Kaunas University of Technology

Faculty of Informatics

Experimental Evaluation of A Relationship Between Digital Filter and Autoencoder Structures in Audio Signal-Based Anomaly Detection

Master's Final Degree Project

Yuki Tagawa

Project author

Prof. dr. Rytis Maskeliūnas

Supervisor

KAUNAS, 2021

Kaunas University of Technology

Faculty of Informatics

Experimental Evaluation of A Relationship between Digital Filter and Autoencoder Structures in Audio Signal-Based Anomaly Detection

Master's Final Degree Project

Informatics (code 6211BX00)

Yuki Tagawa

Project author

prof. dr. Rytis Maskeliūnas

Supervisor

Dr. R. Damaševičius

Reviewer

KAUNAS, 2021

Kaunas University of Technology

Faculty of Informatics

Yuki Tagawa

Experimental Evaluation of A Relationship between Digital Filter and Autoencoder Structures in Audio Signal-Based Anomaly Detection

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (after this – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project is plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Yuki Tagawa

Confirmed electronically

Yuki Tagawa, Experimental Evaluation of A Relationship between Digital Filter and Autoencoder Structures in Audio Signal-Based Anomaly Detection. Master's Final Degree Project, prof. dr., Rytis Maskeliūnas; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): IFMU-9.

Keywords: Anomaly Detection, Sound Data, Machine Learning, Autoencoder, Loss Function, Kalman Filter, Unscented Kalman Filter, Robust Estimation, Sparsity.

Kaunas, 2021. p.59.

Summary

Sound data-based anomaly detection is getting attraction. This is due to the recent development of a deep neural network applicable for representing complex-structured data, which has propelled its application in a real-world problem. However, the real-world sound data are usually contaminated with background noise, which hinders a model training process for anomaly detection. This report proposed applying various adaptive digital filters for the pre-processing of the sound data and studied the optimization of an autoencoder architecture to improve the anomaly detection performance when noisy data is applied. This study demonstrated the proposed approach with an open-source sound dataset of industrial machinery and discussed the relationship between the adaptive digital filters and the autoencoder architecture.

Yuki Tagawa, Ryšio Tarp Skaitmeninių Filtrų Ir Automatinio Kodavimo Įrenginių Tyrimas, Siekiant Nustatyti Garso Signalu Pagrįstą Anomaliją. Magistro baigiamasis projektas, prof. dr., Rytis Maskeliūnas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): IFMU-9.

Reikšminiai žodžiai: Anomalijų aptikimas, garso duomenys, mašininis mokymasis, automatinis koderis, nuostolių funkcija, „Kalman“ filtras, bekvapis „Kalman“ filtras, patikimas įvertinimas, retumas.

Kaunas, 2021. p.59,

Santrauka

Garso duomenimis pagrįstas anomalijų aptikimas tampa vis patrauklesnis. Taip yra dėl to, kad neseniai buvo sukurtas gilus neuroninis tinklas, pritaikytas vaizduoti sudėtingos struktūros duomenis, ir tai paskatino jį pritaikyti realaus pasaulio problemoje. Tačiau tikrojo pasaulio garso duomenys paprastai yra užteršti fono triukšmu, o tai trukdo modelio mokymui nustatyti anomalijas. Šioje ataskaitoje buvo pasiūlyta pritaikyti įvairius adaptyvius skaitmeninius filtrus išankstiniam garso duomenų apdorojimui ir ištirtas automatinio kodavimo įrenginio optimizavimas, siekiant pagerinti anomalijų aptikimo našumą, kai naudojami triukšmingi duomenys. Šis tyrimas pademonstravo siūlomą požiūrį naudojant pramoninių mašinų atviro kodo garso duomenų rinkinį ir aptarė adaptyvių skaitmeninių filtrų ir autokoderio architektūros ryšį.

Table of contents

List of figures

List of tables

1. INTRODUCTION

1.1. Relevance of the problem

Anomaly detection is a significant problem that has been researched within diverse research areas and application domains. Many anomaly detection techniques have been specifically developed for specific application domains, while others are more generic [1]. It has recently gotten further attention today's drastic improvement of machine learning techniques, thanks to the development of an efficient algorithm and computer performance in the last decade.

Anomaly detection is not just an academic research topic. The development in the Internet of Things causing the explosive growth of big data has encouraged anomaly detection in the actual business field. Anomaly detection is recognized as an essential technique in an application for preventive maintenance of the industrial machine. In the IoT era, industrial machinery are connected to a central control system, and this provides enormous and various diagonal data to the centre from the equipped sensors such as temperature, pressure, electric current, vibration, and sound. Among these various kinds of available data, sound data is easy to sample in a real factory due to its relatively low installation cost of microphones to existing facilities. This potential motivated various approaches for sound-based anomaly detection studies [2;3;4;5]. However, noise is known to exacerbate anomaly detection performance. In most business application scene, sound data is generally contaminated by environmental sound. Isolating the machine sound from the environmental sound is not an easy task. Hence, extracting meaningful sound from the contaminated sound relies on the performance of the anomaly detection device. Therefore, developing a noise-tolerant machine learning methodology is crucial for propelling sound data-based anomaly detection in a real factory.

Based on this demand from a business application, we launched the master thesis final project of "Experimental evaluation of a relationship between Digital Filter and Autoencoder Structures in Audio Signal-Based Anomaly Detection" ("**Project**"). In this Project, we challenge noisy sound data and aim to enhance its anomaly detection by understanding the theoretical backdrop of signal processing and machine learning.

1.2. Objectives of the Project

The Project's central objective is to improve the accuracy in classifying normal and anomalous conditions of industrial machine based on noisy sound data. The performance enhancement is evaluated using the Area Under the Curve ("**AUC**") value in anomaly detection for noisy data from baseline results published elsewhere.

1.3. The scientific novelty of this Project

In this Project, we proposed optimising the autoencoder architecture in line with the property of adaptive signal processing. To our knowledge, few studies are focusing on the relationship between data pre-processing and autoencoder architecture. Therefore, we demonstrated novel insights into anomaly detection.

1.4. Document Structure

The paper is structured as follows. Chapter 2 depicts the analysis, which consists of a literature survey related to this Project. Chapter 3 describes the projections, which gives the theoretical backdrops of

the learning algorithms of an autoencoder and the statistical signal processing including state estimation problems, accompanying our proposal for improvement. Chapter 4 is the central part of this report, which presents our experimental protocols and reports the numerical results with discussion. Chapter 5 wraps up the result and suggests future works.

1.5. Acknowledgements

We give warm thanks to prof. Chin-ya Huang at National Taiwan University of Science and Technology for stimulating discussions on adaptive filtering theories during the visit as an exchanging student. The funding support for the exchange programme from the Kaunas University of Technology committee is gratefully acknowledged.

2. ANALYSIS

2.1. Development of deep learning

Deep learning is a part of a family of machine learning. Deep learning method rooted in an artificial neural network model. The concept was to discover rich, hierarchical models over the kinds of data encountered in artificial intelligence applications. So far, the most striking successes in deep learning have involved discriminative models, usually those that map a high-dimensional input to a class label for pattern recognition [2].

The concept of neural network itself existed in the early days of the computer. Rosenblatt advocated the idea of "perceptron" in the 1950s. It developed a probabilistic model emulating the information process in a brain [6]. Later, in the 1980s, the neural network model for a mechanism of visual pattern recognition nicknamed "neocognitron" was posed to emulate a brain pattern recognition mechanism [7]. The model had a function of self-organization, which continues by employing 'learning without a teacher", which is today called "unsupervised learning."

However, such models did not get attention at the timing and faced the stagnation of research activities, so-called the AI winter. This stagnation was due to the expensive computation cost, which exceeded the performance of then-available computers. Improvement of representation performance requires to increase in the number of nodes. Such a neural network with multi nodes, generally greater than three nodes, requires expensive computation and yields problems such as vanishing gradient problem and trapping in local optimization. An overlearning is part of the local optimization problem. The overlearning happens if a model is fitted to a training data strictly and cannot evaluate novel data correctly, meaning to say, it loses generalization performance.

The recent development in both hardware and neural models has overcome the challenges of artificial intelligence as a thriving field with many practical applications and active research topics. Many studies proposed learning techniques to overcome the constraints mentioned above inherent in a deep neural network. A deep neural network had a difficulty in way of updating the parameters, because it requires the chain of deferential. The backpropagation solved this problem [8;9]. The optimization algorithms for updating bias and weight parameter in a backpropagation process are also essential in training a model. Many optimization algorithms were advocated and applied. In terms of vanishing of gradient, the introduction of a rectified linear function ("**ReLU**") as an activation function suppresses the problem [10;11]. Stochastic gradient descent ("**SDG**") algorithm samples randomly at every update. Momentum SDG is an algorithm modified from SDG by adding momentum terms to realize smooth updating [12]. AdaGrad algorithm can adjust the extent of updating automatically. For instance, these optimization algorithms had drawbacks; for instance, SDG updates all the parameters equally, therefore, cannot focus on meaningful parameters in networks. AdaGrad is known to tend to stop updating parameters while learning. Adaptive moment estimation ("**Adam**") was proposed to overcome these issues and realize that each parameter is updated as per the appropriate scale [13]. The details of these learning algorithms are discussed in section **3.1.2**.

The aforementioned concept of the neocognitron later became the basis for the convolutional neural network ("**CNN**") [14;15]. The CNN contributed to the meteoric rise of deep learning and was recognized as a breakthrough where the error rate for object recognition was halved and triggered the rapid adoption of deep learning by the computer vision community. The modern CNN is a version of multi-node perceptron architectures. And consists of convolution node, pooling nodes and fully

connected nodes. The convolution operation is a kind of linear operation. The operation is a prevalent computation technique used in image processing. At the convolution nodes, input information is processed by using much of the data's locality and converting it to several images with emphasizing characteristics in the image, which is called sparse interactions or sparse connectivity. Therefore, the node is sometimes called the detector stage. The pooling node is located just after the convolution nodes. The Pooling nodes consists of a pooling function. A pooling function divides an image into several regions and picks up representative information from each region, for instance, the maximum value in a max-pooling method, to deliver a new image. The pooling function provides a reduced size of an image, which is robust for a location change and contributes to reducing computation cost. Fully connected nodes are the same as a node used in conventional neural network architectures. CNN utilizes padding locates pixels with a particular value, typically zero, around an input image to maintain an image size same after convolution and capture characteristics at the edge of the image. Training of CNN model is done by backpropagation same as conventional neural network architectures. In contrast, pooling nodes have no parameter to be updated in n training.

Another epoch progress of deep neural network architecture is the generative adversarial network ("**GAN**") [16]. GAN is categorized as a generative model and a framework for estimating generative models via an adversarial process in which two models, a discriminator and a generator, are trained simultaneously. A generator generates counterfeit images based on an input noise, and a discriminator judges an input image to an original or the counterfeit one. The learning process in the original GAN framework is recognized as a min-max game where a generator and a discriminator are optimized with value function $V(D, G)$ formulated as,

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim Pdata(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim Pz(\boldsymbol{z})}\left[\log\left(1 - D\left(G(z)\right)\right)\right] \quad (1)$$

where $p_z(x)$ is the input noise variable, and the mapping to data space is represented as $G(z; \theta_g)$. $D(x)$ represents the probability that x came from the data rather than p_g. A GAN with built-in convolution nodes is called a deep convolutional generative adversarial network ("**DCGAN**") [17]. A CNN in a GAN and this combination is reported to generate images with higher resolution than a simple GAN.

The Recurrent Neural Network ("**RNN**") is a variation of a neural network that is intended for time-serial data [18]. RNN has intermediate nodes which have a loop structure from its output to its input. RNN is, as a nature of the recursive structure, good at possessing a short-term memory. Instead, it is not good at possessing long-term memory. A long short-term memory ("**LSTM**") was invented in order to conquer this drawback by adding a gate that judge discard past information or not and carry only necessary information [19]. The unique circuit is called an LSTM block. An LSTM block consists of a memory cell, input gates, output gates, and forget gates.

2.2. Anomaly detection with machine learning

Anomaly detection refers to finding patterns in data that do not conform to expected behaviour [1]. These atypical patterns are referred to as various terminology of anomalies, outliers and exceptions. No matter how it is called, the common principles measure the difference between normal and anomalous data in a numerical fashion. There have been many techniques, and models posed that should be selected, taking into account characteristics of data, the behaviour of anomalous data, and application purpose. We categorize anomaly detection techniques into traditional machine learning methods and deep learning methods to explain the existing literature.

