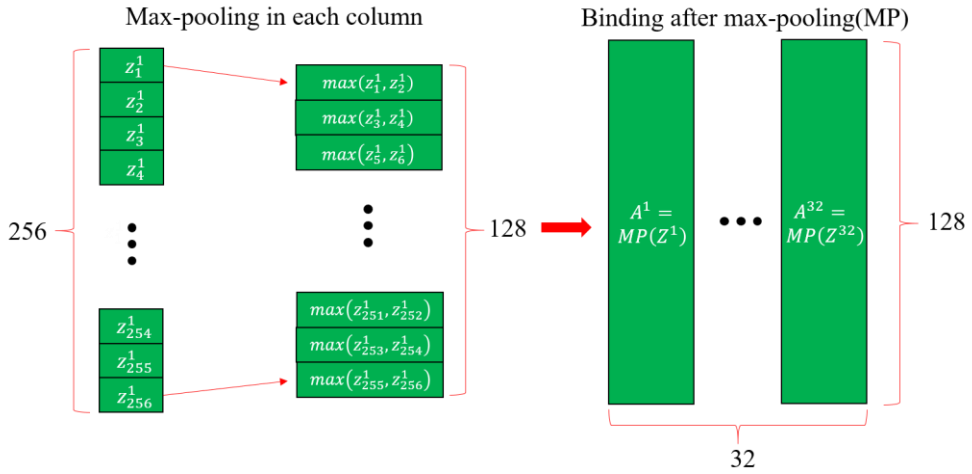
	Weight array size	Number of weight arrays	Stride size	Pooling size	Output size
Hidden layer 1	3 x 1	32	1 x 1	2 x 1	128 x 32
Hidden layer 2	3 x 32	64	1 x 1	2 x 1	64 x 64



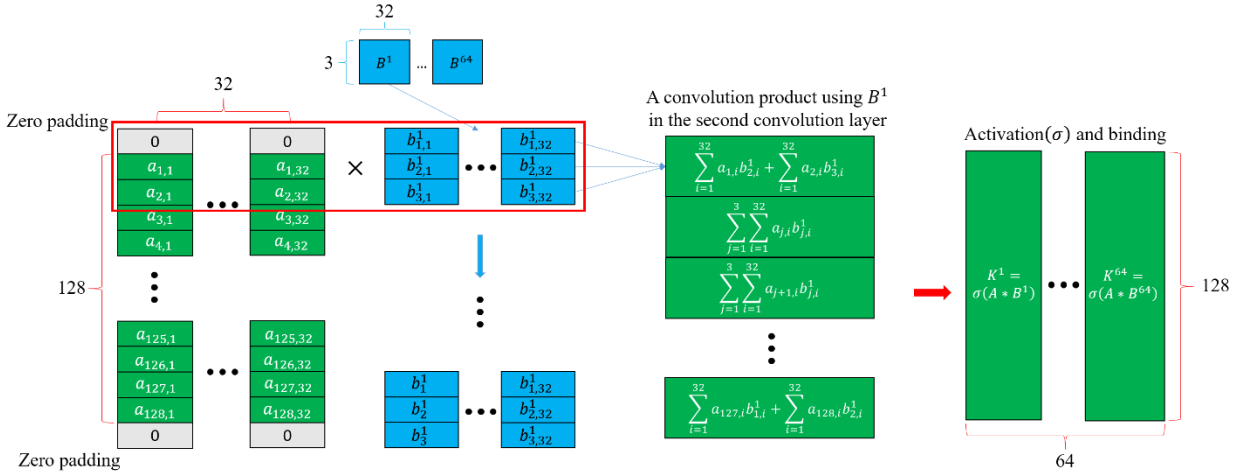
(a)


1
3
W^1 ... W^{32}
Zero padding
W^1
A convolution product of the input (V) and W^1
Activation(σ) and binding
256
$Z^1 = \sigma(V * W^1)$
$Z^{32} = \sigma(V * W^{32})$
32
Zero padding

(b)


Max-pooling in each column
Binding after max-pooling(MP)
256
128
$A^1 = MP(Z^1)$
$A^{32} = MP(Z^{32})$
32

(c)


32
3
B^1 ... B^{64}
Zero padding
A convolution product using B^1 in the second convolution layer
Activation(σ) and binding
128
$K^1 = \sigma(A * B^1)$
$K^{64} = \sigma(A * B^{64})$
64
Zero padding

(d)

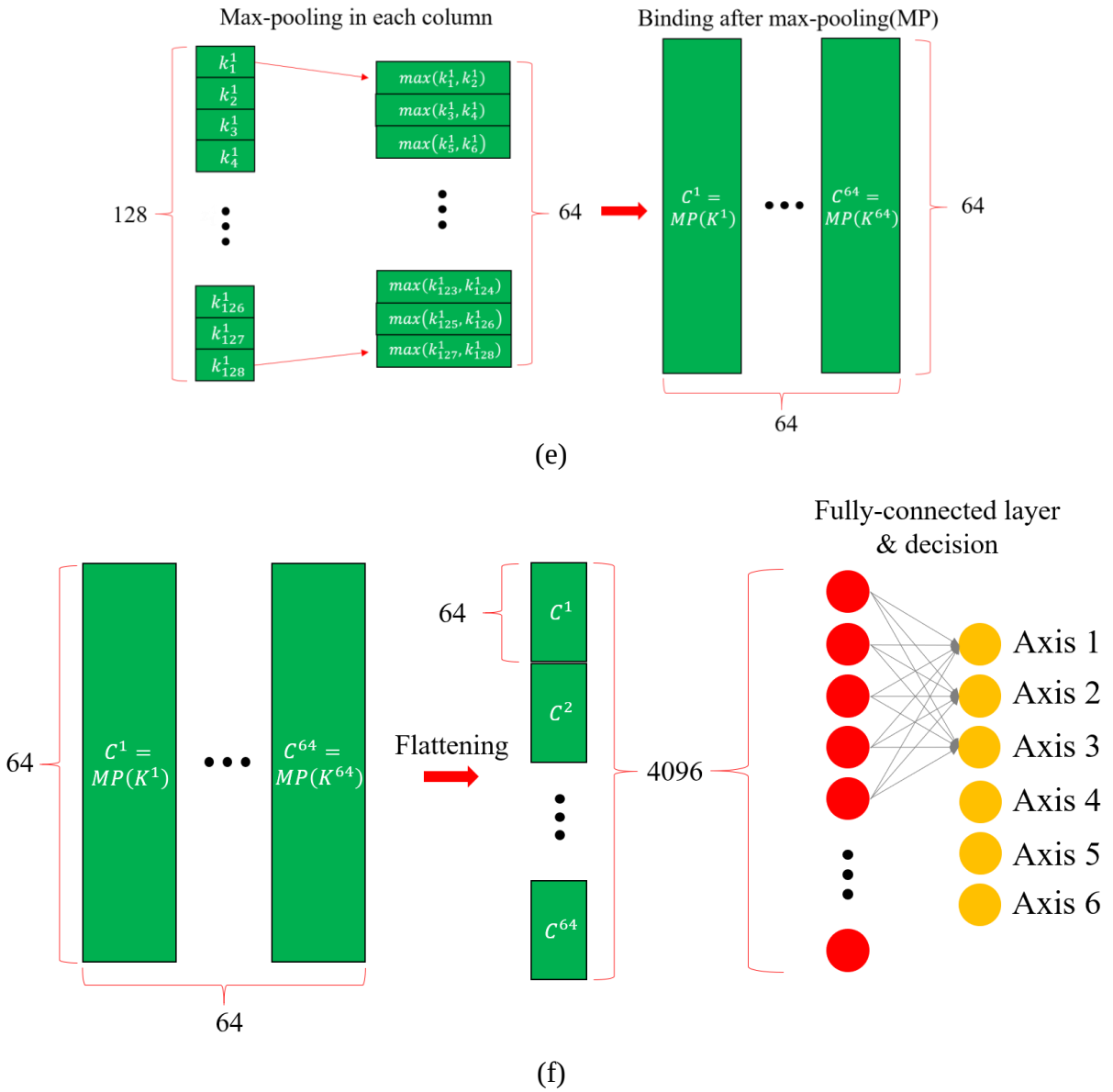


Figure 4.8 As a preliminary study, a 2-hidden-layer CNN is designed to take a 1-dimensional sound PSD vector (0Hz - 255Hz) as input to predict the axis number in operation. The entire structure of our design is depicted in (a). (b) and (c) illustrates the first convolution and pooling process, and (d) and (e) illustrates the second convolution and pooling process. (f) describes the fully-connected layer (FCL) for final decision (6 axes).

Figure 4.8-(a) portrays the entire CNN procedure. We included 2 hidden layers, and each hidden layer consists of a convolution layer and a pooling layer. Initially, an input vector passes through the first convolution layer. As shown in Figure 4.8-(b), two zeros are attached to each side of the input vector to maintain the original size of input. After zero-padding, a 3x1 weight vector W is

utilized to compute the convolution product vector, in size of 256x1. Since we have 32 different W, a 256x32 matrix is derived as result. Next, the 256x32 matrix is compressed into 128x32 in the first pooling layer, as depicted in Figure 4.8-(c). As we selected the max-pooling method, a larger value is selected in every neighboring 2 different cells. In the second convolution layer, the 3x32 weight matrices are used for convolution. Similar to the first convolution layer, 64 zeros are used for zero padding in the second convolution layer. As explained in Figure 4.8-(d), the pointwise multiplications of 2-dimensional matrices build up the convolution product in the second convolution layer. We have 64 different B, hence an output of the second convolution layer has the size of 128x64. As shown in Figures 4.8-(e) and (f), the output is compressed to 64x64 again in the second pooling layer, then flattened to 4096x1 vector in order to go through the fully-connected layer (FCL). In the FCL, the weight array performs the multiplication in fixed location as the general NNs or autoencoders do, instead of convolution or sweep. The FCL produces a 6x1 vector, which indicates the axis number. The target 6x1 vector is composed of single 1 and five 0, and the location of 1 stands for the axis number. For example, [0 0 0 1 0 0] and [0 1 0 0 0 0] indicate the axis 4 and axis 2, respectively. Then the errors between CNN output and target are computed for update process.

In the training session, the Adam optimizer [69] was employed with learning rate 0.0005, and the ReLU function was utilized as activation. Since the aim of this Chapter is not enhancing the performance, no further tuning process was carried out. The size of the training and testing data sets were 400 and 100, respectively. Figure 4.9 summarizes the testing results from both stethoscopes by trained CNNs, showing that the CNNs classify the joint number fairly, by learning from 1-dimensional spectrums. As a result, the signals from stethoscope 1 and 2 were matched to the joint number precisely with 92.03% and 91.7% accuracy respectively, using trained CNNs.