The *classification-based* methods are generally supervised anomaly detection. In this approach, a model or classifier is trained from a labelled data instance, and the trained model classifies instance [3]. Both multi-class and one class anomaly detection techniques are available. Multi-class anomaly detection is a technique that assumes training data contains what belongs to multiple normal classes. The model has to learn to be a binary classifier. It distinguish between normal class against the rest of the classes, which can be considered as anomalous or outliers. If a test data is not classified as normal, then it is considered an outlier. This technique gives its prediction as a confidence score. Therefore, this technique is applicable for data whose normal classes are known.

The Support Vector Machine ("**SVM**") is the widely used binary classification-based methodology to discover novelties in an unsupervised way [20;21]. The SVM is a particular case of a support vector machine that learns a hyperplane to separate all the data points from the origin in a feature space corresponding to the kernel and maximizes the margin from this hyperplane the origin. The expectation is that anomalous test data will have the SVM fits for outlier detection. The model is first trained by normal condition data. The model learns to keep these training data away from the origin in the coordination. Thus, a hyperplane is established to separate normal condition area. With the trained model, test data of anomalous condition data is supposed to be plotted near the origin in the coordination. If the plotted data is inside of the constructed hyperplane, the data is detected as an anomaly.

The *distribution-based* method is to model the distribution of normal data. Methods differ in the features used to describe the data and the probabilistic model used to estimate the normal distribution. As the data spaces are high dimension, the distance cannot be measured in Euclid fashion, and therefore various measurement methodologies were proposed, such as Local Outlier Factor ("**LOF**") as a density-based method [22], and Nearest-Neighborhood as a distance-based method [23].

These classical approaches are already recognized as proven techniques. If input data is simple, these techniques are still the first choice for the application. However, complex data such as image recognition community and audio processing may exceed the modelling assumptions of these machine learning techniques.

The advent of deep learning techniques for anomaly detection has improved the results of traditional methods. Deep learning outperforms traditional machine learning as the scale of data increases [2]. The traditional machine learning approaches can be trapped in the curse of dimensionality. This restriction makes them inadequate tools for the analysis of high-dimensional data. Deep neural networks have been studied to overcome that arise in this context. One of the successful methods using deep learnings is a reconstruction-based method [4;5;24]. The fundamental idea behind the methods is that the normal condition can be reconstructed accurately from a hidden layer with smaller units than the input node. In contrast, the anomalous condition cannot be reconstructed, that is to say, embracing more considerable reconstruction losses. This fashion is suitable for anomaly detection, where anomalous condition data volume is generally far smaller than normal condition data because a model for detection can be trained only by normal condition data.

In the reconstruction context, GAN is also applied for anomaly detection ("**AnoGAN**") [25]. The central idea is that a generator trained with normal data poses high reconstruction loss when generating an anomalous image.

Deep learning can be used as a feature extractor. Deep one-class ("**DOC**") is an approach inspired by kernel-based one-class classification and minimum volume estimation and to trains a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data [26]. Minimizing the volume of the hypersphere forces the network to extract the common factors of variation, and anomaly can be detected if the test instance is plotted out of the boundary of the hypersphere. For sound-based anomaly detection, MARCHI proposed the LSTM approach [27]. It is unsupervised approach based on a denoising autoencoder with bidirectional LSTM.

Anomaly detection using deep learning-based methods emerged recently but already shown promising performances, especially for big and complicated data. We think these approaches are applicable for anomaly detection with audio data, as our concern is to measure the difference between normal and anomalous.

2.3. Adaptive digital filters for denoising

Getting meaningful information from noisy data is a traditional subject in the field of signal processing. The filtering theory, the solution to the problem, was proposed by Kolmogorov and Wiener for stationary time series data during the late 1930s and early 1940s [28]. Kolmogorov developed a comprehensive treatment of the linear prediction problem for discrete-time stochastic processes. Independently, Wiener formulated the continuous-time linear prediction problem and derived an explicit formula, known as the Wiener-Hopf equation, for the optimum predictor. Wiener also considered the filtering problem of estimating a process corrupted by additive noise.

Both Kolmogorov and Wiener assumed an infinite amount of data and assumed the stochastic process to be stationary. Despite the successful applications of their formulations in line with generalization by various researchers, the difficulties of updating with increases in the observation interval and modifying the formulations for vector cases. In the early age of space exploration, these problems were fundamental challenges for determining satellite orbits.

In order to overcome the challenges aforementioned, Kalman introduced state-space representation into the filtering problem and proposed a Kalman filter ("**KF**") [29].The idea is to express a dynamic system in a particular form called the state-space representation. The KF is an algorithm for sequentially updating a linear projection for the system. A system is expressed by using a system equation and measurement equation. The essential assumptions in formulating the KF are that the time series is linear, and the noise has the property of normality. Using these assumptions, KF can describe the dynamics of time series with the first-order moment and the second-order moment of the normal distributions. The theoretical detail and the algorithm of KF are discussed in section **3.2.3**.

In contrast to a linear system, normality is not maintained by a nonlinear transformation in a nonlinear system, making the state estimation problem difficult. The central problem in the nonlinear Kalman filter is how to estimate state space from the nonlinearly transformed distribution. There have been proposed various approximation approaches for the solution. The main approximation methods are linearization and statistical sampling. The linearization method is acquiring linearized approximation of the nonlinear function using Taylor expression and applying the traditional linear Kalman filter. The widely used filtering strategy based on this idea is the Extended Kalman filter ("**EKF**") [30].

The deterministic sampling approach is the way to approximate a probability distribution instead of approximating the nonlinear function. This approach is more computationally expensive than EKF.

The breakthrough was the unscented Kalman filter ("**UKF**") by Julier and Uhlmann [31]. The UKF selects the finite number of sigma points to represent the nonlinearly transformed distributes, and accordingly, it can be computed to the same extent of complexity as the EKF. The theoretical detail and the algorithm of UKF are discussed in section **3.2.4**.

2.4. Sparsity and robust estimation

From a practical perspective, the environmental presence of disturbances is unavoidable. In the sound-based anomaly detection problem, noise is present in sound data used for training and testing. For a system to be robust, small disturbances should only result in small estimation errors. How to estimate meaningful information can be broken down to two folds. The one utilizing the sparsity and another applies robust estimation.

The application of the Lasso norm to separate signal and noise for sparse estimation was originated in Donoho and Stark's work [32]. They proved the uncertainty principles and said signal and its Fourier transform could not both be highly concentrated. In the 1990's, such sparsity induced representation was systematized and various application, including neuro-science field was studied [33].

As we discuss later in section **3.1.5**, the autoencoder with appropriate architecture and activate functions can learn sparse representation. There are various variants of the autoencoder proposed for reinforcing its sparse representation. Sparse Auto-Encoders are given by introducing the Kullback-Leibler divergence as the regularization term [34;35]. The regularization term contributes to suppress the mean activity of the hidden units [36]. Vincent proposed the De-noising AutoEncoder ("**DAE**") [37]. The DAE aims at minimizing the reconstruction error between a sample and the reconstructed vector using its corrupted version. Rifai proposed the Contractive Auto-Encoder ("**CAE**") [38]. The CAE incorporates the squared Frobenius norm of the Jacobian matrix as a regularization term. This term aims at minimizing the sensitivity of the hidden representation to slight changes in input.

If we recognize noise as outlier, the robust estimation problem can be applied. Many methods of robust parameter estimation have been proposed to reduce the bias induced by outlier presence [39]. There is various substitution of the ordinary least square for robust estimation. The mean absolute error can be used for the purpose, since the estimation represents a sample median. Maximum likelihood estimation is a typical form of parameter estimation, but it is also well known that it is not robust against outliers. To develop robust estimation, the modifications of divergence functions were proposed. Fujisawa and Eguchi discussed extending a cross entropy and proposed the γ-dibergence [40]. It contains the logarithmic function and therefore it is more robust against outliers than other conventional divergence functions.

2.5. Dataset of industrial machinery sounds

One of the most critical tasks in machine learning research work is to utilize a well-defined large-scale dataset. Thanks to the numerous researchers and practitioners' efforts, there have been many open sound datasets. These open datasets have contributed to the recent remarkable advancement in data mining technique using sound data. For instance, Google researchers created audio event recognition of various sound segments extracted from YouTube [41]. Dekkers published the dataset of sound recorded daily activities in a building with annotations [42].

Regarding anomaly detection of machinery sound, Koizumi *et al.* introduced the dataset called "ToyADMOS", designed for anomaly detection [5]. The dataset collected from miniature machines (toys) operated in the controlled laboratory environment, not only normal condition sound but also abnormal condition sound, which is generally regarded challenging to sample. In some cases, such as research by Dong et al., they prepare suitable sound data from a surface-mounted device machine themselves for their model and opened it for other scholar's review [43]. These datasets have answered the growing attention on sound data-based industrial machinery anomaly detection and the emerging demand for appropriate sound datasets. However, these datasets are not enough when studying anomaly detection comes to various types of machine sounds in real factory environments, where the sound is inherently contaminated with background noise.

In September 2019, researchers at a Japanese manufacturing enterprise, Hitachi Co., Ltd., published a new dataset "Malfunctioning Industrial Machine Investigation and Inspection (MIMII)"("**MIMII Dataset**"), which is publicly available under *Creative Commons Attribution-Share Alike 4.0 International (CC BY-SA 4.0) license* [44]. The MIMII Dataset contains the sound of four different types of machines. The valves, pumps, fans, and slide rails sound are recoded. The pumps, which we utilized mainly for our experimental study, are water pumps that drain water from a pool and discharge water to the pool continuously. In this Project, we tested our proposed approaches on the MIMII Dataset to detail the property in section **4.3**.

2.6. Conclusion of the analysis

We had the literature review related to the Project in advance to discuss potential improvement of anomaly detection in noisy data. The statistical signal processing approaches and machine learning approaches have been intensively studied to extract desirable information from contaminated data. Interestingly, we found that these efforts have been studied independently. Signal processing and machine learning can be derived from multivariate analysis techniques such as the ordinary least square.

However, to our knowledge, few pieces of literature refer to the relationship between these approaches. Hence, we explored the relationship and developed a system incorporating signal processing and machine learning to improve the anomaly detection performance in noisy sound data. We selected the Kalman filter and the unscented Kalman filter as the adaptive digital filters. In terms of machine learning techniques, we chose to use an autoencoder, because a baseline study we adopted utilized an autoencoder, and also it is relatively succinct to apply modifications to the architecture. The modification should be encouraged by the insights of robust and sparsity induced estimation given in the literature.

3. PROJECTIONS

In this section, we introduce the theoretical backdrop of learning algorithms in an autoencoder. Since there are various kinds of architectures, optimization techniques and model training hacks, this chapter mainly describes the theories relevant to the model used in our research.

3.1. LEARNING ALGORITHMS OF AUTOENCODER

3.1.1. Feedforward neural network

The feedforward process is the process to evaluate output information deriving from the input information based on neural network architecture. We describe this process based on a simple instance, following the literature [36]. **Figure 1** describes the network diagram for a diagram for an autoencoder which has a double-layer architecture consisting of the input units (l=1), the hidden units (l=2) and the output units (l=3).

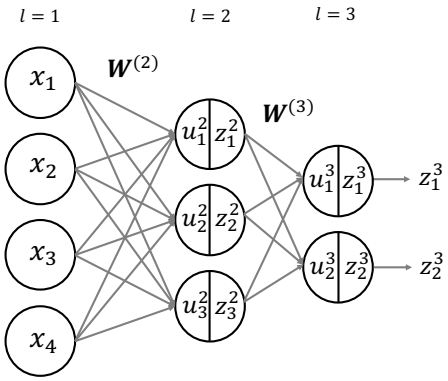

Figure 1 Network diagram for an autoencoder which has a double-layer architecture consisting of the input units (l=1), the hidden units (l=2) and the output units (l=3).

Each unit receives the inputs from the preceding units. The inputs are weighted and added a bias, and turns to its total input. For the example shown in **Figure 1**, each unit in the hidden units (l=2) receives four inputs from the layer (l=1) and returns its total inputsu_i, $(i = 1,2,3,4)$ as,

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} w_{11}w_{12}w_{13}w_{14} \\ w_{21}w_{22}w_{23}w_{24} \\ w_{31}w_{32}w_{33}w_{34} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}. \tag{2}$$

Then computes its outputs z_i, $(i = 1,2,3,4)$ by operating an activation function as,

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} f(u_1) \\ f(u_2) \\ f(u_3) \\ f(u_4) \end{pmatrix}. \tag{3}$$

This operation can be generally expressed by vectors and matrixes indicating its layer with l as,

$$\boldsymbol{u}^{(l+1)} = \boldsymbol{W}^{(l+1)}\boldsymbol{z}^{(l)} + \boldsymbol{b}^{l+1} \tag{4}$$

$$\boldsymbol{z}^{(l+1)} = \boldsymbol{f}\left(\boldsymbol{u}^{(l+1)}\right). \tag{5}$$

A variety of activation functions have been proposed and used. One of the most widely used and known by its Rectified linear function (ReLU) is used. The ReLU is written,

$$f(u) = \max(u, 0)\,. \tag{6}$$

The ReLU and its derivative are simple as shown below, and therefore, its computation complexity is relatively inexpensive.

$$\frac{\partial f}{\partial u} = \begin{cases} 1\ for\ u \geq 0 \\ 0\ for\ u < 0 \end{cases} \tag{7}$$

3.1.2. Learning algorithms

The parameters $\boldsymbol{W}$ are required to be optimized through the training. The optimization can be formulated as acquiring the estimation of parameters $\widehat{\boldsymbol{W}}$ when the data of the training data $\mathcal{D}$ consisting of observation $\boldsymbol{x}_n$ and target $\boldsymbol{d}_n$,

$$\mathcal{D} = \{(x_1, d_1), \cdots, (x_N, d_N)\} \tag{8}$$

by defining $\widehat{\boldsymbol{W}}$ to minimize a loss function $E(\boldsymbol{w})$. For instance, in case that the error is evaluated as ordinal squared error, then this is formulated as,

$$\widehat{\boldsymbol{W}} = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, E(\boldsymbol{w}), \quad E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\boldsymbol{d}_n - y(\boldsymbol{x}_n; \boldsymbol{w})\|^2 \tag{9}$$

Generally, $E(\boldsymbol{w})$ is not convex function, and therefore, the global minimum is almost impossible. Instead, the local minimum is acquired. The widely used approach to acquire a local minimum starts from an initial position and recursively updates its position. The gradient descent method is one of the simplest methods for recursive computation. The gradient is defined in vector form as,

$$\nabla E \equiv \frac{\partial E}{\partial \boldsymbol{w}} = \left[\frac{\partial E}{\partial w_1} \cdots \frac{\partial E}{\partial w_M}\right]^{\mathrm{T}} \tag{10}$$

By using this gradient, the gradient descent method updates the current weights as follows,

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \epsilon \nabla E \tag{11}$$

where ϵ denotes a learning rate, set as a hyperparameter. Despite various optimization techniques in nonlinear programming, the gradient descent method is the dominant method in machine learning tasks, especially when using the deep learning approach. A deep neural network architecture has a massive set of parameters, and it is not feasible to compute the second-order derivatives of the loss function. Since the derivative needs to be calculated, the method is only applicable to differentiable functions and can get stuck in a local minimum.

Stochastic gradient descent ("**SGD**") is the extension of the gradient descent method. In SGD, training data is randomly sampled from a dataset for each training iteration. This randomness can mitigate the possibility of trapping to a local minimum.

We mentioned that the learning rate is a hyperparameter. The learning rate is a significant factor in the training process. It swings the speed of convergence and that stability. Therefore, the methods to

determine and optimizing the learning rate have been studied, and various improvements have been proposed.

Momentum is one of the methods to improve the convergence performance of gradient descent. In momentum, updating of weight parameters $\epsilon \nabla E_t$ is calibrated the precedent updating, in other words, the average movement expressed as

$$\begin{aligned} \boldsymbol{w}^{(t+1)} &= \boldsymbol{w}^{(t)} - \epsilon \nabla E_t + \mu \Delta \boldsymbol{w}^{(t-1)}, \\ \Delta \boldsymbol{w}^{(t-1)} &= \boldsymbol{w}^{(t)} - \boldsymbol{w}^{(t-1)} \end{aligned} \tag{12}$$

where μ is a hyperparameter, generally determined in 0.5 ~ 0.9. The effect of momentum is illustratively shown in **Figure 2** [35]. The shown case represents that the loss function has a flat bottom floor, which makes stochastic descent fluctuate. Meanwhile, momentum term can calibrate the process and stably converges as a result.

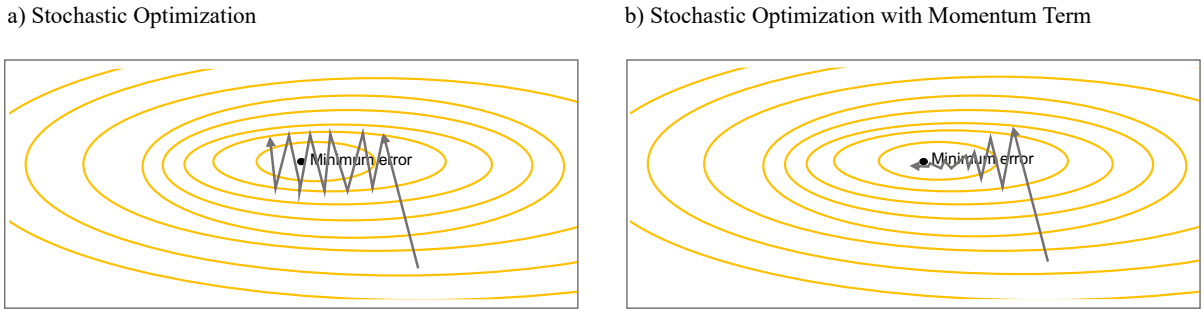

Figure 2 Illustration of a) stochastic descent process and b) stochastic optimization process with momentum term [35].

The RMSProp is another approach to suppress the fluctuation of the convergence process in SGD. In the RMSProp, the learning rate is calibrated by the significance of the gradient,

$$\boldsymbol{v}_t = \rho v_{t-1} + (1-\rho)\left(\nabla E\left(\boldsymbol{w}^{(t)}\right)\right)^2, \tag{13}$$

$$\Delta \boldsymbol{w}^{(t)} = -\frac{\eta}{\sqrt{\boldsymbol{v}_t + \varepsilon}} \nabla E\left(\boldsymbol{w}^{(t)}\right), \tag{14}$$

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \Delta \boldsymbol{w}^{(t)}, \tag{15}$$

where ε is the machine epsilon. A significant fluctuation results in the considerable value of v and consequently results in the small value of w. This feedback mechanism can suppress the fluctuation. The Adam, the name stands for the adaptive moment estimation, combines the Momentum and the RMSProp [13]. The Adam is widely used in recent research works, and we also use the algorithm in our network training.

3.1.3. Backpropagation

In order to update all the parameters in the neural network, the loss function needs to be derived with respect to all the parameters. This computation is complicated in the case of a multi-layered network. A backpropagation can simplify the computation by updating the units from the output layer to the input layer backwards. We describe the algorithm referring to the illustration in **Figure 3**.

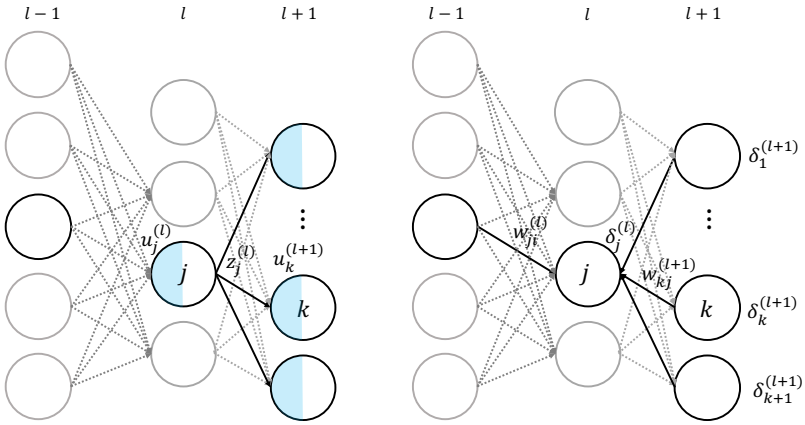

Figure 3 Illustration of a feedforward neural network. (Left) describes the differential of the total inputs on the unit impact on the loss function via the following layer's units. (Right) describes the backpropagation of the delta from l+1th to lth layer and computation of $\partial E/\partial w_{ji}^{(l)}$[36].

Let derive the loss function E_n with respect to the parameter $w_{ji}^{(l)}$ in the l-th layer. By applying the chain rule of differentiation, the derivative can be factorized as,

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial u_j^{(l)}} \frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} \tag{16}$$

where $u_j^{(l)}$ is the input to the l-th layer's j-th unit. The first term in the right-hand can be expressed as,

$$\frac{\partial E_n}{\partial u_j^{(l)}} = \sum_k \frac{\partial E_n}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}} \tag{17}$$

This expression represents that the $u_j^{(l)}$ effects on the loss function via the accompanied layer units $u_k^{(l+1)}$ as described in **Figure 3**. For simplification we introduce $\delta_j^{(l)}$ as,

$$\delta_j^{(l)} = \frac{\partial E_n}{\partial u_j^{(l)}} \tag{18}$$

Then the (18) can be described as,

$$\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} \left(u_{kj}^{(l+1)} f'\left(u_j^{(l)} \right) \right) \tag{19}$$

This equation indicates that the $\delta_j^{(l)}$ can be calculated by using the $\delta_j^{(l+1)}$. In an iterative manner, the computation can be started from the output layer toward the input layer. The second term in the right hand of the (16) is obviously,

$$\frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} = z_i^{(l-1)} \tag{20}$$

To the conclusion of the backpropagation, the derivative of the loss function with respect to the weight parameter is the product of the delta and output at the unit as,

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)} \tag{21}$$

The right fold of **Figure 3** highlights the factors to compute this differentiation over the l-th layer's j-th unit.

3.1.4. Loss functions in a neural network

The principal of anomaly detection with an autoencoder is that anomalous condition sound cannot be accurately encoded to the hidden layer and reconstructed from the hidden layer if the model is trained only with normal condition sound data. The accuracy of such decoding and reconstructing operation is evaluated with a loss function which measures the discrepancy between the input information and the reconstructed information.

$$L_{AE}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d) = \left\| \boldsymbol{x} - D\big((\boldsymbol{x}|\boldsymbol{\theta}_e)|\boldsymbol{\theta}_d\big) \right\|_2^2, \tag{22}$$

The loss function in a neural network architecture dictates the objective of the optimization. The solution acquired in MSE is $\hat{\mu} = \underset{\mu}{\operatorname{argmin}}\, L_{AE}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d)$ and the sample mean $\frac{1}{n}\sum_{i=1}^{n} y_i$. The loss function should be selected in accordance with the statistical assumption of the data to analyse. For instance, if noise is Gaussian, we can acquire the Best Linear Unbiased Estimator ("**BLUE**") as a solution the ordinary squared error.

3.1.5. Sparsity and robustness

It is a general problem to avoid overfitting in the training step. Overfitting to the training data could lower its generalization performance since the model may be optimized to the training data very rigidly. The other overfitting problem Various penalization is introduced to suppress the overfitting. There are two types of widely used regularization terms in a machine learning problem. One is the Ridge regularization, and another is the Lasso (Least Absolute Shrinkage and Selection Operator) regularization. The Ridge regularization was proposed in order to suppress multicollinearity. The Ridge regularisation is the addiction of l_2 norm of the parameters to the MSE as,

$$\text{Ridge regularization} \quad L_{AE_{ridge}}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d) = \left\| \boldsymbol{x} - D\big((\boldsymbol{x}|\boldsymbol{\theta}_e)|\boldsymbol{\theta}_d\big) \right\|_2^2 + \lambda \|\boldsymbol{\theta}\|_2 \tag{23}$$

Where λ denotes the regularization parameter and $\|\boldsymbol{\theta}\|_{\mathrm{p}}$ for the set of parameters $\boldsymbol{\theta} = \{\theta_1, \cdots, \theta_q\}$ is the l_p norm calculated as,

$$\|\boldsymbol{\theta}\|_p = l_p \text{ norm} = \left(\sum_{i=1}^{q} {\theta_i}^p \right)^{\frac{1}{p}} \tag{24}$$

On the other hand, the MSE with Lasso regularization is expressed as [31],

$$\text{Lasso regularization} \quad L_{AE_{lasso}}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d) = \left\| \boldsymbol{x} - D\big((\boldsymbol{x}|\boldsymbol{\theta}_e)|\boldsymbol{\theta}_d\big) \right\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1 \tag{25}$$

Figure 4 illustrates the effects of these regularizations [45]. The weight parameters approach zero, and this removes features unnecessary for representing the data structure. The L2 regularization makes the solution apart from the minimum error and the weights approach to zero but almost always never reaches. So, using the L2 regularization term enables sparsity induced representation.