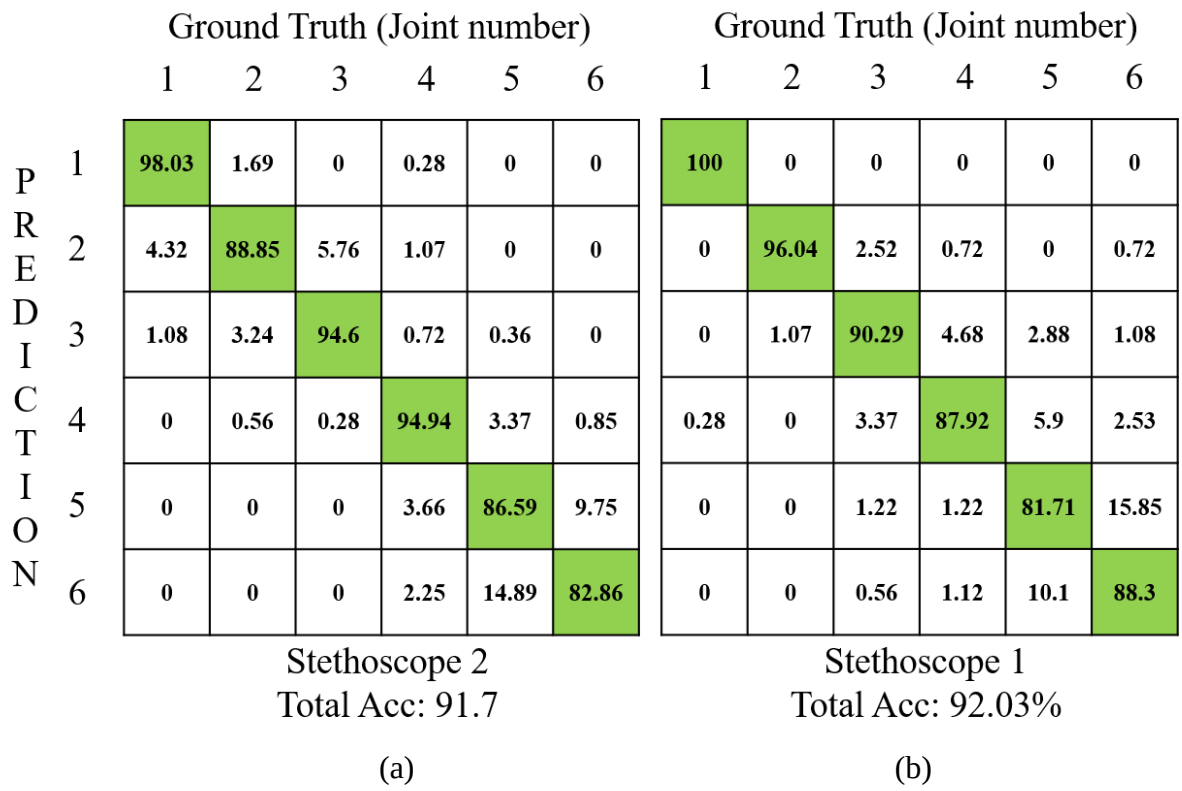


Figure 4.9 Joint number prediction results of a CNN using a stethoscope at: (a) wrist, and (b) base.

Even though the prediction results are not state-of-the-art, we notice that the NN is a feasible approach in the classifying problem on spectral patterns. Motivated by this preliminary study, our work continues to build the anomaly detection algorithm using another NN framework, the autoencoder.

4.6 Autoencoder design

In designing the structure of the autoencoder, the dimension of the input layer is determined by the time-length selection in the feature extraction process. The 2-dimensional features are stretched into 1-dimensional vectors then fed into the autoencoder. In the encoding phase, each dimension of the hidden layers is 1/4 of the dimension of the former layer. In contrast, in the decoding phase, the dimension of the latter layer is 4 times as large as the previous dimension. To avoid overfitting,

20% dropout rate was applied [70] between every layer. In the tuning process, the dimension of the first input (256 x n) and the depth of the autoencoder were controlled in search of the optimal design. The basic design of the autoencoder and controls on structural hyperparameters are described in Figure 4.10.

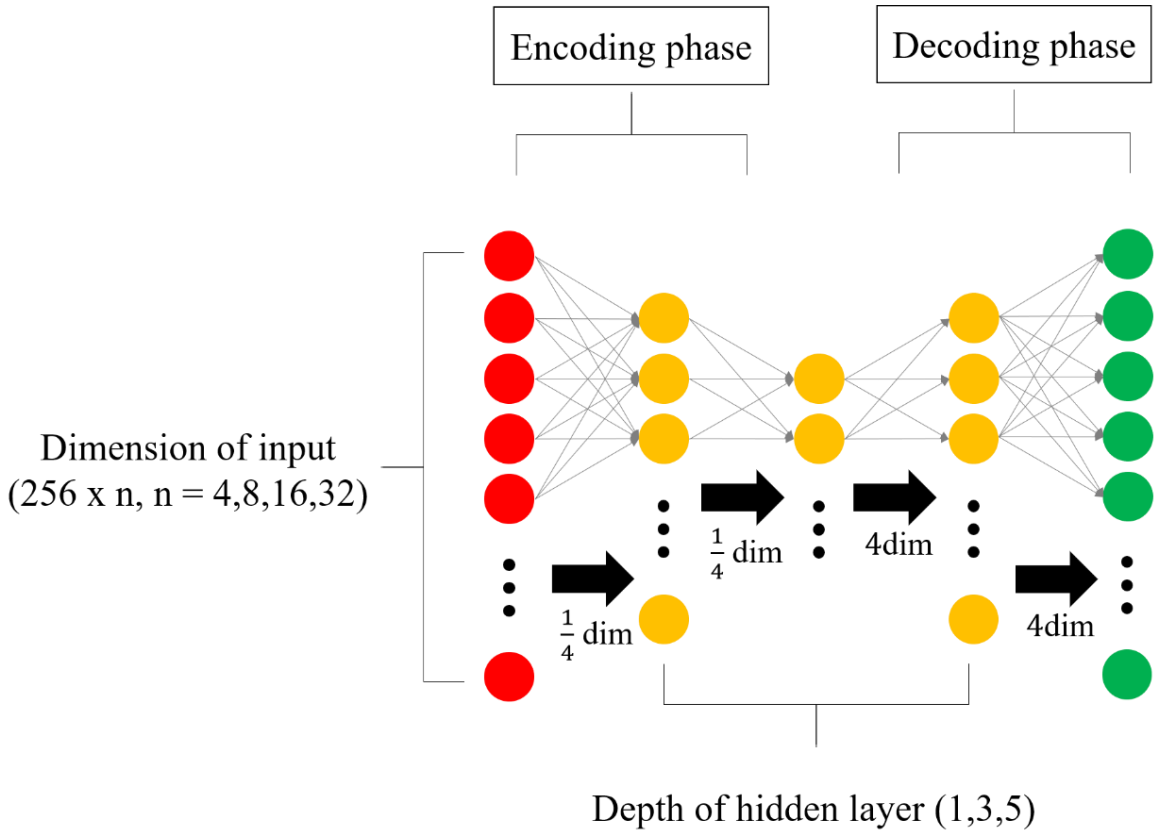


Figure 4.10 The structure of autoencoders is controlled by 4 different input dimensions (4, 8, 16, and 32) and 3 different hidden layer depths (1, 3, and 5).

To decide the NN structure for use among the given options, we established some standards as below:

1) Higher reconstruction performance: Smaller final losses of training and validation are better.

2) Avoiding overfitting: The loss difference between training and validation should be small. In other words, the ratio of the two losses closer to 1 is better.

3) Lighter computation load: If restrictions 1) and 2) are similarly satisfied among different structures, the simpler structure (lower depth and dimension) is better.

The selections of NN structure in each axis are detailed in chapter 3.2.6, by comparing the above standards after training.

4.7 Results

4.7.1 Data preparation

According to Figure 1.3, we separated the load conditions into two categories: normal (0lb - 2.5lb) and abnormal (5.0lb - 12.5lb). Since 5.0lb of load is close to the load capacity combined with the load of the connector, we classified the 5.0lb load as warning status. Firstly, data sets for training and validation were acquired for 12 days, only assigning normal loads (0lb - 2.5lb). Among them, the data for actual training were chosen from the first 10 days. Other data from the last two days were used for validation. After training autoencoders for each condition, data for testing were gathered for 10 days. Different from the first period of gathering, anomalous load conditions were also performed in the experiments. Figure 4.11 describes the overall data preparation process.

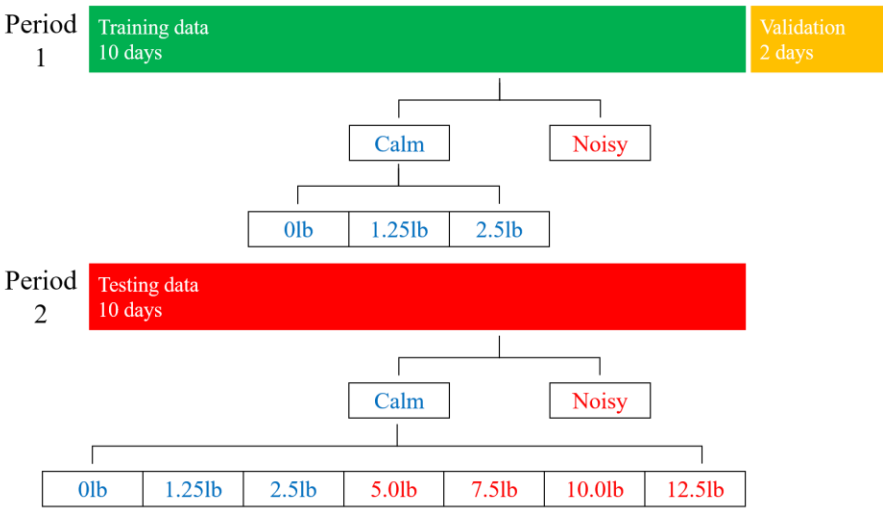


Figure 4.11 Training and testing data sets were prepared both in calm and noisy conditions, by applying various load conditions.

We trained autoencoders with different structural hyperparameters: the dimension of input, and the depth of the autoencoder. The most favorable structures were selected by measuring REs.

Figure 4.12 summarizes data separations and training / testing procedures of the autoencoder for each axis. The validation processes were included in the training session. Consequently, 6 differently trained autoencoders were created, with respect to 6 independent axes. After training the autoencoders, data sets collected in another period were used for testing.

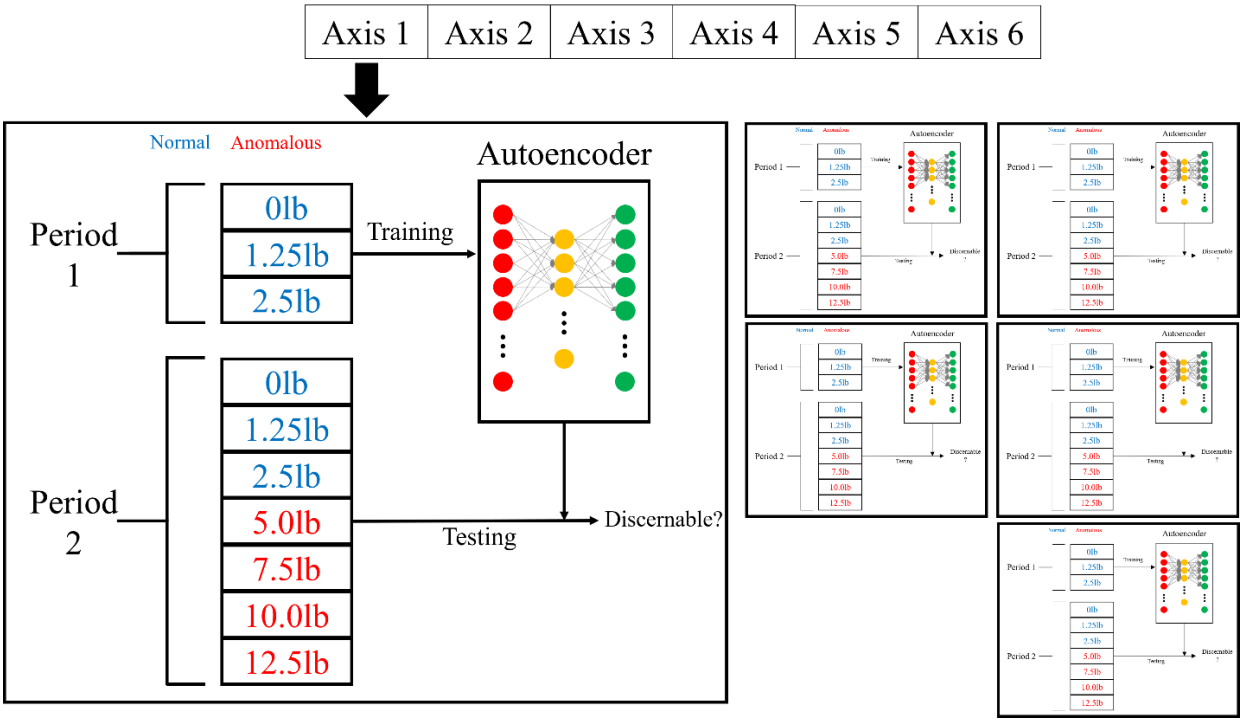


Figure 4.12 Gathered data sets were utilized for training and testing the autoencoder in each axis.