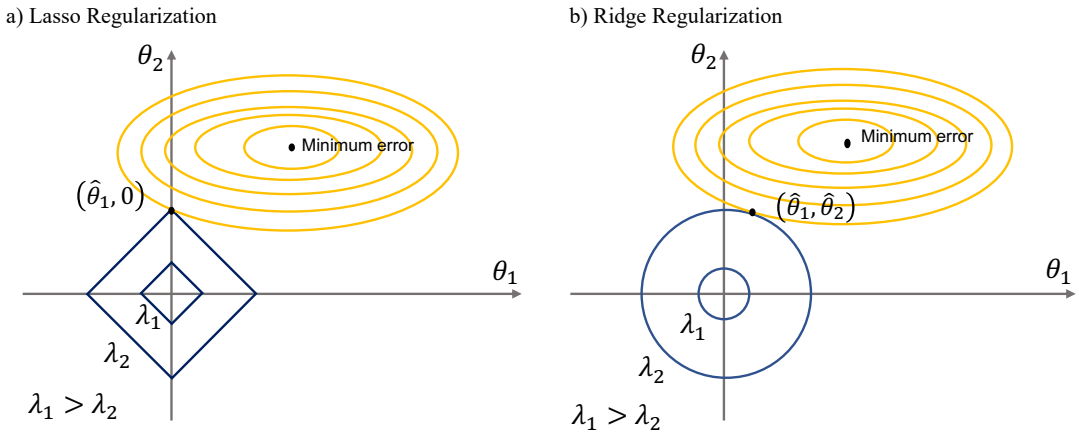

Figure 4 Illustration of a) Lasso regularization and b) Ridge regularization in the space spanned by two parameters [45].

The methods mentioned above can give us an estimator for sparse modelling. It works to keep only significant parameters among the possible set of parameters. Thus, L1 regularization enables sparse coding. As a fact, MSE with Lasso regularization term is known to be the dual problem of minimizing the problem,

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\text{minimize}} \|\boldsymbol{\theta}\|_1 \text{ subject to } \boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} \tag{26}$$

Kullback-Leibler divergence ("**KLD**") is also widely used as a regularization term for inducing sparsity [36;34]. It evaluates the dissimilarity between two densities of g and f, and it is defined as

$$\mathrm{D}_{\mathrm{KL}}(g, f) = \mathrm{E}_g\left[\log\frac{g(X)}{f(X)}\right] = \int g(x)\log\frac{g(x)}{f(x)}dx \tag{27}$$

KLD satisfies the following properties,

$$\mathrm{D}_{\mathrm{KL}}(g, f) \geq 0 \tag{28}$$

$$\mathrm{D}_{\mathrm{KL}}(g, f) = 0 \Leftrightarrow g(x) = f(x) \text{ almost everywhere.} \tag{29}$$

Implementation can be applied a binomial distribution for the density functions above, and thus the KLD can be written as,

$$\mathrm{D}_{\mathrm{KL}}(g, f) = \sum_{j=1}^{s_2}\left\{\rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right\}, \tag{30}$$

where s_2 is the number of neurons in the hidden layers. The index j is summing over the hidden units in our network. The overall loss function is now written,

$$L_{AE_{sparse}}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d) = \left\|\boldsymbol{x} - D\left((\boldsymbol{x}|\boldsymbol{\theta}_e)|\boldsymbol{\theta}_d\right)\right\|_2^2 + \sum_{j=1}^{s_2} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right). \tag{31}$$

Another approach to extract meaningful information from noisy data is the Least Absolute Deviance ("**LAD**") approach. The LAD considers the Mean Absolute Error ("**MAE**") as a loss function. The estimator acquired by the MAE can be expressed as,

$$\hat{\mu}' = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{n} |y_i - \mu| \tag{32}$$

To analyze the property of $\hat{\mu}'$, we sub-differentiate the summation and assume it is equal to zero,

$$\sum_{i:y_i<\hat{\mu}'} 1 + \sum_{i:y_i=\hat{\mu}'} \delta_i - \sum_{i:y_i>\hat{\mu}'} 1 = 0 \tag{33}$$

It shows that $\hat{\mu}'$ is equivalent to the sample median. Median is known as the robust statistics against outliers [46]. Hence the estimator acquired by MAE can weaken the effect of the outliers compared to the MSE. The MAE, however, may discard too much information during the estimator computation. Therefore, in a robust estimation problem, Huber loss, or called Smooth L1 loss, is widely used. The Huber loss is the modification of MAE to incorporate the information in a central region as a quadratic form, same as the MSE [39]. Mathematically, the Huber loss provided in `pytorch` is expressed as

$$\mathrm{HuberLoss}(x, y) = \frac{1}{n}\sum_i z_i \tag{34}$$

where z_i is given by

$$z_i = \begin{cases} \dfrac{0.5 \times (x_i - y_i)^2}{\beta}, & if\,|x_i - y_i| < \beta \\ |x_i - y_i| - 0.5 \times \beta, & otherwise. \end{cases} \tag{35}$$

Both of the estimators are known to be a class of M-estimator. This estimator has a solution which satisfies $\sum_{i=1}^{n} \psi(x_i; \theta) = 0$.

3.1.6. Autoencoder and principal component analysis

If the dimension of the hidden layer in an autoencoder is smaller than that of an input layer, its decoding operation can be understood as the extraction of embedded information from the input information. We describe the extraction mechanism by using an analogy of the principal component analysis ("**PCA**"). In PCA, the dispersion of data in p-dimension space is described with the covariance matrix $\boldsymbol{\Phi}$ of the training sample dataset $\boldsymbol{x} = \{x_1, \cdots, x_n\}$ can be expressed using the sample mean $\bar{x}$,

$$\boldsymbol{\Phi} = \frac{1}{N}\sum_{n=1}^{N} (x_n - \bar{x})\,(x_n - \bar{x})^{\mathsf{T}}. \tag{36}$$

If the data is embedded into a subspace, the subspace is spanned with the eigenvector of $\boldsymbol{\Phi}$. Let D_{hidden} denote the dimension of the hidden layer and $\mathbf{U}_{D_{hidden}}$ be a matrix containing the D_{hidden}

eigenvectors in descending order. Then the matrix $\left(\mathbf{U}_{D_{hidden}}, \bar{\boldsymbol{x}}\right)$ are the solution of the following optimization problem [46;47],

$$\left(\mathbf{U}_{D_{hidden}}, \bar{\boldsymbol{x}}\right) = (\boldsymbol{\Gamma}, \boldsymbol{\xi}) = \min_{\boldsymbol{\xi}, \boldsymbol{\Gamma}} \sum_{n=1}^{N} \left\| (x_n - \boldsymbol{\xi}) - \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{\Gamma} (x_n - \boldsymbol{\xi}) \right\|^2 \tag{37}$$

The equation implies that the eigenvector of $\boldsymbol{\Phi}$ spans the D_{hidden}-dimension subspace, which minimizes the Euclid norm. Therefore, the autoencoder embeds the input information into the D_{hidden}-dimension subspace in the encoding process. The decoding process of the autoencoder means reconstructing the data from the embedded subspace.

3.2. STATISTICAL SIGNAL PROCESSING AND STATE ESTIMATION

In this chapter, we introduce the theoretical fundamentals of statistical signal processing. The linear and non-linear system to be discussed. Then state-space model both for the linear system and the non-linear system is introduced. This estimation is made without precise knowledge of the underlying dynamic system. The Kalman filter algorithm computes the following two steps recursively. The one is the prediction step, where the $\boldsymbol{x}$ (state) and the $\boldsymbol{P}$ (state error covariance) are estimated using the previous state. The other is the correction step, where the state and error covariance are corrected using the current measurement.

3.2.1. Effects of noise presence on the ordinary least square solution

First, we discuss how noise effect the solution acquired by the MSE. The discussion in this section refers to the literature [48]. In the linear state-space model, let state-space be $\boldsymbol{x}$, and observation be $\boldsymbol{y}$ then the linear equation is written as,

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}, \tag{38}$$

where $\boldsymbol{H}$ is a linear operator of the M×N matrix, which maps $\boldsymbol{x}$ to $\boldsymbol{y}$. We consider the singular decomposition of $\boldsymbol{H}$ with the singular vector $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$, then $\boldsymbol{H}$ can be decomposed,

$$\begin{aligned} \boldsymbol{H} &= (\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_M) \begin{pmatrix} \gamma_1 & 0 & \cdots & 0 \\ 0 & \gamma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma_N \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_N^{\mathrm{T}} \end{pmatrix} \\ &= \sum_{j=1}^{N} \gamma_j \boldsymbol{u}_j \boldsymbol{v}_j^{\mathrm{T}} . \end{aligned} \tag{39}$$

By using $\boldsymbol{H}$, the optimal solution of the equation can be computed as,

$$\hat{\boldsymbol{x}} = \{(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\}\boldsymbol{y} = \left\{ \sum_{j=1}^{N} \frac{1}{\gamma_j} \boldsymbol{u}_j \boldsymbol{v}_j^{\mathrm{T}} \right\} \boldsymbol{y}. \tag{40}$$

If the observation is contaminated with noise, the equation is extended to include the noise vector $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_N)^\top$, where ε_j iid $\sim N(0, \sigma^2)$, then the equation is written as,

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{\varepsilon}. \tag{41}$$

In this case, the least square solution is modified as,

$$\begin{aligned}\hat{\boldsymbol{x}} &= \sum_{j=1}^{N} \frac{1}{\gamma_j} \boldsymbol{u}_j \boldsymbol{v}_j^\top \boldsymbol{y} = \sum_{j=1}^{N} \frac{1}{\gamma_j} \boldsymbol{u}_j \boldsymbol{v}_j^\top (\boldsymbol{H}\boldsymbol{x} + \boldsymbol{\varepsilon}) \\ &= \left\{ \sum_{j=1}^{N} \frac{1}{\gamma_j} \boldsymbol{u}_j \boldsymbol{v}_j^\top \right\} \boldsymbol{H}\boldsymbol{x} + \sum_{j=1}^{N} \frac{\boldsymbol{u}_j^\top \boldsymbol{\varepsilon}}{\gamma_j} \boldsymbol{v}_j \\ &= \left\{ \sum_{j=1}^{N} \frac{1}{\gamma_j} \boldsymbol{u}_j \boldsymbol{v}_j^\top \right\} \left\{ \sum_{j=1}^{N} \gamma_j \boldsymbol{u}_j \boldsymbol{v}_j^\top \right\} \boldsymbol{x} + \sum_{j=1}^{N} \frac{\boldsymbol{u}_j^\top \boldsymbol{\varepsilon}}{\gamma_j} \boldsymbol{v}_j \\ &= \boldsymbol{x} + \sum_{j=r+1}^{N} \frac{\boldsymbol{u}_j^\top \boldsymbol{\varepsilon}}{\gamma_j} \boldsymbol{v}_j \,. \end{aligned} \tag{42}$$

This result indicates that the estimator $\hat{\boldsymbol{x}}$ is biased from the true value $\boldsymbol{x}$ by the noise vector $\boldsymbol{\varepsilon}$ and the extent of the bias is amplified by $\frac{1}{\gamma_j}$. Therefore, if some of the singular value elements (for instance, $j = r+1, \cdots, N$) in $\boldsymbol{H}$ is nearly zero, the second term may be much larger than the true value,

$$\boldsymbol{x} \ll \sum_{j=r+1}^{N} \frac{\boldsymbol{u}_j^\top \boldsymbol{\varepsilon}}{\gamma_j} \boldsymbol{v}_j \,. \tag{43}$$

Such degeneracy of the singular value elements can happen if the observations have similar value to each other. Therefore, noise is encouraged to be removed for appropriate estimation with the least square approach, or we should seek other loss function as we discussed in section **3.1.5**. Henceforth, we discuss the adaptive filters of the KF and the UKF, starting from a fundamental information theory.