4.7.2 Training results

In the training session, we applied the Adam optimizer [69], setting the initial learning rate to 0.0005. Other hyperparameters and size of training and validation data sets are given in Table 4.3.

Table 4.3 Size of data sets for training and validation, and hyperparameters in training process; the number of features varies by the dimension of input (n), hence the size of training / validation data sets are represented by using n (=4,8,16,32).

	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Training size	$\mathbf{3083-n}$	$\mathbf{2119-n}$	$\mathbf{1575-n}$	$\mathbf{4272-n}$	$\mathbf{2801-n}$	$\mathbf{6779-n}$
Validation size	$\mathbf{633-n}$	$\mathbf{434-n}$	$\mathbf{314-n}$	$\mathbf{870-n}$	$\mathbf{582-n}$	$\mathbf{1393-n}$
Hyperparameters (shared)	Optimizer: Adam Initial Learning rate: 0.0005 Total epochs / batch size: 100 / 50					

With the hyperparameters given in Table 4.3, we scrutinized possible tuning options then selected the most favorable results according to the standards in Chapter 4. After choosing the structures, testing data sets were fed into the trained autoencoder to verify the separability of changes in load. The foundation results for selecting the structures are described in Tables 4.4 – 4.9.

Table 4.4 Comparison of training results in axis 1.

	Stethoscope #1 (wrist)			Stethoscope #2 (base)		
Dim \ Depth	1	3	5	1	3	5
4	Train loss 0.0043	Train loss 0.0037	***Train loss 0.0016***	Train loss 0.0065	Train loss 0.0052	Train loss 0.0055
	Val loss 0.0043	Val loss 0.0038	***Val loss 0.0015***	Val loss 0.0072	Val loss 0.0059	Val loss 0.0053
8	Train loss 0.0033	Train loss 0.0030	Train loss 0.0009	Train loss 0.0063	***Train loss 0.0045***	Train loss 0.0050
	Val loss 0.0035	Val loss 0.0053	Val loss 0.0015	Val loss 0.0063	***Val loss 0.0046***	Val loss 0.0048
16	Train loss 0.0011	Train loss 0.0020	Train loss 0.0008	Train loss 0.0050	Train loss 0.0042	Train loss 0.0047
	Val loss 0.0038	Val loss 0.0044	Val loss 0.0014	Val loss 0.0063	Val loss 0.0047	Val loss 0.0050
32	Train loss 0.0174	Train loss 0.0020	Train loss 0.0009	Train loss 0.0045	Train loss 0.0041	Train loss 0.0047
	Val loss 0.0162	Val loss 0.0045	Val loss 0.0013	Val loss 0.0062	Val loss 0.0047	Val loss 0.0050

Table 4.5 Comparison of training results in axis 2.

	Stethoscope #1 (wrist)			Stethoscope #2 (base)		
Dim \ Depth	1	3	5	1	3	5
4	Train loss 0.0082	Train loss 0.0046	Train loss 0.0042	Train loss 0.0065	Train loss 0.0043	***Train loss 0.0035***
	Val loss 0.0075	Val loss 0.0064	Val loss 0.0063	Val loss 0.0065	Val loss 0.0044	***Val loss 0.0038***
8	Train loss 0.0044	***Train loss 0.0017***	Train loss 0.0039	Train loss 0.0059	Train loss 0.0035	Train loss 0.0035
	Val loss 0.0046	***Val loss 0.0020***	Val loss 0.0032	Val loss 0.0065	Val loss 0.0038	Val loss 0.0037
16	Train loss 0.0041	Train loss 0.0012	Train loss 0.0031	Train loss 0.0037	Train loss 0.0034	Train loss 0.0009
	Val loss 0.0046	Val loss 0.0021	Val loss 0.0029	Val loss 0.0049	Val loss 0.0038	Val loss 0.0037
32	Train loss 0.0028	Train loss 0.0013	Train loss 0.0029	Train loss 0.0039	Train loss 0.0032	Train loss 0.0010
	Val loss 0.0044	Val loss 0.0019	Val loss 0.0029	Val loss 0.0046	Val loss 0.0039	Val loss 0.0038

Table 4.6 Comparison of training results in axis 3.

	Stethoscope #1 (wrist)			Stethoscope #2 (base)		
Dim \ Depth	1	3	5	1	3	5
4	Train loss 0.0072	Train loss 0.0068	Train loss 0.0046	Train loss 0.0121	Train loss 0.0081	Train loss 0.0062
	Val loss 0.0067	Val loss 0.0071	Val loss 0.0068	Val loss 0.0114	Val loss 0.0088	Val loss 0.0092
8	Train loss 0.0057	Train loss 0.0064	***Train loss 0.0047***	Train loss 0.0102	Train loss 0.0079	***Train loss 0.0085***
	Val loss 0.0067	Val loss 0.0064	***Val loss 0.0047***	Val loss 0.0102	Val loss 0.0091	***Val loss 0.0089***
16	Train loss 0.0053	Train loss 0.0047	Train loss 0.0045	Train loss 0.0074	Train loss 0.0055	Train loss 0.0085
	Val loss 0.0067	Val loss 0.0048	Val loss 0.0047	Val loss 0.0098	Val loss 0.0088	Val loss 0.0088
32	Train loss 0.0046	Train loss 0.0036	Train loss 0.0177	Train loss 0.0052	Train loss 0.0042	Train loss 0.0172
	Val loss 0.0066	Val loss 0.0050	Val loss 0.0154	Val loss 0.0099	Val loss 0.0089	Val loss 0.0160

Table 4.7 Comparison of training results in axis 4.

	Stethoscope #1 (wrist)			Stethoscope #2 (base)		
Dim \ Depth	1	3	5	1	3	5
4	Train loss 0.0023	Train loss 0.0017	Train loss 0.0005	***Train loss 0.0015***	Train loss 0.0008	Train loss 0.0011
	Val loss 0.0022	Val loss 0.0024	Val loss 0.0026	***Val loss 0.0018***	Val loss 0.0030	Val loss 0.0036
8	Train loss 0.0004	Train loss 0.0010	Train loss 0.0009	Train loss 0.0014	Train loss 0.0006	Train loss 0.0009
	Val loss 0.0023	Val loss 0.0011	Val loss 0.0011	Val loss 0.0022	Val loss 0.0030	Val loss 0.0035
16	Train loss 0.0007	Train loss 0.0008	***Train loss 0.0006***	Train loss 0.0014	Train loss 0.0005	Train loss 0.0032
	Val loss 0.0036	Val loss 0.0012	***Val loss 0.0005***	Val loss 0.0026	Val loss 0.0031	Val loss 0.0065
32	Train loss 0.0003	Train loss 0.0008	Train loss 0.0006	Train loss 0.0013	Train loss 0.0002	Train loss 0.0032
	Val loss 0.0029	Val loss 0.0011	Val loss 0.0005	Val loss 0.0024	Val loss 0.0029	Val loss 0.0065

Table 4.8 Comparison of training results in axis 5.

	Stethoscope #1 (wrist)			Stethoscope #2 (base)		
Dim \ Depth	1	3	5	1	3	5
4	Train loss 0.0006	**Train loss 0.0012**	Train loss 0.0020	Train loss 0.0037	***Train loss 0.0031***	Train loss 0.0038
	Val loss 0.0016	**Val loss 0.0014**	Val loss 0.0020	Val loss 0.0036	***Val loss 0.0033***	Val loss 0.0038
8	Train loss 0.0012	Train loss 0.0008	Train loss 0.0017	Train loss 0.0036	Train loss 0.0028	Train loss 0.0034
	Val loss 0.0013	Val loss 0.0013	Val loss 0.0014	Val loss 0.0036	Val loss 0.0034	Val loss 0.0035
16	Train loss 0.0009	Train loss 0.0007	Train loss 0.0074	Train loss 0.0034	Train loss 0.0027	Train loss 0.0075
	Val loss 0.0013	Val loss 0.0013	Val loss 0.0074	Val loss 0.0036	Val loss 0.0033	Val loss 0.0080
32	Train loss 0.0008	Train loss 0.0007	Train loss 0.0074	Train loss 0.0029	Train loss 0.0028	Train loss 0.0075
	Val loss 0.0012	Val loss 0.0014	Val loss 0.0074	Val loss 0.0037	Val loss 0.0034	Val loss 0.0080

Table 4.9 Comparison of training results in axis 6.

	Stethoscope #1 (wrist)			Stethoscope #2 (base)		
Dim \ Depth	1	3	5	1	3	5
4	Train loss 0.0122	Train loss 0.0111	Train loss 0.0059	Train loss 0.0053	***Train loss 0.0007***	Train loss 0.0018
	Val loss 0.0140	Val loss 0.0105	Val loss 0.0077	Val loss 0.0059	***Val loss 0.0010***	Val loss 0.0027
8	Train loss 0.0111	Train loss 0.0063	Train loss 0.0047	Train loss 0.0054	Train loss 0.0005	Train loss 0.0016
	Val loss 0.0113	Val loss 0.0103	Val loss 0.0050	Val loss 0.0055	Val loss 0.0011	Val loss 0.0027
16	Train loss 0.0063	Train loss 0.0043	Train loss 0.0036	Train loss 0.0047	Train loss 0.0005	Train loss 0.0044
	Val loss 0.0065	Val loss 0.0052	Val loss 0.0037	Val loss 0.0055	Val loss 0.0010	Val loss 0.0048
32	Train loss 0.0063	Train loss 0.0032	***Train loss 0.0013***	Train loss 0.0046	Train loss 0.0005	Train loss 0.0044
	Val loss 0.0052	Val loss 0.0040	***Val loss 0.0016***	Val loss 0.0056	Val loss 0.0010	Val loss 0.0048

In Tables 4.4 - 4.9, the final losses of various autoencoder structures in each stethoscope are given. The elected designs are highlighted with bold, underlined characters. After selecting proper structures, the distributions of REs from different noise conditions were investigated to set up thresholds for anomaly detection. As a first step, the maximum RE values were designated as thresholds in each experiment. Figures 4.13 and 4.14 compare the REs and designate the first thresholds, and Tables 4.10 and 4.11 depict the maximum RE values to setup thresholds. The thresholds designated from training results were utilized to detect anomalies in the testing data set, in order to separate load conditions over capacity from acceptable load conditions.

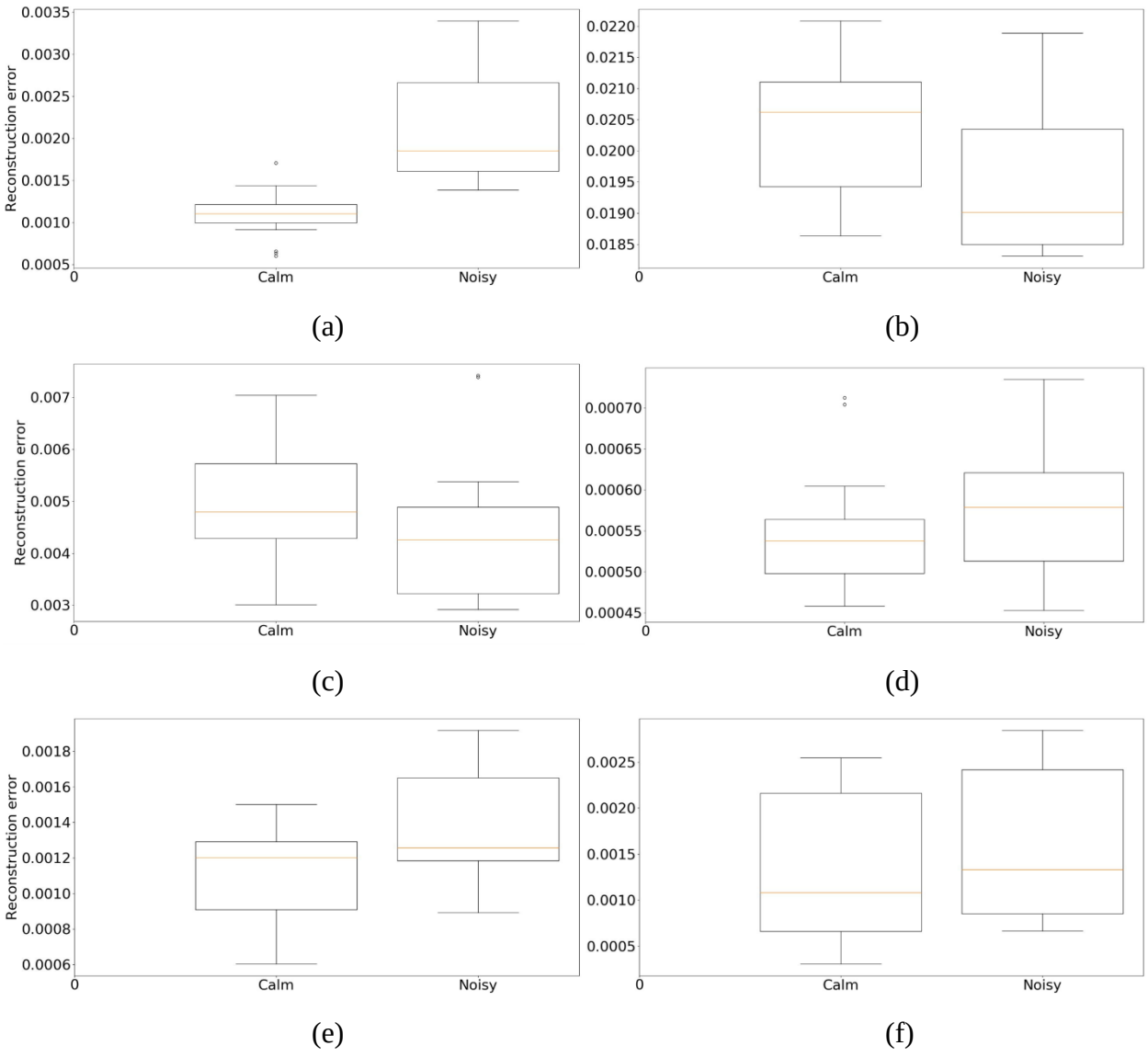


Figure 4.13 After training, REs are compared to set up the first threshold using mic #1, in: (a) axis1, (b) axis 2, (c) axis 3, (d) axis 4, (e) axis 5, (f) axis 6.

Table 4.10 First threshold values in each axis are settled from the maximum REs using mic #1.

	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Maximum RE	0.00339	0.0220	0.0074	0.000734	0.00192	0.00266
First threshold	0.0035	0.0225	0.0075	0.00075	0.002	0.00275

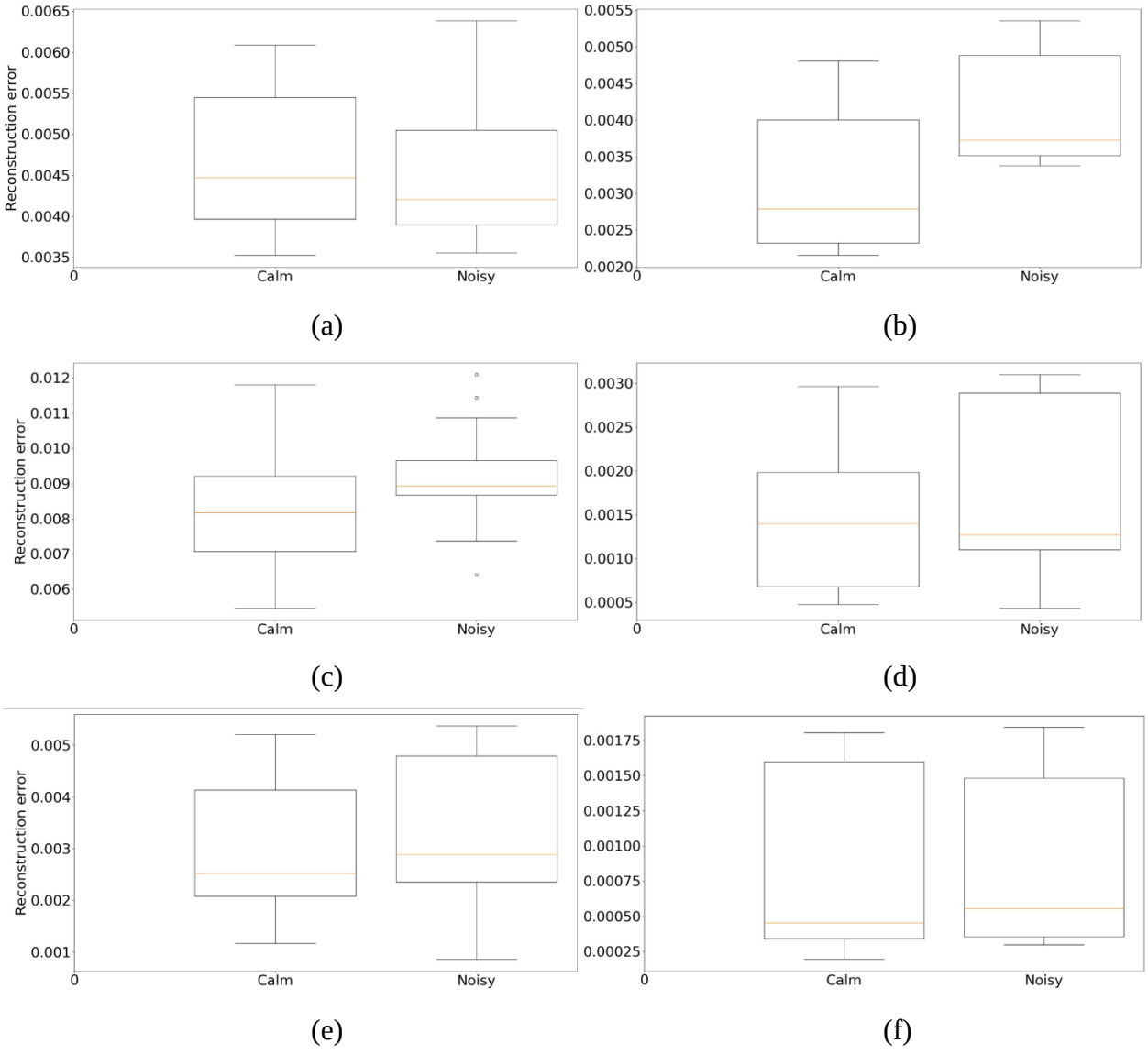


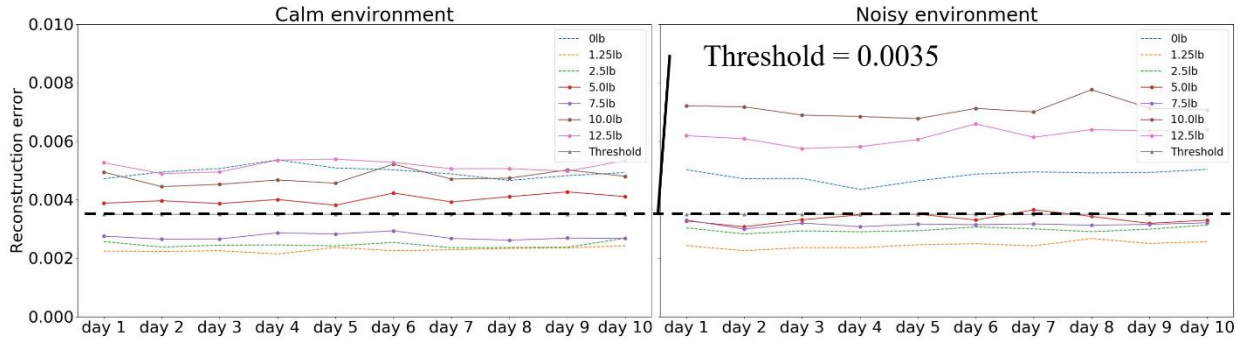
Figure 4.14 After training, REs are compared to set up the first threshold using mic #2, in: (a) axis1, (b) axis 2, (c) axis 3, (d) axis 4, (e) axis 5, (f) axis 6.

Table 4.11 First threshold values in each axis are settled from the maximum REs using mic #2.

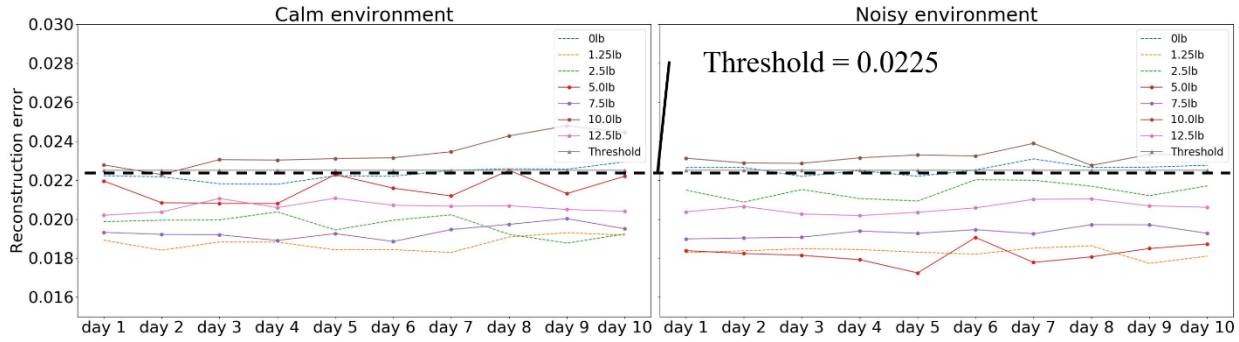
	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Maximum RE	0.0063	0.00535	0.0121	0.0031	0.00537	0.00184
First threshold	0.0065	0.0055	0.01225	0.0032	0.0055	0.0019

4.7.3 Testing results

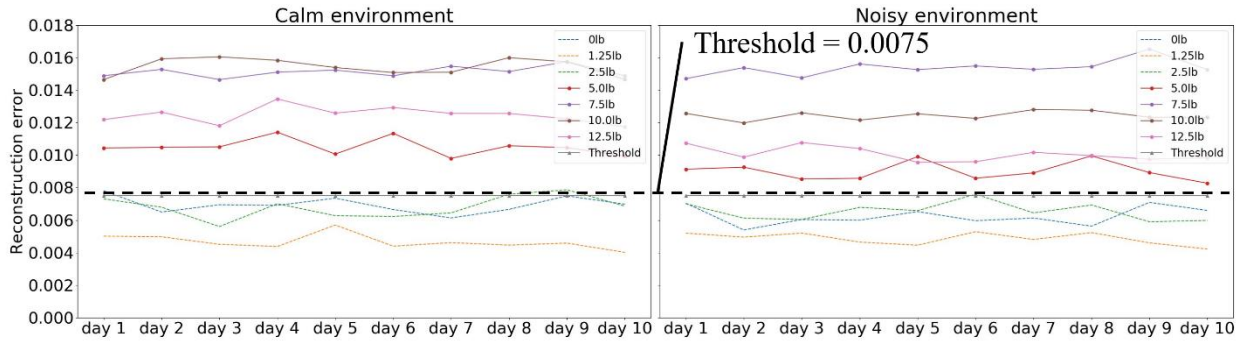
After training and tuning autoencoders, test data sets were collected for 10 days. In each daily experiment, 7 different load conditions were assigned. Similar to the training session, experiments were performed in both calm and noisy conditions. The processed signals were fed into trained autoencoders to compute REs in each experiment. As illustrated in Figures 4.15 and 4.16, daily REs were compared to show separability of anomalies with preset thresholds from training results. The success rates of detection are summarized in Tables 4.12 - 4.17.