3.2.2. Bayes estimation and Cramer–Rao inequality in an optimized filter

The foundation of estimating unknown parameters based on a posteriori probability distribution is the problem of Bayes estimation. This section states how parameter estimation is formalized and the differences between linear and nonlinear systems [49]. An estimated error is computed with unknown parameter x and its estimate $\hat{x} = f(y)$,

$$e = x - f(y). \tag{44}$$

Let $l(e)$ be an error function, then the Bayes risk $R[f]$ is defined as the expectation of $l(e)$,

$$\begin{aligned} R[f] &= E\{l(x - f(y))\} \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} l(x - f(y)) p(x, y) dx dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} l(x - f(y)) p(x|y) p(y) dy \,. \end{aligned} \tag{45}$$

Bayes estimate is the estimated value of $f(y)$ which minimize the Bayes risk. In addition, we define the conditional Bayes risk as follows,

$$R_c[f] = E\{l(x - f(y))|y\} = \int_{-\infty}^{\infty} l(x - f(y))p(x|y)dx\,. \tag{46}$$

Then, the Bayes risk is the expectation of the conditional Bayes risk. It is known that the $\hat{x} = f(y) = \underset{x}{\operatorname{argmin}} R_c[f]$ concurrently minimizes the Bayes risk, that is to say, the Bayes estimate can be acquired by minimizing the conditional Bayes risk. By using an information matrix and its inverse matrix, an estimated error covariance matrix can be bounded. Let $x \in \mathbb{R}^n$be an unknown parameter vector, $y \in \mathbb{R}^p$be an observation vector, and $f(y)$ be an estimate of x based on y. Bayes information matrix is defined as,

$$J = -\mathrm{E}\left\{\frac{\partial^2 \log p(x,y)}{\partial x^2}\right\}, \tag{47}$$

where the elements are expressed as,

$$J_{i,j} = -\mathrm{E}\left\{\frac{\partial^2 \log p(x,y)}{\partial x_i \partial x_j}\right\}, \quad i,j = 1,\dots,n. \tag{48}$$

For any nonlinear system, the posteriori Cramer-Rao inequality is

$$\mathrm{E}\{[\check{x}_t^- - x_t][\check{x}_t^- - x_t]^\top\} \geq J_t^{-1}(x_t), \tag{49}$$

$$\mathrm{E}\{[\check{x}_t - x_t][\check{x}_t - x_t]^\top\} \geq J_t^{-1}(x_t) \tag{50}$$

where $\check{x}_t^-$ and $\check{x}_t$ denote the one-step prediction estimation and the one-step filtered estimation. The (49) and (50) indicate that these estimation for any nonlinear filter can be bounded by the inverse of Bayes information matrix.

3.2.3. Linear system estimation and Kalman filter

Kalman filter can be derived based on the following assumptions, which is collectively called Linear-Quadratic-Gaussian control problem; linearity, white, Gaussian, and quadratic error criteria. The KF considers a normal distribution and describes a state estimate as the mean value of normal distribution, and evaluate its uncertainty by the covariance matrix. The Kalman filtering is a recursive process. It recursively updates the mean value and covariance matrix. **Figure 5** describes the process [30]. As the data acquired stepwise in the order $k-4, k-3, \dots, k$, the covariance matrix is updated, and the uncertainty decreases.

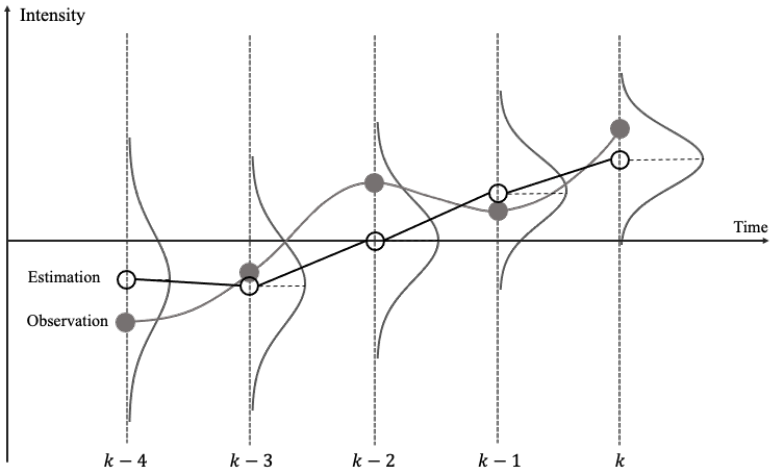

Figure 5 Illustrative concept of Kalman filtering and its state estimation process [30].

In the mathematical context, the Kalman filtering assumes that the linear discrete-time state-space model can express a scalar time series data as,

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{b}v(k), \tag{51}$$

$$y(k) = \boldsymbol{c}^{\top}\boldsymbol{x}(k) + w(k). \tag{52}$$

The linear predictor model is written using the priori estimate $\hat{\boldsymbol{x}}^{-}(k)$ and the latest observation $y(k)$,

$$\hat{\boldsymbol{x}}(\mathrm{k}) = \boldsymbol{G}(k)\hat{\boldsymbol{x}}^{-}(k) + \boldsymbol{g}(k)y(k). \tag{53}$$

$$\hat{\boldsymbol{x}}^{-}(k) = \boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k) \tag{54}$$

Then MSE can be defined as,

$$\hat{\boldsymbol{x}}(k) = \underset{\boldsymbol{x}(k)}{\operatorname{argmin}} J(k), \qquad J(k) = \mathrm{E}\left[\tilde{\boldsymbol{x}}^{2}(k)\right]. \tag{55}$$

The Kalman filtering is the process to acquire the optimal estimate which satisfies the minimum mean square error. For its computation, the Kalman filtering is based on the assumption that the noise follows the normal distribution. This assumption derives two essential characteristics of the normality of random variables and of that the normal distribution can be defined by only first-order moment (mean) and second-order moment (variance).

The characteristics that normality is conserved through linear transformation is the fundamental characteristics of designing the KF. A probabilistic density function of multivariate $\boldsymbol{x}$ with mean vector $\bar{\boldsymbol{x}}$ and covariance matrix $\boldsymbol{P}_x$ is expressed as

$$p(x) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{P}_x|}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\top} P_x^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}})\right]. \tag{56}$$

The affine transformation over the multivariate produces a multivariate $\boldsymbol{y}$,

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \tag{57}$$

where $\boldsymbol{y}$ follows Gaussian distribution and its mean and covariance can be expressed,

$$\bar{\boldsymbol{y}} = \boldsymbol{A}\bar{\boldsymbol{x}} + \boldsymbol{b}, \qquad \boldsymbol{P_y} = \boldsymbol{A}\boldsymbol{P_x}\boldsymbol{A}^T. \tag{58}$$

The affine transformation modifies the mean and covariance matrix, but the probability density function ("**PDF**") of $\boldsymbol{y}$ can be expressed in a Gaussian distribution manner, using this mean and covariance matrix.

$$p(\boldsymbol{y}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{P}_y|}} \exp\left[-\frac{1}{2}(\boldsymbol{y} - \bar{\boldsymbol{y}})^\top \boldsymbol{P}_y^{-1}(\boldsymbol{y} - \bar{\boldsymbol{y}})\right]. \tag{59}$$

If the system noise and observation noise are normal, the first-order moment and second-order moment can determine the characteristics of the distribution. The following equation expresses posteriori state estimation and the orthogonal principal,

$$\mathrm{E}[\tilde{\boldsymbol{x}}^-(k)y(i)] = \mathrm{E}\big[\big(\boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k)\big)y(i)\big] = \boldsymbol{0}, \;\; i = 1,2,\dots,k. \tag{60}$$

A posteriori covariance matrix $\boldsymbol{P}^-(k)$ is defined as,

$$\boldsymbol{P}^-(k) = \mathrm{E}\left[\tilde{\boldsymbol{x}}^-(k)\big(\tilde{\boldsymbol{x}}^-(k)\big)^\top\right] = \mathrm{E}[\{\boldsymbol{x}(k) - \hat{\boldsymbol{x}}^-(k)\}\{\boldsymbol{x}(k) - \hat{\boldsymbol{x}}^-(k)\}^\top]. \tag{61}$$

Then we obtain the Kalman gain

$$\boldsymbol{g}(k) = \frac{\boldsymbol{P}^-(k)\boldsymbol{c}}{C^\top \boldsymbol{P}^-(k)\boldsymbol{c} + \sigma_w^2}. \tag{62}$$

We henceforth describe the KF algorithm. The initialization step is,

$$\hat{\boldsymbol{x}}(0) = \mathrm{E}[\boldsymbol{x}(0)] = \boldsymbol{x}_0, \tag{63}$$

$$\boldsymbol{P}(0) = \mathrm{E}[(\boldsymbol{x}(0) - \mathrm{E}[\boldsymbol{x}(0)])(\boldsymbol{x}(0) - \mathrm{E}[\boldsymbol{x}(0)])^\top] = \boldsymbol{\Sigma}_0. \tag{64}$$

It is being updated sequentially for $k = 1,2,\cdots,N$. The prediction step is,

$$\text{Priori estimate}: \; \hat{\boldsymbol{x}}^-(k) = \boldsymbol{A}\hat{\boldsymbol{x}}(k-1), \tag{65}$$

$$\text{Priori covariance matrix}: \; \boldsymbol{P}^-(k) = \boldsymbol{A}\boldsymbol{P}(k-1)A^\top + \sigma_v^2 \boldsymbol{b}\boldsymbol{b}^\top. \tag{66}$$

Once obtain the actual observation $y(k)$, it proceeds to the filtering step,

$$\text{Kalman Gain}: \; \boldsymbol{g}(k) = \frac{\boldsymbol{P}^-(k)\boldsymbol{c}}{C^\top \boldsymbol{P}^-(k)\boldsymbol{c} + \sigma_w^2}, \tag{67}$$

$$\text{Posteriori estimation}: \; \hat{\boldsymbol{x}}(k) = \hat{\boldsymbol{x}}^-(k) + \boldsymbol{g}(k)\big(y(k) - \boldsymbol{c}^\top \hat{\boldsymbol{x}}^-(k)\big), \tag{68}$$

$$\text{Posteriori covariance matrix}: \boldsymbol{P}(k) = (I - \boldsymbol{g}(k)\boldsymbol{c}^\top)\boldsymbol{P}^-(k). \tag{69}$$

The essence of the Kalman filter is this iterative update process of the covariance matrix.

The KF can be extended to a nonstationary process. Let consider the linear discrete-time state-space model with varying coefficients $\{\boldsymbol{A}(k), \boldsymbol{b}(k), \boldsymbol{c}(k)\}$, which satisfies the following state-space model,

$$\boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k) + \boldsymbol{b}(k)v(k), \tag{70}$$

$$y(k) = \boldsymbol{c}^\top(k)\boldsymbol{x}(k) + w(k). \tag{71}$$

Kalman filtering over the nonstationary time-series data $y(k)$ can be developed by substituting invariants set $\{\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}\}$ in the updating process to the varying coefficients set $\{\boldsymbol{A}(k), \boldsymbol{b}(k), \boldsymbol{c}(k)\}$. In this case, system noise and observation noise can be time-varying.

3.2.4. Nonlinear system estimation and the unscented Kalman filter

Considering that the data we use for anomaly detection experiments is the real-world sound data with the background noise recorded in real factories, it is natural to assume the data is a nonlinear system. In contrast to the linear system, it is difficult to estimate the distribution in the nonlinear system by using the first and second-order moments because third and higher-order moments are not negligible and hinder state estimation. In non-linear filtering, it is essential to consider an approximation filter. An approximation value of the error covariance matrix is required. Posterior Cramer–Rao inequality is used to evaluate the estimated error covariance matrix [49]. A non-linear discrete system is expressed as,

$$\boldsymbol{x}(k+1) = \boldsymbol{f}\big(\boldsymbol{x}(k)\big) + \boldsymbol{b}v(k), \tag{72}$$

$$y(k) = h\big(\boldsymbol{x}(k)\big) + w(k). \tag{73}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a state vector, y is an observation value, v is a system noise value, and w is an observation noise vector. The function $\boldsymbol{f}(\cdot)$ is a non-linear vector function and $h(\cdot)$ is a scalar value non-linear function. Likewise, the formulation we discussed on Kalman filtering, the state estimation program is defined as finding the optimized estimator $\widehat{\boldsymbol{x}}_{t+m/t}$ Which minimize the Bayes risk,

$$J = \mathrm{E}\left\{\left\|\boldsymbol{x}_{t+m} - \widehat{\boldsymbol{x}}_{t+\frac{m}{t}}\right\|^2\right\}, \qquad m = 0,1. \tag{74}$$

Linearization is the process deriving Jacobian from a nonlinear function use only first-order series. Using the linearization in approximating a nonlinear system is the extended Kalman filter ("**EFK**"). This can introduce large errors in the mean vector and covariance matrix, which may lead to lower performance of the filter. The UKF addresses this problem by using a deterministic sampling approach [31;50]. Instead of using a linear approximation of the nonlinear function, approximating the probability density function is the central concept. In the deterministic approach, the posterior state distribution is again approximated as a normal distribution, but is now represented with a set of sample points. This approach is more expensive than the linearization approach but the UKF has the computational complexity similar extent to that of EKF [30]. **Figure 6** illustrates a linear system and approximation approaches used in the linear transformation, the EKF and the UKF [30].