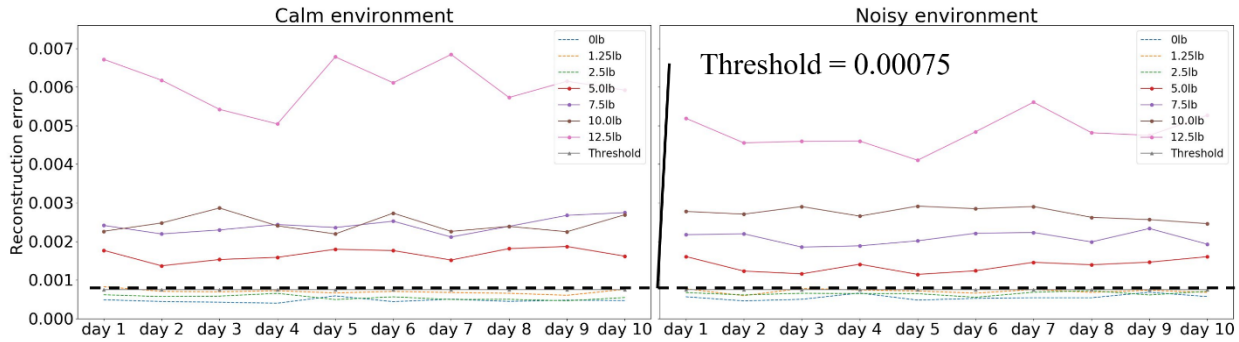
(a)



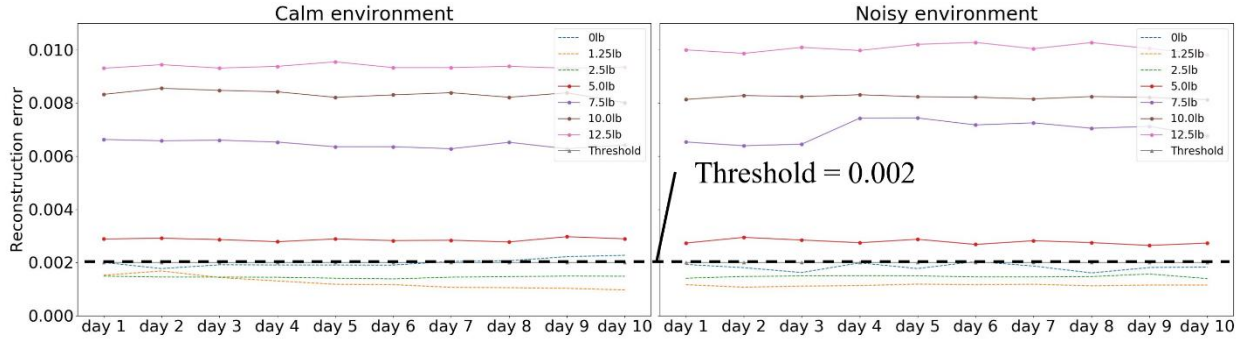
(b)



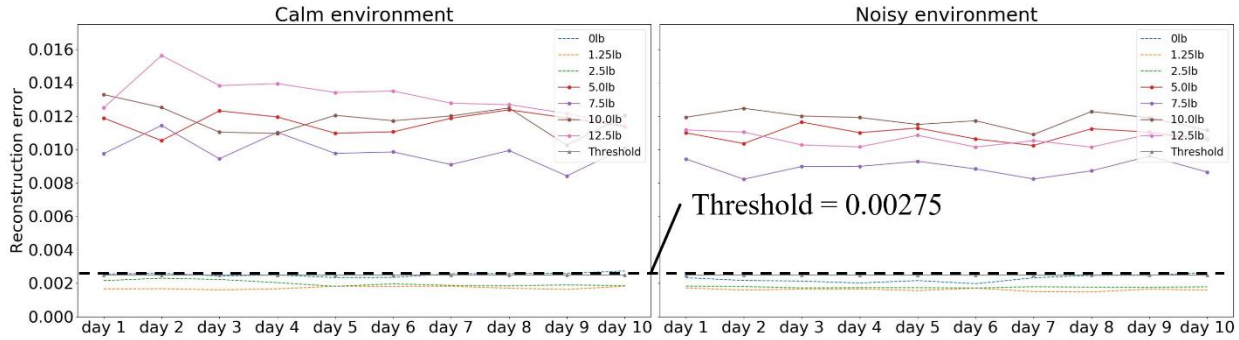
(c)



(d)

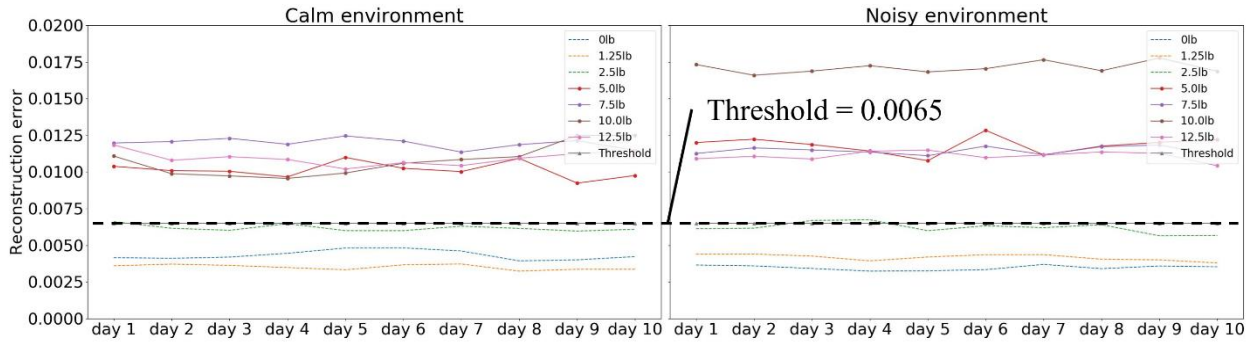


(e)

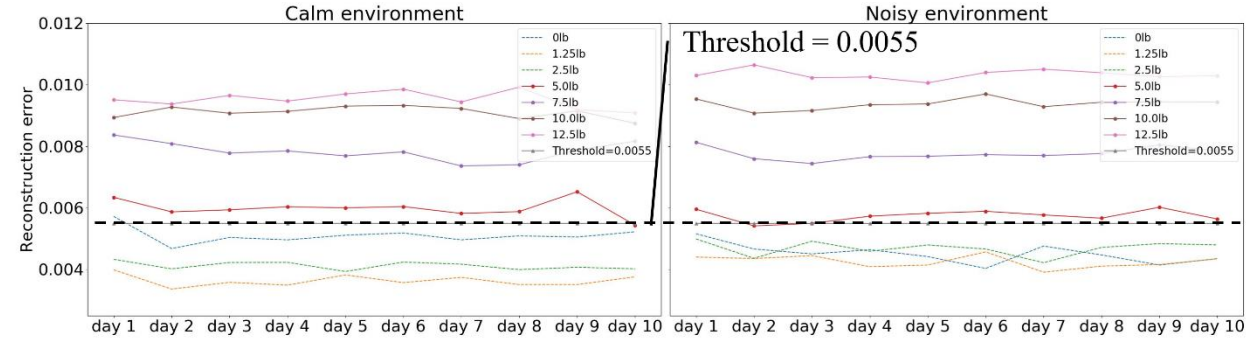


(f)

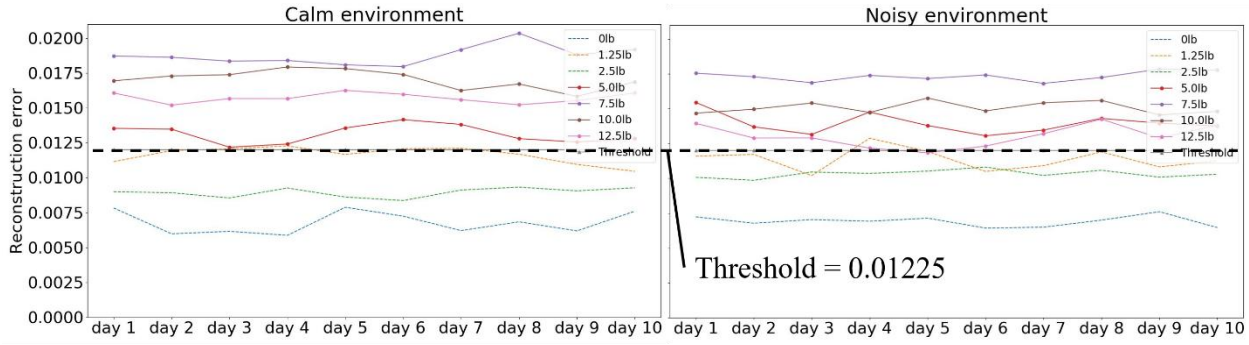
Figure 4.15 Thresholds from the training result are utilized to distinguish the normal and the abnormal status using mic #1, in: (a) axis 1, (b) axis 2, (c) axis 3, (d) axis 4, (e) axis 5, (f) axis 6; the anomalies can be separated in axes 3 - 6 using mic #1.


Calm environment
Noisy environment
Reconstruction error
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
day 1 day 2 day 3 day 4 day 5 day 6 day 7 day 8 day 9 day 10
0lb
1.25lb
2.5lb
5.0lb
7.5lb
10.0lb
12.5lb
Threshold
Threshold = 0.0065

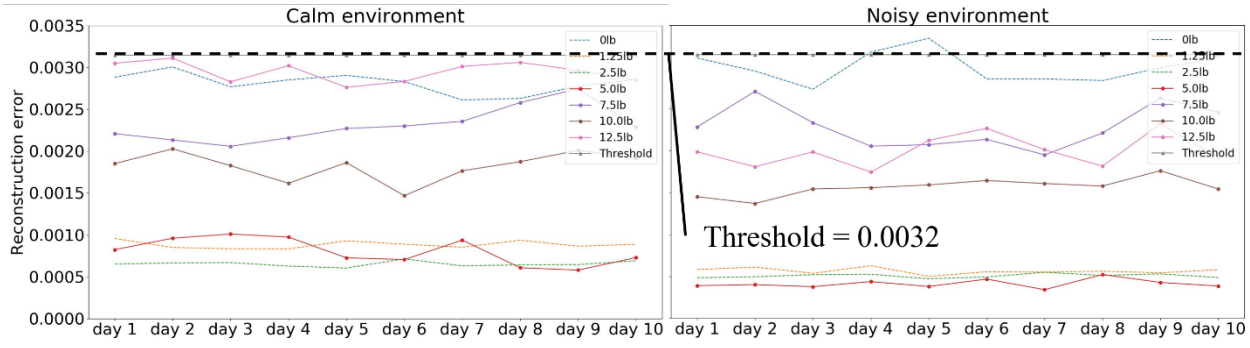
(a)


Calm environment
Noisy environment
Reconstruction error
0.012
0.010
0.008
0.006
0.004
day 1 day 2 day 3 day 4 day 5 day 6 day 7 day 8 day 9 day 10
0lb
1.25lb
2.5lb
5.0lb
7.5lb
10.0lb
12.5lb
Threshold=0.0055
Threshold = 0.0055

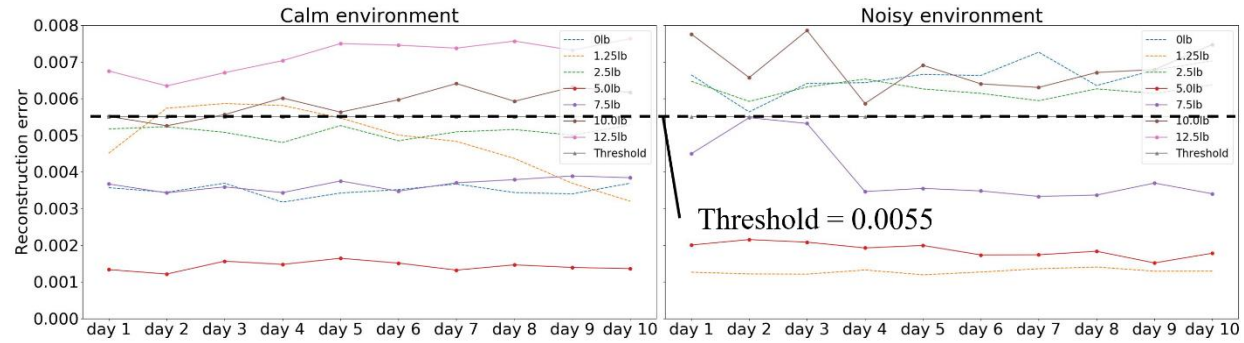
(b)


Calm environment
Noisy environment
Reconstruction error
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
day 1 day 2 day 3 day 4 day 5 day 6 day 7 day 8 day 9 day 10
0lb
1.25lb
2.5lb
5.0lb
7.5lb
10.0lb
12.5lb
Threshold
Threshold = 0.01225

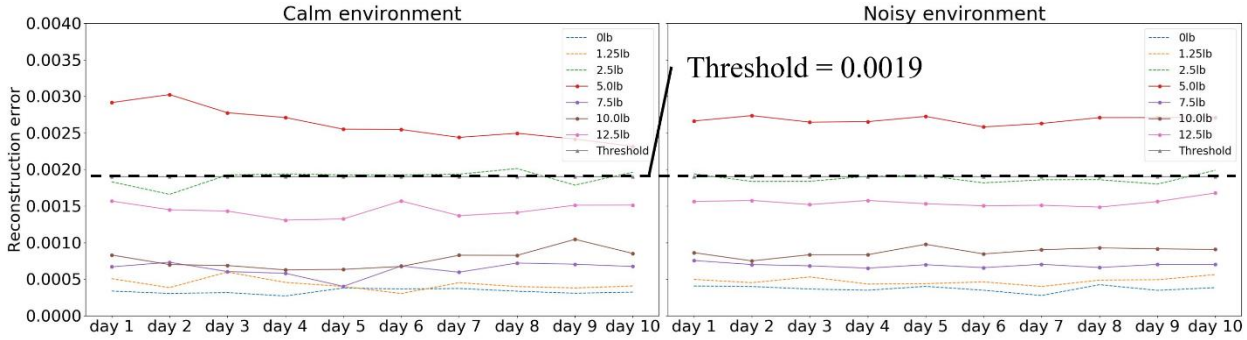
(c)


Calm environment
Noisy environment
Reconstruction error
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
day 1 day 2 day 3 day 4 day 5 day 6 day 7 day 8 day 9 day 10
0lb
1.25lb
2.5lb
5.0lb
7.5lb
10.0lb
12.5lb
Threshold
Threshold = 0.0032

(d)



(e)



(f)

Figure 4.16 Thresholds from the training result are utilized to distinguish the normal and the abnormal status using mic #2, in: (a) axis 1, (b) axis 2, (c) axis 3, (d) axis 4, (e) axis 5, (f) axis 6; the anomalies can be separated in axes 1 - 3 using mic #2.

Table 4.12 Detection results in testing (Axis 1);
Mic 1 fails to separate anomalous groups from normal groups (66%, 65%), while Mic 2 provides almost clear separation (93.3%, 100%).

	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
Mic #1	Normal	0/20	20/20	20/20	8/20	20/20	0/20	0/20	40/60
	Anomalous	20/20	0/20	0/20	12/20	0/20	20/20	20/20	52/80
Mic #2	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	20/20	20/20	16/20	0/20	0/20	0/20	0/20	56/60
	Anomalous	0/20	0/20	4/20	20/20	20/20	20/20	20/20	80/80

Table 4.13 Detection results in testing (Axis 2);
Mic 1 fails to separate anomalous groups from normal groups (86.7%, 75%), while Mic 2 provides almost clear separation (95%, 98.8%).

Mic #1	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	12/20	20/20	20/20	19/20	0/20	1/20	0/20	52/60
	Anomalous	8/20	0/20	0/20	1/20	20/20	19/20	20/20	60/80
Mic #2	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	19/20	20/20	18/20	1/20	0/20	0/20	0/20	57/20
	Anomalous	1/20	0/20	2/20	19/20	20/20	20/20	20/20	79/80

Table 4.14 Detection results in testing (Axis 3);
Mic 1 successfully separates anomalous groups from normal groups (91.7%, 100%), and Mic 2 also provides almost clear separation (91.7%, 98.8%).

Mic #1	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	18/20	20/20	17/20	0/20	0/20	0/20	0/20	55/60
	Anomalous	2/20	0/20	3/20	20/20	20/20	20/20	20/20	80/80
Mic #2	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	20/20	15/20	20/20	0/20	0/20	0/20	1/20	55/60
	Anomalous	0/20	5/20	0/20	20/20	20/20	20/20	19/20	79/80

Table 4.15 Detection results in testing (Axis 4);
Mic 1 successfully separates anomalous groups from normal groups (91.7%, 100%), while Mic 2 fails to provide separation (96.7%, 0%).

Mic #1	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	20/20	15/20	20/20	0/20	0/20	0/20	0/20	55/60
	Anomalous	0/20	5/20	0/20	20/20	20/20	20/20	20/20	80/80
Mic #2	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	18/20	20/20	20/20	20/20	20/20	20/20	20/20	58/60
	Anomalous	2/20	0/20	0/20	0/20	0/20	0/20	0/20	0/80

Table 4.16 Detection results in testing (Axis 5);
Mic 1 successfully separates anomalous groups from normal groups (90%, 100%), while Mic 2 fails to provide separation (61.7%, 48.8%).

Mic #1	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	14/20	20/20	20/20	0/20	0/20	0/20	0/20	54/60
	Anomalous	6/20	0/20	0/20	20/20	20/20	20/20	20/20	80/80
Mic #2	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	10/20	17/20	10/20	20/20	20/20	1/20	0/20	37/60
	Anomalous	10/20	3/20	10/20	0/20	0/20	19/20	20/20	39/80

Table 4.17 Detection results in testing (Axis 6);
Mic 1 successfully separates anomalous groups from normal groups (88.3%, 100%), while Mic 2 fails to provide separation (81.7%, 25%).

Mic #1	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	20/20	20/20	13/20	0/20	0/20	0/20	0/20	53/60
	Anomalous	0/20	0/20	7/20	20/20	20/20	20/20	20/20	80/80
Mic #2	load / decision	0lb	1.25lb	2.5lb	5.0lb	7.5lb	10.0lb	12.5lb	Success rate
	Normal	20/20	20/20	9/20	0/20	20/20	20/20	20/20	49/60
	Anomalous	0/20	0/20	11/20	20/20	0/20	0/20	0/20	20/80

4.8 Discussion

4.8.1 Feasibility of proposed method

From the testing results, we compared the success rates from different conditions. Different from supervised classification problem, the anomalous status should be discernable by setting thresholds without predefinition of the class. When the algorithm confuses the estimation of the normal signal, new thresholds should be provided to include outliers from normal status. For example, the threshold in Axis 2 - microphone #2 (base) can be modified to make a perfect discrimination. However, Axis 4 - microphone #2 (base) has no proper threshold for a perfect discrimination, because of some overlaps between normal and abnormal status. Hence microphone #2 fails to

provide a solution for anomaly detection in Axis 4. Based upon the foundations above, feasible sensing sources for the autoencoder are illustrated in Table 4.18.

Table 4.18 The feasibility of stethoscopes in each location is summarized; the stethoscope (#1) located at the wrist is applicable for monitoring axis 3 - axis 6, while the stethoscope (#2) at the base can be applied for monitoring axis 1 - axis 3.

	Stethoscope #1 (wrist)	Stethoscope #2 (base)
Axis 1	Not available	Available
Axis 2	Not available	Available
Axis 3	Available	Available
Axis 4	Available	Not available
Axis 5	Available	Not available
Axis 6	Available	Not available

4.8.2 Summary and suggestions

So far, we have explored ways to utilize the autoencoder using only "normal" signals for anomaly detection, instead of the supervised categorical learning. Sound signals were collected, transferred, then analyzed through stethoscope-USB microphone systems. It was shown that the features from low-frequency spectrogram (0Hz - 255Hz) were enough to build an algorithm in this application.

Although the proposed method succeeded in discerning different groups, however, it does not reflect the detailed harshness of condition. For example, in Axis 3 - microphone #1 (wrist), the increasing trend in the REs does not follow the order of load amount (REs of 7.5lb, 10lb are usually higher than REs of 12.5lb). Likewise, this application may have some difficulties when the users seek the prediction of exact load condition values.

Similarly, the algorithm may confuse when subtle status appears between normal and abnormal. As shown in Axis 5 - microphone #1 (wrist), the REs from the 5.0lb condition are closer to the normal group than the anomalous group. This implies that the 5.0lb load condition has not changed the patterns in the spectrogram sufficiently in some axes.

Besides, the importance of the mounting location of the sensor should be emphasized. In the initial stage of work, deciding where to attach the stethoscopes was a primary issue. As summarized in Table 4.18, each stethoscope has its available range of detection with respect to its location. Thus, it is required to implement multiple stethoscopes when applying this work to other works which have multiple sources of sound emission.

CHAPTER 5. CONCLUSION

5.1 Benefits and drawbacks

In this thesis, the autoencoder-based framework has been successfully implemented for anomaly detection of the industrial robot arm. In addition, it has been shown that the stethoscope is a viable sensing tool in factory areas as well as hospitals. However, there are some limitations in both the hardware and software aspects. In the contents below, the characteristics of the proposed method are summarized to help the achievements in future applications.

5.1.1 Stethoscope as a sensing tool

One of the prime attributes of the stethoscope is its focusing effect. In our equipment incorporation, USB microphones are attached at the end of the rubber tube of the stethoscopes. Because of this focusing attribute, external noise become negligible when the target is emitting sounds. We verified this point by comparing spectrograms acquired from calm and noisy circumstances.

As stated in Chapter 3.3, the low frequency response range is another special feature of the stethoscope. When the target application does not require a high frequency range of data, stethoscopes can provide easier and cost-effective solutions.