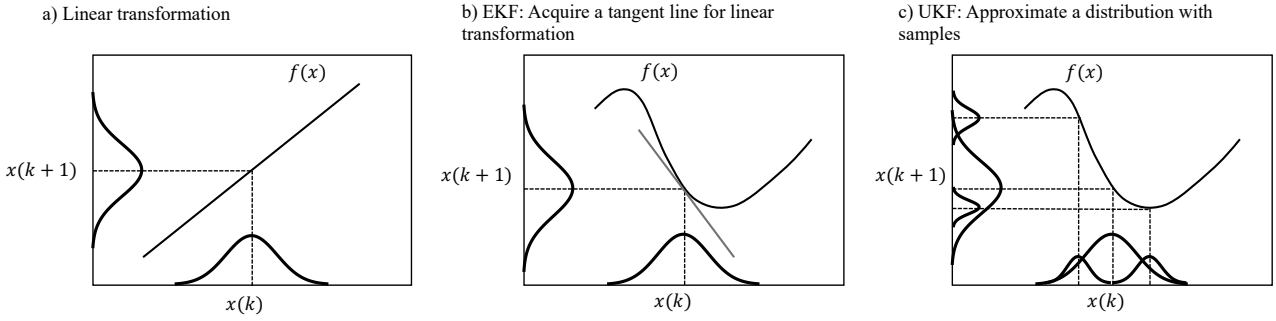

Figure 6 a) Illustrative concept of a linear transfer from x(k) to x(k+1), b) illustration of EKF which utilizes a tangent line of the nonlinear mapping function, c) illustration of UKF, which represents approximated distribute with sample points [30].

The central concept of the UKF is an approximation of posterior probabilistic density function (PDF) with normal distribution, instead of acquiring a tangent line of a nonlinear system, which is the central concept of the EKF. In order to approximate a posterior PDF, the UKF introduces the unscented transformation ("**UT**"). We describe the UT for the preparation of the UKF algorithm discussed later. We consider a non-linear mapping function $f: \Re^n \rightarrow \Re^n$ which transform n-dimensional random variables x to n-dimensional random variables y,

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}). \tag{75}$$

Let $\bar{\boldsymbol{x}}$ be the mean of x, and P_x be covariance matrix of x. The problem can be defined as computing the first and second-order moments of y. The sigma points $\{\boldsymbol{\mathcal{X}}_i, i = 0,1,2, \dots ,2n\}$ are selected according to the following rules,

$$\boldsymbol{\mathcal{X}}_0 = \bar{x}, \tag{76}$$

$$\boldsymbol{\mathcal{X}}_i = \bar{x} + \sqrt{n+\kappa}\left(\sqrt{P_x}\right)_i, \tag{77}$$

$$\boldsymbol{\mathcal{X}}_{n+i} = \bar{x} - \sqrt{n+\kappa}\left(\sqrt{P_x}\right)_i, \tag{78}$$

where κ is a scaling parameter and $\left(\sqrt{P_x}\right)_i$ is the i-th column of the square root of matrix P_x. P_x is positive determinant, and the matrix square root is computed by Cholesky factorization or singular value decomposition. Then weights on each sigma point are given as,

$$w_0 = \frac{\kappa}{n+\kappa}, \tag{79}$$

$$w_i = \frac{1}{2(n+\kappa)}, \quad i = 1, 2, \dots, 2n. \tag{80}$$

where the weights are normalized to suffice $\sum_{i=0}^{2n} w_i = 1$.

$$\boldsymbol{\mathcal{Y}}_i = \boldsymbol{f}(\boldsymbol{\mathcal{X}}_i), i = 0, 1, \dots, 2n. \tag{81}$$

By using $\boldsymbol{\mathcal{Y}}_i$, the first order and second-order moments of the transformed y, mean $\bar{y}$ and covariance matrix $\boldsymbol{P}_y$, respectively, can be computed as,

$$\bar{y} = \sum_{i=0}^{2n} w_i \boldsymbol{\mathcal{Y}}_i \tag{82}$$

$$\boldsymbol{P}_y = \sum_{i=0}^{2n} w_i \left(\boldsymbol{\mathcal{Y}}_i - \bar{y}\right)\left(\boldsymbol{\mathcal{Y}}_i - \bar{y}\right)^{\mathsf{T}} \tag{83}$$

The estimates of the first-order moment and the second-order moment are equivalent to the second-order Taylor expression of any nonlinear function. **Figure 7** illustrates how the sampled sigma points are mapped by a nonlinear function [30].

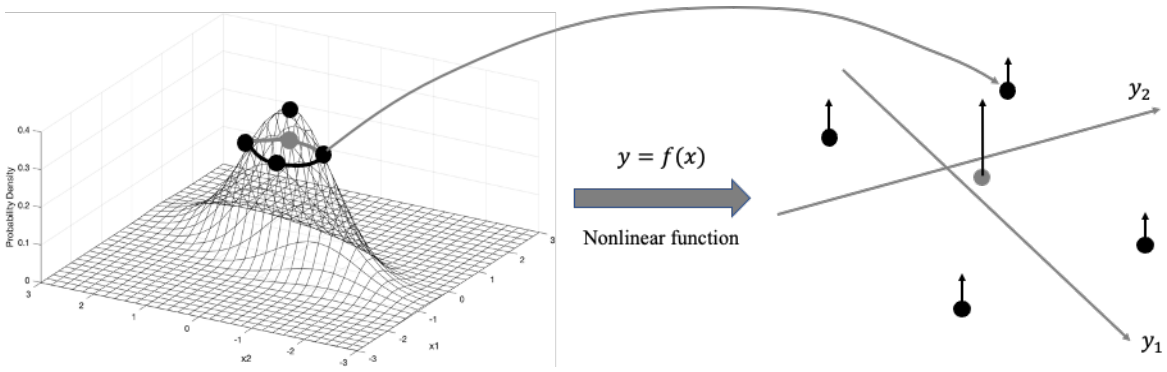

Figure 7 Illustrative of sigma points on bivariate normal distribution probability density function and its U-transformation mapping of the points onto the $y_1 - y_2$ plane [30].

Assumption that the conditional probabilistic density function follows gaussian distribution,

$$p(x_t|Y^{t-1}) = N\left(x_t\middle|\hat{x}_{\frac{t}{t}-1}, P_{\frac{t}{t}-1}\right). \tag{84}$$

Likewise, as the KF, we describe the UKF algorithm. First, we have the initialization step,

$$\hat{\boldsymbol{x}}(0) = \mathrm{E}[\boldsymbol{x}(0)] = \boldsymbol{x}_0, \tag{85}$$

$$\boldsymbol{P}(0) = \mathrm{E}[(\boldsymbol{x}(0) - \mathrm{E}[\boldsymbol{x}(0)])(\boldsymbol{x}(0) - \mathrm{E}[\boldsymbol{x}(0)])^{\mathrm{T}}] = \boldsymbol{\Sigma}_0. \tag{86}$$

It is being updated sequentially for $k = 1,2,\cdots,N$. The UKF has to compute sigma points to prepare for approximation,

$$\boldsymbol{\mathcal{X}}_0(k-1) = \hat{\boldsymbol{x}}(k-1), \tag{87}$$

$$\boldsymbol{\mathcal{X}}_i(k-1) = \hat{\boldsymbol{x}}(k-1) + \sqrt{n+\kappa}\left(\sqrt{\boldsymbol{P}(k-1)}\right)_i, \tag{88}$$

$$\boldsymbol{\mathcal{X}}_{n+i}(k-1) = \hat{\boldsymbol{x}}(k-1) - \sqrt{n+\kappa}\left(\sqrt{\boldsymbol{P}(k-1)}\right)_i. \tag{89}$$

where the weights are computed as follows,

$$w_0 = \frac{\kappa}{n+\kappa}, \qquad w_i = \frac{1}{2(n+\kappa)}, \qquad i = 0, 1, \dots, 2n. \tag{90}$$

Then, the prediction step updates the sigma points using the mapping function $\boldsymbol{f}$, acquiring

$$\boldsymbol{\mathcal{X}}_i^-(k) = \boldsymbol{f}\big(\boldsymbol{\mathcal{X}}_i(k-1)\big), i = 0, 1, \dots, 2n. \tag{91}$$

A priori estimation is weighted summation of the mapped values,

$$\hat{\boldsymbol{x}}^-(k) = \sum_{i=0}^{2n} w_i\, \boldsymbol{\mathcal{X}}_i^-(k). \tag{92}$$

Using the (92), A priori covariance matrix is written as,

$$\boldsymbol{P}^-(k) = \sum_{i=0}^{2n} w_i \{\boldsymbol{\mathcal{X}}_i^-(k) - \hat{\boldsymbol{x}}^-(k)\}\{\boldsymbol{\mathcal{X}}_i^-(k) - \hat{\boldsymbol{x}}^-(k)\}^{\mathrm{T}} + \sigma_v^2 \boldsymbol{b}\boldsymbol{b}^{\mathrm{T}}. \tag{93}$$

Then, we re-calculate sigma points as,

$$\boldsymbol{\mathcal{X}}_0(k) = \hat{\boldsymbol{x}}(k), \tag{94}$$

$$\boldsymbol{\mathcal{X}}_i(k) = \hat{\boldsymbol{x}}(k) + \sqrt{n+\kappa}\left(\sqrt{\boldsymbol{P}^-(k)}\right)_i, \tag{95}$$

$$\boldsymbol{\mathcal{X}}_{n+i}(k-1) = \hat{\boldsymbol{x}}(k) - \sqrt{n+\kappa}\left(\sqrt{\boldsymbol{P}^-(k)}\right)_i. \tag{96}$$

Update the observation using the sigma points and compute a priori observation estimate,

$$\mathcal{Y}_i^- = h\big(\boldsymbol{\mathcal{X}}_i^-(k)\big). \tag{97}$$

$$\hat{y}^-(k) = \sum_{i=0}^{2n} w_i \, \mathcal{Y}_i^-. \tag{98}$$

A priori observation error covariance matrix can be computed as,

$$\boldsymbol{P}_{yy}^-(k) = \sum_{i=0}^{2n} w_i \, \{\mathcal{Y}_i^-(k) - \hat{y}^-(k)\}^2. \tag{99}$$

Similarly, A priori state and observation error covariance matrix can be computed as,

$$\boldsymbol{P}_{xy}^-(k) = \sum_{i=0}^{2n} w_i \, \{\boldsymbol{\mathcal{X}}_i^-(k) - \hat{\boldsymbol{x}}^-(k)\}\{\mathcal{Y}_i^-(k) - \hat{y}^-(k)\}. \tag{100}$$

Thus, we acquire the Kalman gain in the UKF algorithm,

$$\boldsymbol{g}(k) = \frac{\boldsymbol{P}_{xy}^-(k)}{\boldsymbol{P}_{yy}^-(k) + \sigma_w^2}. \tag{101}$$

For the filtering step, the estimation and the error covariance matrix are updated using the Kalman gain as,

$$\text{a posteriori estimation } \hat{\boldsymbol{x}}(k) = \hat{\boldsymbol{x}}^-(k) + \boldsymbol{g}(k)\{y(k) - \hat{y}^-(k)\}. \tag{102}$$

$$\text{a posteriori error covariance matrix } \boldsymbol{P}(k) = \boldsymbol{P}^-(k) - \boldsymbol{g}(k)\boldsymbol{P}_{xy}^-(k)^\top. \tag{103}$$

3.3. PROPOSED APPROACH

In this section, we formalize the settings for the system design proposed to improve the anomaly detection performance with noisy sound data. From the motivation of developing a noise-tolerable detection system, we propose to combine two approaches; denoising the noisy sound data with a digital adaptive filter and optimizing an autoencoder structure for sparse coding. As we mentioned in section **2**, few preceding studies treat the digital filters and neural network as a single system. In contrast, our proposed approach explores the relationship between the digital filter and the autoencoder structure and improve the anomaly detection performance.