However, the stethoscope is restricted in measuring sound signals. When it comes to the user's need to construct a physical model and verify dynamic behaviors of the target, stethoscopes may fail to give the solution. Furthermore, the low frequency characteristic of stethoscopes can be not only a benefit, but also a limitation when broader range analysis is necessary.

5.1.2 Neural Network (NN) frameworks

The NN-based data analysis has an advantage that the users do not have to be experts with scientific background in each subject. As exhibited in our application, the strategy based upon NN enabled us to proceed toward given problems without establishing a structural model of the robot.

However, still we cannot verify the sources of pattern changes or anomalies in features solely by the NN approach. In addition, we cannot be certain that our designed NN models are the best solutions, since there is no theoretical way for finding the optimal tuning.

Nevertheless, NN-based data-driven method can be easily implemented in the initial stage of research in a variety of topics, to recognize the relationships between designated input and measured output. After this first step, designing the scientific models can be regarded as a follow-up study in the scheme of research.

5.2 Future work

5.2.1 Further development of the sound sensor

Although the stethoscope is successfully employed as a sound sensing tool for anomaly detection, it is limited to the low frequency response range. By maintaining the benefit of the stethoscope, we are planning to develop a new sound sensing device for broader bandwidth, which exploits the focusing effect by sealing. Our target is to expand the range to the audible frequency range (up to 20kHz).

5.2.2 Increasing the number of data sets by using data augmentation

There have been many discussions about the size of data sets for proper training of NNs. Besides extensive works on optimizing the size of data sets [43], [71], it is recommended as rule of thumb to set the training size to at least 10 times the number of updatable parameters.

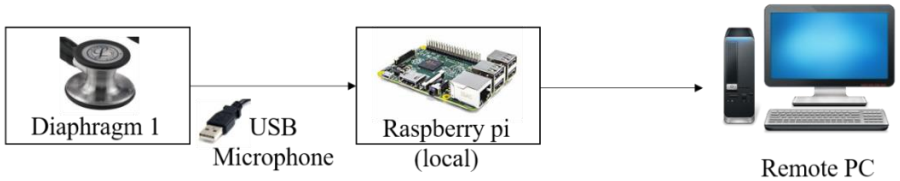
However, the number of weights in our work exceeds the above standard. Practically, it is difficult to collect the data from experiments to satisfy the recommendation since our designed NNs have more than millions of variables. Instead, the data can be generated by other direct methods and NN strategies. As introduced in Chapter 2, adding noise or direct modulations can be a simple way to reproduce real signals. The GAN has also been widely employed to resolve the lack of experiment data in manufacturing subjects.

5.3 Other ongoing applications

5.3.1 Remote health monitoring of hydraulic motor (with Standard Industrial)

In our laboratory, some work on remote factory access using Raspberry pi is in progress. As a part of the work, we attached a stethoscope on a hydraulic motor of the pipe bending machine located in the Standard Industrial facility, and are collecting sound signals daily. Our goal is to build a cloud server to store a long-term history of factories and combine this with data analytics techniques.

(a)

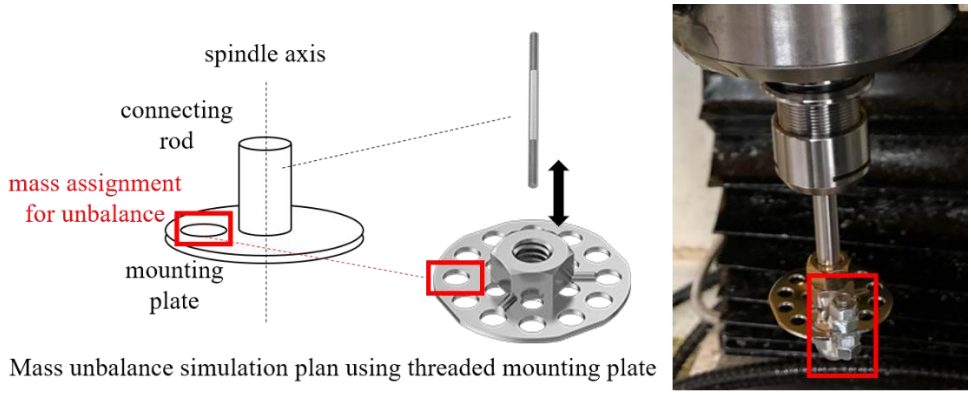


(b)

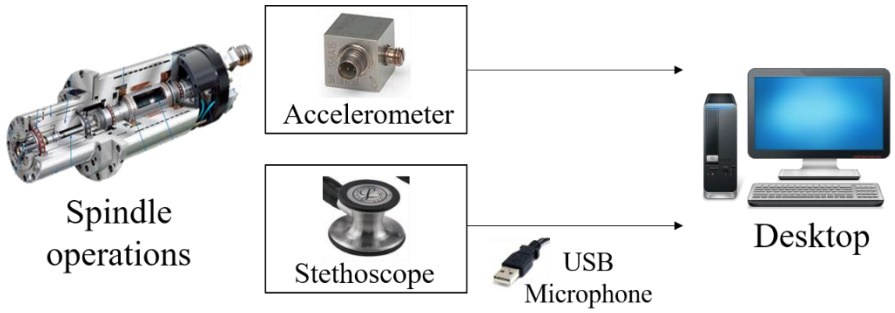
Figure 5.1 A hydraulic motor in the Standard Industrial is monitored by: (a) collecting sound signals using a stethoscope, and (b) delivering the signals to remote PC using Raspberry Pi.

5.3.2 Spindle unbalance detection (with Korea Institute of Machinery and Materials)

Applying a similar autoencoder-based framework, we are establishing a mass unbalance detection algorithm. As shown in Figure 5.1, we are gathering sound and vibration signals by changing the unbalance level. To simulate the mass unbalance, some nuts are mounted on the plate at the end of the spindle. This work is a collaborative project with the Korea Institute of Machinery and Materials.



(a)



(b)

Figure 5.2 A case study on mass unbalance detection of spindle is performed by: (a) assigning mass unbalance, and (b) capturing the sound and vibration signals using accelerometer and stethoscope.

REFERENCES

[1] A. A. Hayat, A. D. Udai, and S. K. Saha, "Kinematic-Chain of an Industrial Robot and Its Torque Identification for Gravity Compensation," in *Proceedings of the 1st International and 16th National Conference on Machines and Mechanics – iNaCoMM2013,* IIT Roorkee, India, pp. 1–8 , Dec. 2013.

[2] A. C. Bittencourt, K. Saarinen, S. Sander-Tavallaey, S. Gunnarsson, and M. Norrlöf, "A data-driven approach to diagnostics of repetitive processes in the distribution domain – Applications to gearbox diagnostics in industrial robots and rotating machines," *Mechatronics*, vol. 24, no. 8, pp. 1032–1041, Dec. 2014.

[3] P. D. McFadden and J. D. Smith, "Model for the vibration produced by a single point defect in a rolling element bearing," *J. Sound Vib.*, vol. 96, no. 1, pp. 69–82, Sep. 1984.

[4] M. Behzad, A. AlandiHallaj, A. Rohani Bastami, D. Mba, B. Eftekharnejad, and B. Charnley, "Defect size estimation in rolling element bearings using vibration time waveform," *Insight - Non-Destr. Test. Cond. Monit.*, vol. 51, no. 8, pp. 426–430, Aug. 2009.

[5] F. Cong, J. Chen, G. Dong, and M. Pecht, "Vibration model of rolling element bearings in a rotor-bearing system for fault diagnosis," *J. Sound Vib.*, vol. 332, no. 8, pp. 2081–2097, Apr. 2013.

[6] N. Tandon and A. Choudhury, "A review of vibration and acoustic measurement methods for the detection of defects in rolling element bearings," *Tribol. Int.*, vol. 32, no. 8, pp. 469–480, Aug. 1999.

[7] M. Elforjani and S. Shanbr, "Prognosis of Bearing Acoustic Emission Signals Using Supervised Machine Learning," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5864–5871, Jul. 2018.

[8] W. Zhou, T. G. Habetler, and R. G. Harley, "Stator Current-Based Bearing Fault Detection Techniques: A General Review," in *2007 IEEE International Symposium on Diagnostics for Electric Machines, Power Electronics and Drives* – Cracow, Poland, 2007, pp. 7–10, Sep, 2007.

[9] M. Blodt, P. Granjon, B. Raison, and G. Rostaing, "Models for Bearing Damage Detection in Induction Motors Using Stator Current Monitoring," *IEEE Trans. Ind. Electron.*, vol. 55, no. 4, pp. 1813–1822, Apr. 2008.

[10] P. D. McFadden and J. D. Smith, "The vibration produced by multiple point defects in a rolling element bearing," *J. Sound Vib.*, vol. 98, no. 2, pp. 263–273, Jan. 1985.

[11] A. C. Bittencourt and S. Gunnarsson, "Static Friction in a Robot Joint—Modeling and Identification of Load and Temperature Effects," *J. Dyn. Syst. Meas. Control*, vol. 134, no. 5, pp. 1–10, Jul. 2012.

[12] T. H. Loutas, D. Roulias, and G. Georgoulas, "Remaining Useful Life Estimation in Rolling Bearings Utilizing Data-Driven Probabilistic E-Support Vectors Regression," *IEEE Trans. Reliab.*, vol. 62, no. 4, pp. 821–832, Dec. 2013.

[13] M. S. Safizadeh and S. K. Latifi, "Using multi-sensor data fusion for vibration fault diagnosis of rolling element bearings by accelerometer and load cell," *Inf. Fusion*, vol. 18, pp. 1–8, Jul. 2014.

[14] F. Immovilli, M. Cocconcelli, A. Bellini, and R. Rubini, "Detection of Generalized-Roughness Bearing Fault by Spectral-Kurtosis Energy of Vibration or Current Signals," *IEEE Trans. Ind. Electron.*, vol. 56, no. 11, pp. 4710–4717, Nov. 2009.