The real-world sound data is generally contaminated with noise, and therefore it is natural to conjecture that denoising and sparse coding are essential to improve anomaly detection performance. We propose to use the KF and the UKF as adaptive digital filters for the data pre-processing. As aforementioned, the KF is derived from a linear system, and the UKF is from a non-linear system. Both digital filters are also derived from Gaussian noise. For the autoencoder system, we propose to use the finer input vector, finer hidden units, and various loss functions, in addition to the baseline autoencoder model used in the literature by the real-world dataset provider. The combination of the pre-processing and the autoencoder variants conducted experimental evaluations is summarized in **Table 1**.

Table 1 The list of the pre-processing and the autoencoder variants conducted the experimental evaluation in this study. The "Baseline" denotes the property of the baseline autoencoder model mentioned above.

Pre-processing	Input unit / hidden unit dimension	Loss function
Unprocessed - Baseline	320 dim / 8 dim - Baseline	Mean Squared Error (MSE) - Baseline
Kalman Filter	2565 dim / 8 dim	MSE with Lasso regularization (L1)
Unscented Kalman Filter	2565 dim / 2565 dim	MSE with Ridge regularization (L2)
		MSE with Kullback-Leibler divergence (KLD) regularization
		Mean Absolute Error (MAE)
		Huber Loss

When applying the digital filters of the KF and UKF for the sound data, we acquire the state estimation results as pre-processed sound data. Although we have little mechanical and operational details about the machine we analyse, it is natural to address that the UKF has better state estimation performance.

In its simplest form, an autoencoder is composed of two parts, an encoder and a decoder. It was introduced as a technique for dimensionality reduction, and it is known as very similar to the PCA if the loss function is the MSE as mentioned in section **3.1.6**. We address that using the other loss functions listed above could encourage robustness of the representation in noisy data. MSE with Lasso is supposed to encourage sparseness of the parameters in autoencoder, thus it can avoid overfitting the model to incorporate noise. MSE with Ridge is tested for reference purpose in our study. We expect it prevent overfitting, but the trained model may incorporate noise. The MSE with KLD is supposed to play a similar role to the MSE with Lasso. The MAE is for robust estimation purpose. Huber loss is also for robust estimation but designed to incorporate more information than the MAE. It is our interest to experimentally obtain an insight that the noise in the pre-processed data should be treated as an outlier or just noise.

4. EXPERIMENTS

4.1. Experiment protocols

The performance of anomaly detection is measured in the index of AUC values [51]. It is a proven methodology to evaluate binary classifier output quality used in communication engineering. In the evaluation process, a Receiver Operating Characteristic ("**ROC**") is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold changes. An AUC value is defined by the area under the ROC curve. AUC has a range of 0 to 1. The higher AUC means the higher performance of binary classification, and 0.5 means the discriminator judges the result randomly. By plotting true positive rate (TPR) against the false positive rate (FPR) at various threshold settings, these fractions create the curve. Compared to metrics such as the subset accuracy, the Hamming loss, or the F1 score, ROC does not require explicitly optimizing a threshold for each label.

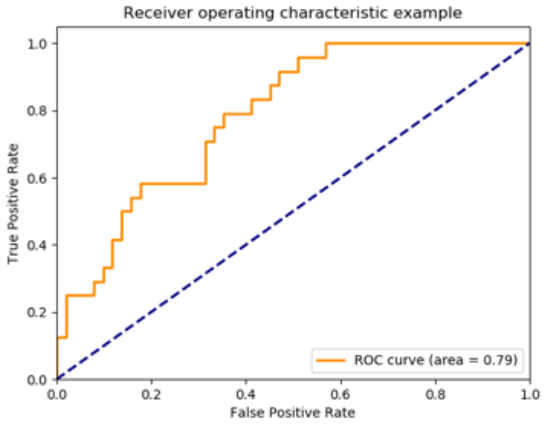

Figure 8 Example of the area under the curve and receiver operating characteristic [52].

Figure 9 illustrates the overall experiment procedures of the proposed architecture. For the proposed architecture examination, the dataset was pre-processed Unscented Kalman Filter. Meanwhile, we use the dataset (raw) to verify our implementation and environment for the reproductive work. The UKF filtered dataset and the raw dataset are converted to time-frequency space by Short Time Fourier Transformation ("**STFT**"). For the reproductive work, a log-Mel spectrogram was generated from the raw dataset. Both of the input features are tested for the proposed architecture, and AUC results were evaluated and discussed.

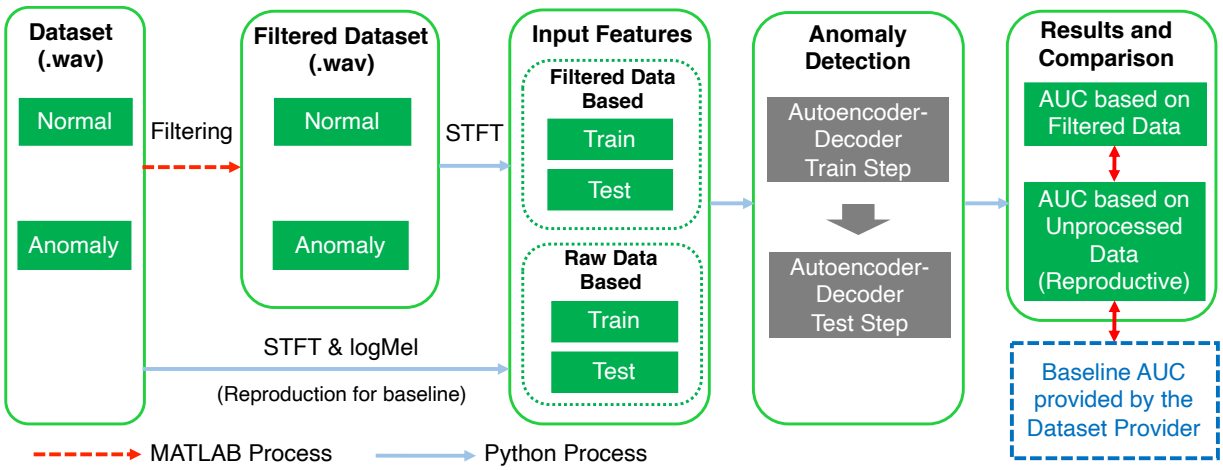

Figure 9 Overall workflow in this experiment.

4.2. Analysis tools

4.2.1. Implementation environment

We used a conventional laptop of MacBook Pro (ver. later 2019) for the main machine of analysis. The machine specification is as follows; Processor – 8 Core Intel Core i9, Processor Clock – 2.4 GHz (1 processor), No of Processor – 1, and RAM – 32GB.

4.2.2. Software

Python (version 3.7.3) is the primary programming language for our research. The widely-used and proven libraries such as `PyTorch`, `scikit-learn`, and `Keras` are used. The purpose of usage of these libraries are expedition of model development and reduce coding error which might mislead results and discussion. A novelty portion in a proposed algorithm in this research will be developed from scratch if the proposed algorithm cannot be developed by modifying the existing library's application programming interface.

MATLAB (version R2020b Update 5, 64-bit) is used for pre-processing the dataset and the data visualization. MATLAB is recognized as one of the most reputable software for signal processing, especially for state estimation. The KF and the UKF methods are provided as the add-ons of "Control System Toolbox", "DSP System Toolbox", and "Signal Processing Toolbox" [53].

4.3. Dataset and feature engineering

Our proposed approach was tested on the real-world sound dataset, the MIMII Dataset, aforementioned in section **2.5** [44]. The dataset was collected using a microphone. It consists of eight separate microphones which enables us to evaluate multi-channel approaches. The microphone array was kept at a distance of 50 cm from the and 10-second sound segments were recorded. The dataset accordingly contains eight separate channels for each segment. The sound was recorded as 16-bit audio signals sampled at 16 kHz. The file of one channel in one segment consists of 160000 samples (=160000 × 10) of time frames. Apart from the target machine sound, background noise in real factories was recorded and mixed with the target machine sound for simulating real environments at

the three-level of Signal-Noise-Ratio ("**SNR**"); 6dB, 0dB and -6dB. These levels can be considered as clean, moderate and noisy sound data respectively. The SNR is defined to be $10\log_{10}(a/b_j)$, where a is the average power over all segments of the machine models, and b_j is a power of a background-noise segment is randomly selected and denoted as j. The dataset is provided in the format of Waveform Audio File (.wav extension).

The MIMII dataset contains the sound of four different types of machines as we introduced in the section **2.5**. Among the four kinds of machines, pumps and fans are stationary, and others are non-stationary. Since non-stationary types of machinery require relevant domain knowledge in actual operation, we decided to focus on the pump dataset to implement proposed architectures. The pump's sounds were recorded for both normal condition and abnormal condition. An example of the anomalous conditions of pumps was leakage, contamination and clogging. The anomalous sound data was labelled only as anomalous condition, and no further description was provided for each wav data file. The list of sound data files of pumps is summarized in **Table 2**. The pump sound dataset consists of four different pumps, labelled Model ID00, 02, 04 and 06. The number of segments for the normal condition of each machine is seven to ten times larger than that of anomalous condition.

Table 2. MIMII Dataset pump content detail [44].

Model ID	Segments for normal condition	Segments for anomalous condition
ID00	1006	143
ID02	1005	111
ID04	702	100
ID06	1036	102

The dataset provider conducted the baseline experiment. In their experiment, anomaly detection was performed for each segment by thresholding the reconstruction error averaged over 10 sec. The detail of the autoencoder neural network they used is described in section **4.5**. The network is trained by the Adam optimization technique for 50 epochs to minimize the loss function of the least square as formulated in the (22).

For each model ID, all the segments were split into a training dataset and a test dataset. All the anomalous segments were used as the test dataset. And the same number of normal segments dataset was randomly selected and used as the test dataset. All the rest of the normal segments were used as the training dataset. Anomaly detection was performed for each segment by thresholding the reconstruction error averaged over ten seconds in accordance with the Dataset provider's experimental protocols [44]. **Table 3** shows the baseline results of the AUC for the pumps. Based on the intense relationship between the input SNR and the resulting AUC, we mainly used the model ID 06 to experiment with our proposed approach. This relationship was reproduced and confirmed in our initial data analysis shown in subsection **4.6.3**.

Table 3. The baseline AUCs of pumps with ID: 00, 02, 04 and 06 at input SNR of 6 dB, 0 dB and -6 dB presented by the Dataset provider [44].

Model ID	Input SNR		
	6 dB	0 dB	-6 dB
ID00	0.84	0.65	0.58
ID02	0.45	0.46	0.52
ID04	0.99	0.95	0.93
ID06	0.94	0.76	0.61

The feature engineering in the experiment follows the dataset provider's procedure. Each segment of waveform sound data is mapped into frequency-intensity space with the STFT. The Fast Fourier Transformation ("**FFT**") window size was 1024, and the hop size was 512. The Hanning window was applied. Among the one segment of 160000 samples, the beginning and ending 128 samples are trimmed and acquired 1159744 (=160000-256) samples for one segment. With the above set window size and hop size, we obtained 311 time-frames for one segment, and each time frame of the snapshot has frequency bins of 513 (=1024/2+1). The log-Mel spectrogram was generated with Mel 64 filter bank from the frequency-intensity space. The input feature of the 320-dimensional feature vector was generated by combining the adjacent five timeframes. In addition to the input feature vectors in line with the baseline mentioned above work for reproductive work purpose, we developed feature inputs with finer frequency resolution ("**Finer Input Feature Vector**"). Instead of applying Mel filter bank, we used the features acquired by the STFT directory. Accordingly, this finer resolution feature input vector has 513 bins in frequency and generates a 2565-dimensional input feature vector by combining adjacent five timeframes. These processes are illustratively shown in **Figure 10**.