[15] S. Khan and T. Yairi, “A review on the application of deep learning in system health management,” *Mech. Syst. Signal Process.*, vol. 107, pp. 241–265, Jul. 2018.
[16] H. R. Martin and F. Honarvar, “Application of statistical moments to bearing failure detection,” *Appl. Acoust.*, vol. 44, no. 1, pp. 67–77, April. 1995.
[17] P. Kadarno, Z. Taha, T. Dirgantara, and K. Mitsui, “Vibration Analysis of Defected Ball Bearing using,” in *Proceedings of the 9th Asia Pasific Industrial Engineering & Management Systems Conference* – Bali, Indonesia, pp. 2832–2840, Dec. 2008.
[18] H. Wang and P. Chen, “Fault Diagnosis Method Based on Kurtosis Wave and Information Divergence for Rolling Element Bearings,” *WESAS Transactions on Systems*, vol. 8, no. 10, pp. 1155–1165, Oct. 2009.
[19] Y. Lei, Z. He, Y. Zi, and X. Chen, “New clustering algorithm-based fault diagnosis using compensation distance evaluation technique,” *Mech. Syst. Signal Process.*, vol. 22, no. 2, pp. 419–435, Feb. 2008.
[20] D. Wang, P. W. Tse, and K. L. Tsui, “An enhanced Kurtogram method for fault diagnosis of rolling element bearings,” *Mech. Syst. Signal Process.*, vol. 35, no. 1–2, pp. 176–199, Feb. 2013.
[21] H. Qiu, J. Lee, J. Lin, and G. Yu, “Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics,” *J. Sound Vib.*, vol. 289, no. 4–5, pp. 1066–1090, Feb. 2006.
[22] J. Chebil, G. Noel, M. Mesbah, and M. Deriche, “Wavelet Decomposition for the Detection and Diagnosis of Faults in Rolling Element Bearings,” *Jordan Journal of Mechanical and Industrial Engineering*, vol. 3, no. 4, pp. 260–267, Dec. 2009.
[23] G. S. Chen and Q. Z. Zheng, “Online chatter detection of the end milling based on wavelet packet transform and support vector machine recursive feature elimination,” *Int. J. Adv. Manuf. Technol.*, vol. 95, no. 1–4, pp. 775–784, Mar. 2018.
[24] Z. K. Peng, P. W. Tse, and F. L. Chu, “A comparison study of improved Hilbert–Huang transform and wavelet transform: Application to fault diagnosis for rolling bearing,” *Mech. Syst. Signal Process.*, vol. 19, no. 5, pp. 974–988, Sep. 2005.
[25] V. K. Rai and A. R. Mohanty, “Bearing fault diagnosis using FFT of intrinsic mode functions in Hilbert–Huang transform,” *Mech. Syst. Signal Process.*, vol. 21, no. 6, pp. 2607–2615, Aug. 2007.
[26] J. A. Antonino-Daviu, M. Riera-Guasp, M. Pineda-Sanchez, and R. B. Perez, “A Critical Comparison Between DWT and Hilbert–Huang-Based Methods for the Diagnosis of Rotor Bar Failures in Induction Machines,” *IEEE Trans. Ind. Appl.*, vol. 45, no. 5, pp. 1794–1803, Oct. 2009.
[27] Muheng Wei, Maoyin Chen, D. Zhou, and W. Wang, “Remaining useful life prediction using a stochastic filtering model with multi-sensor information fusion,” in *2011 Prognostics and System Health Management Conference* – Shenzhen, China, 2011, pp. 1–6, May. 2011.
[28] T. Segreto, A. Simeone, and R. Teti, “Sensor Fusion for Tool State Classification in Nickel Superalloy High Performance Cutting,” *Procedia CIRP*, vol. 1, pp. 593–598, Dec. 2012.
[29] T. Segreto, A. Simeone, and R. Teti, “Multiple Sensor Monitoring in Nickel Alloy Turning for Tool Wear Assessment via Sensor Fusion,” *Procedia CIRP*, vol. 12, pp. 85–90, Sep. 2013.
[30] V. Balsamo, A. Caggiano, K. Jemielniak, J. Kossakowska, M. Nejman, and R. Teti, “Multi Sensor Signal Processing for Catastrophic Tool Failure Detection in Turning,” *Procedia CIRP*, vol. 41, pp. 939–944, Jan. 2016.

[31] K. M. Reichard, M. Van Dyke, and K. Maynard, “Application of sensor fusion and signal classification techniques in a distributed machinery condition monitoring system,” presented at the AeroSense 2000, Orlando, FL, 2000, pp. 329–336, Nov. 2000.
[32] E. T. Esfahani, S. Wang, and V. Sundararajan, “Multisensor Wireless System for Eccentricity and Bearing Fault Detection in Induction Motors,” *IEEEASME Trans. Mechatron.*, vol. 19, no. 3, pp. 818–826, Jun. 2014.
[33] M. Xia, T. Li, L. Xu, L. Liu, and C. W. de Silva, “Fault Diagnosis for Rotating Machinery Using Multiple Sensors and Convolutional Neural Networks,” *IEEEASME Trans. Mechatron.*, vol. 23, no. 1, pp. 101–110, Feb. 2018.
[34] L. Eren, “Bearing Fault Detection by One-Dimensional Convolutional Neural Networks,” *Math. Probl. Eng.*, vol. 2017, pp. 1–9, Jul. 2017.
[35] W. You, C.-Q. Shen, X.-J. Guo, and Z.-K. Zhu, “Bearing Fault Diagnosis Using Convolution Neural Network and Support Vector Regression,” *DEStech Trans. Eng. Technol. Res.*, no. icmeca, pp.5–11, Jul. 2017.
[36] W. Fuan, J. Hongkai, S. Haidong, D. Wenjing, and W. Shuaipeng, “An adaptive deep convolutional neural network for rolling bearing fault diagnosis,” *Meas. Sci. Technol.*, vol. 28, no. 9, pp. 1–17, Sep. 2017.
[37] Y. Wang, J. Xiang, R. Markert, and M. Liang, “Spectral kurtosis for fault detection, diagnosis and prognostics of rotating machines: A review with applications,” *Mech. Syst. Signal Process.*, vol. 66–67, pp. 679–698, Jan. 2016.
[38] A. A. Jaber and R. Bicker, “Fault diagnosis of industrial robot gears based on discrete wavelet transform and artificial neural network,” *Insight - Non-Destr. Test. Cond. Monit.*, vol. 58, no. 4, pp. 179–186, Apr. 2016.
[39] H. Liu, J. Zhou, Y. Xu, Y. Zheng, X. Peng, and W. Jiang, “Unsupervised fault diagnosis of rolling bearings using a deep neural network based on generative adversarial networks,” *Neurocomputing*, vol. 315, pp. 412–424, Nov. 2018.
[40] J. Gao, M. Kang, J. Tian, L. Wu, and M. Pecht, “Unsupervised Locality-Preserving Robust Latent Low-Rank Recovery-Based Subspace Clustering for Fault Diagnosis,” *IEEE Access*, vol. 6, pp. 52345–52354, Sep. 2018.
[41] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection,” in *International Conference on Machine Learning 2016 Anomaly Detection Workshop* - New York, USA, 2016, pp. 1–5, Jul. 2016.
[42] D. Oh and I. Yun, “Residual Error Based Anomaly Detection Using Auto-Encoder in SMD Machine Sound,” *Sensors*, vol. 18, no. 5, pp. 1-14, Apr. 2018.
[43] A. Alwosheel, S. van Cranenburgh, and C. G. Chorus, “Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis,” *J. Choice Model.*, vol. 28, pp. 167–182, Sep. 2018.
[44] X. Li, W. Zhang, Q. Ding, and J.-Q. Sun, “Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation,” *J. Intell. Manuf.*, vol. 1, pp.1–20, Nov. 2018.
[45] I. Goodfellow *et al.*, “Generative Adversarial Nets,” in *Neural Information Processing Systems 2014* – Montreal, Canada, pp. 1–9, Dec. 2014.

[46] H. Nishizaki, “Data augmentation and feature extraction using variational autoencoder for acoustic modeling,” in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* – Kuala Lumpur, Malaysia, 2017, pp. 1222–1227, Dec. 2017

[47] S. A. Khan, A. E. Prosvirin, and J.-M. Kim, “Towards bearing health prognosis using generative adversarial networks: Modeling bearing degradation,” in *2018 International Conference on Advancements in Computational Sciences (ICACS)* – Lahore, Pakistan, 2018, pp. 1–6, Feb. 2018.

[48] Y. O. Lee, J. Jo, and J. Hwang, “Application of deep neural network and generative adversarial network to industrial maintenance: A case study of induction motor fault detection,” in *2017 IEEE International Conference on Big Data (Big Data)* – Boston, MA, 2017, pp. 3248–3253, Dec. 2017.

[49] D. B. Verstraete, E. L. Droguett, V. Meruane, M. Modarres, and A. Ferrada, “Deep semi-supervised generative adversarial fault diagnostics of rolling element bearings,” *Struct. Health Monit.*, vol. 18, no. 4, pp. 1–22, May 2019.

[50] T. Szecsi, “Cutting force modeling using artificial neural networks,” *J. Mater. Process. Technol.*, vol. 92-93, pp. 344-349, Aug. 1999.

[51] P. G. Benardos and G. C. Vosniakos, “Prediction of surface roughness in CNC face milling using neural networks and Taguchi’s design of experiments,” *Robot. Comput.-Integr. Manuf.*, vol. 18, no. 5–6, pp. 343–354, Oct. 2002.

[52] A. Khorasani and M. R. S. Yazdi, “Development of a dynamic surface roughness monitoring system based on artificial neural networks (ANN) in milling operation,” *Int. J. Adv. Manuf. Technol.*, vol. 93, no. 1–4, pp. 141–151, Oct. 2017.

[53] J. Dennis, H. D. Tran, and H. Li, “Spectrogram Image Feature for Sound Event Classification in Mismatched Conditions,” *IEEE Signal Process. Lett.*, vol. 18, no. 2, pp. 130–133, Feb. 2011.

[54] V. Tra, S. A. Khan, and J.-M. Kim, “Diagnosis of bearing defects under variable speed conditions using energy distribution maps of acoustic emission spectra and convolutional neural networks,” *J. Acoust. Soc. Am.*, vol. 144, no. 4, pp. EL322–EL327, Oct. 2018.

[55] Q. Meng, D. Sen, S. Wang, and L. Hayes, “Impulse response measurement with sine sweeps and amplitude modulation schemes,” in *2008 2nd International Conference on Signal Processing and Communication Systems* – Gold Coast, Australia, 2008, pp. 1–5, Dec. 2008.

[56] A. Farina, “Simultaneous measurement of impulse response and distortion with a swept-sine technique,” In *Proceedings of the 108th AES Convention* – Paris, France, pp. 1–24, Feb. 2000.

[57] G. Turin, “An introduction to matched filters,” *IEEE Trans. Inf. Theory*, vol. 6, no. 3, pp. 311–329, Jun. 1960.

[58] R. L. Geyer, “The Vibroseis System of Seismic Mapping,” *Canadian Journal of Exploration Geophysics*, vol. 06, no. 1, pp. 39–57, Oct. 1970.

[59] D. F. Aldridge, “Mathematics of Linear Sweeps,” *Canadian Journal of Exploration Geophysics*, vol. 28, no. 1, pp. 62–68, Jun. 1992.

[60] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, Jul. 1958.

[61] R. Raina, A. Madhavan, and A. Y. Ng, “Large-scale deep unsupervised learning using graphics processors,” in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML ’09*, Montreal, Quebec, Canada, 2009, pp. 1–8, Jun. 2009.

[62] E. Hinton, “Learning Internal Representations by Error Propagation,” *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 1 pp. 318-362, Apr. 1986.
[63] M. Gori and A. Tesi, “On the problem of local minima in backpropagation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 1, pp. 76–86, Jan. 1992.
[64] T. Kavzoglu, “Determining Optimum Structure for Artificial Neural Networks,” in *25th Annual Technical Conference and Exhibition of the Remote Sensing Society* – Cardiff, UK, pp. 675–682, Sep. 1999.
[65] Y. Qi, Y. Wang, X. Zheng, and Z. Wu, “Robust feature learning by stacked autoencoder with maximum correntropy criterion,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* – Florence, Italy, 2014, pp. 6716–6720, May 2014.
[66] C. Zhou and R. C. Paffenroth, “Anomaly Detection with Robust Deep Autoencoders,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD ’17*, Halifax, NS, Canada, 2017, pp. 665–674, Aug. 2017.
[67] Z. Zhang, “Derivation of Backpropagation in Convolutional Neural Network (CNN),” pp. 1-7, Oct. 2016.
[68] S. Indolia, A. K. Goswami, S. P. Mishra, and P. Asopa, “Conceptual Understanding of Convolutional Neural Network – A Deep Learning Approach,” *Procedia Comput. Sci.*, vol. 132, pp. 679–688, May 2018.
[69] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *International Conference on Learning Representations* – San Diego, CA, USA, pp. 1–15, May. 2015.
[70] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, Jun. 2014.
[71] S. S. Du, Y. Wang, X. Zhai, S. Balakrishnan, R. Salakhutdinov, and A. Singh, “How Many Samples are Needed to Estimate a Convolutional Neural Network?,” in *32th Conference on Neural Information Processing Systems 2018*, Montreal, Canada, pp. 1–11, May 2018.