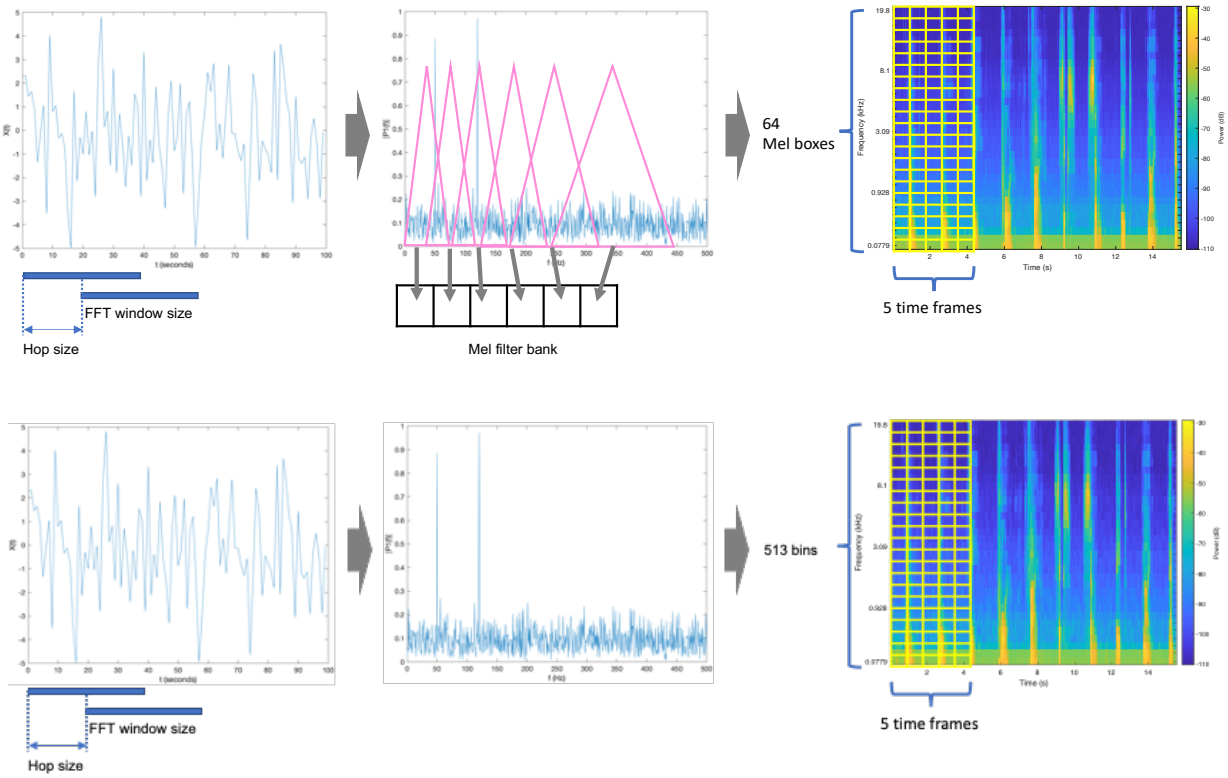

Figure 10 (Top) Schematics of audio data process from time-frequency to log-Mel spectrogram and the input feature vector. (Bottom) Schematics of audio data process from time-frequency to spectrogram and the finer input feature vector of 2565 dimension.

4.4. Adaptive signal processing for the sound data

This study applies the KF and the UKF for state-space estimation for denoising purpose. The dataset provided by the Dataset Provider is processed with MATLAB as stipulated in subsection **4.2.2**. The applied filters are hereunder described.

4.4.1. Kalman filtering for the sound data

The wav files were directly processed, and return the filtered wav files. The "dsp.KalmanFilter System" object provided in MATLAB is an estimator used to obtain a solution for optimal linear filtering recursively. Model of state transition is set two (2), which specify the dimension of a square matrix. The model of the relationship between states and measurement output is set two (2), which dictates a dimension of a row vector with many columns equal to the number of measurements. The covariance of process noise is set at 0.001, which specify a square matrix with the covariance of the white Gaussian process noise and each dimension equal to the number of states. The covariance of measurement noise is 0.1, which specify a square matrix with each dimension equal to the number of states having the covariance of the white Gaussian process noise. The initial value for states is 0, which specify an initial estimate of the states of the model as a column vector with a length equal to the number of states. The initial value for state error covariance is 0.1, which specifies an initial estimate for the state error covariance as a square matrix with each dimension equal to the number of states. The initial system equation with state vector x_k and measurement z_k (scalar) based on the previously-defined properties is expressed as follows,

$$\text{Initial state transition} \quad x_k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0.001 \\ 0.001 \end{pmatrix}, \tag{104}$$

$$\text{Initial measurement} \quad z_k = (1 \quad 1) x_k + (0.1), \tag{105}$$

The script is attached as **Appendix 1** to this report.

4.4.2. Unscented Kalman filtering for the sound data

The wav files were directly processed, and return the filtered wav files. The "unscentedKalmanFilter" class in MATLAB creates an object for online state estimation of a discrete-time nonlinear system using the discrete-time unscented Kalman filter algorithm. Since we cannot assume a system function which describes the Pump sound emission, we decided to apply the default state function mounted in the class. The function is the van del Pol oscillator, which is the widely used nonlinear spring-mass-damper system function written as $\ddot{x} + \mu(x^2 - 1)^2 \dot{x} + x = 0$. Because the Pump is a rotating machine, it could be reasonable to emulate the Pump's physics with such oscillator model if the model considers only very short duration (msec order).

The hyperparameters defined in this study were empirical. The computation of filtering was converged successfully but it should be noted that it did not guarantee this is the best parameter setting for the filter design. The parameter α is set 0.1, which determines the spread of the sigma points around the mean state value. The spread of sigma points is proportional to α, which set to 0.001. Smaller values correspond to sigma points closer to the mean state. The parameter κ, a second scaling parameter, is set zero, which specify the spread of sigma points around the mean state value. The parameter β is set two, which incorporates prior knowledge of the distribution of the state. The configuration of $\beta = 2$ is optimal for Gaussian distribution. The "initialStateGuess" property column

vector was set $(0.1 \quad 0.1)^{\top}$. The measurement noise was set 0.1, and the process noise matrix was set $\mathrm{diag}(0.001\ 0.1)$. The "HasAdditiveMeasurementNoise" property was set false, which means the measurement noise is nonadditive, and the measurement function also specifies how the output measurement evolves as a function of the measurement noise. The script is attached as **Appendix 2** to this report.

4.5. Autoencoder architecture

In this Project, we developed three kinds of autoencoder-decoder neural networks. One is the reproducing model for benchmark, and the other two kinds are for finer dimensions. We conducted our experiment using the autoencoder model by using the `PyTorch` library. The Dataset provider presented the benchmark results with the model developed using the `Keras` library. Accordingly, there is a slight difference between the dataset provider's benchmark and our reproduced benchmarks.

The neural network for the reproducing work follows the dataset provider's model, and it is illustratively shown in **Figure 11**. Hereafter it is called "**Baseline AE**". The encoder network (E(·)) comprises $FC(Input, 64, ReLU)$; $FC(64, 64, ReLU)$ and $FC(64, 8, ReLU)$, and the decoder network (D(·)) incorporates $FC(8, 64, ReLU)$; $FC(64, 64, ReLU)$ and $FC(64, Output,\ none)$ and where $FC(a, b, f)$ mean a fully-connected node with input neurons, b output neurons, and activation function f. No drop-out was applied through this Project. The ReLU denotes the Rectified Linear Units activation function. In this case, the input vector is 320 dimensions. The network is trained with the Adams optimization techniques for 50 epochs.

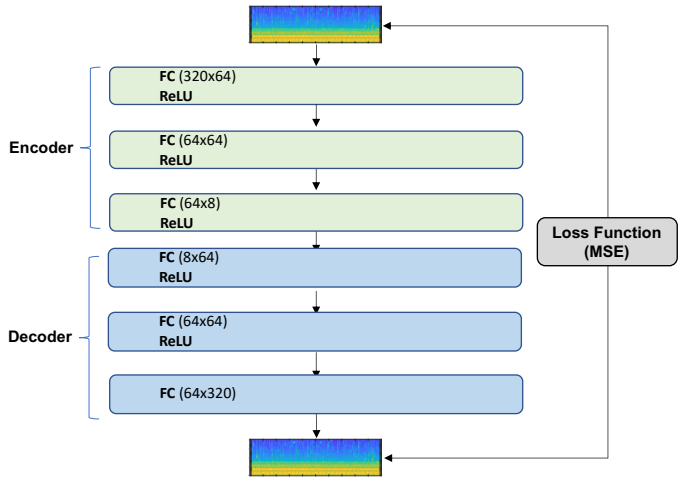

Figure 11 Illustration of the model architecture of autoencoder-and-decoder-based deep neural network for the reproductive experiment as the benchmark (Baseline AE).

One of the proposed methods equipping finer input dimension is illustratively shown in **Figure 12**. Hereafter it is called "**Finer-Input AE**". The encoder network (E(·)) comprises $FC(Input, 128, ReLU)$; $FC(128, 64, ReLU)$ and $FC(64, 8, ReLU)$, and the decoder network (D(·)) incorporates $FC(8, 64, ReLU)$; $FC(64, 128, ReLU)$ and $FC(128, Output,\ none)$. No drop-out was applied. In this case, the input vector is 2565 dimensions. The network is trained with Adams optimization techniques for ten epochs.

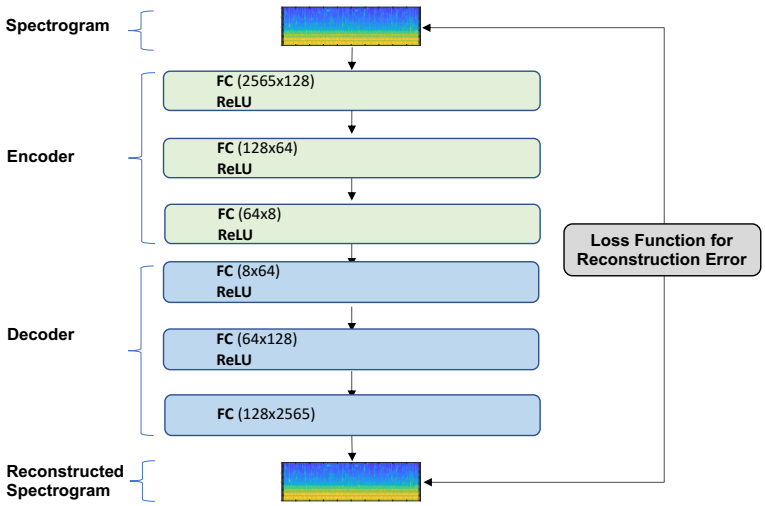

Figure 12 Illustration of the model architecture of autoencoder-and-decoder-based deep neural network with the finer input dimension (Finer-Input AE).

Another type of the proposed methods equipping finer input dimension is illustratively shown in **Figure 13**. Hereafter it is called "**Finer-Hidden AE**". This architecture has a larger latent node compared to the Finer AE. The encoder network (E(·)) comprises $FC(Input, 128, ReLU)$; $FC(128, 64, ReLU)$ and $FC(64, 2565, ReLU)$, and the decoder network (D(·)) incorporates $FC(2565, 64, ReLU)$; $FC(64, 128, ReLU)$ and $FC(128, Output,\ none)$. No drop-out was applied. The input vector was 2565 dimensions. The network is trained with the Adams optimization techniques for ten epochs.

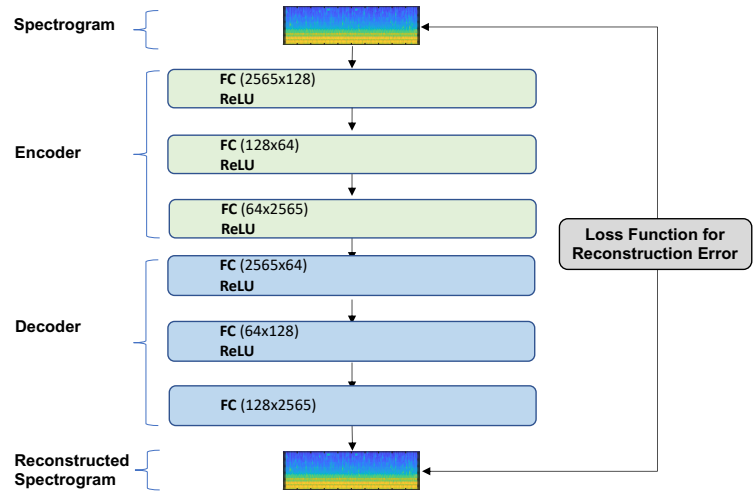

Figure 13 Illustration of the model architecture of autoencoder-and-decoder-based deep neural network with higher dimensions in the hidden layer for sparse modelling (Finer-Hidden AE).

In this report, we conducted various initial data analysis to understand the characteristics of the sound dataset and explore potential improvement approach. Neural networks other than autoencoder-decoder neural networks, such as convolutional neural networks and generative adversary neural networks, and other anomaly detection methods are described in **4.6** in line with its results.

The six types of loss functions mentioned in section **3.1.4** were implemented and tested; The Mean Squared Error ("**MSE**"), MSE with L2 regularization term ("**MSE + Ridge**"), MSE with L1

regularization term ("**MSE + Lasso**"), the loss function with Kullback-Leibler Divergence ("**MSE + KLD**"). The Mean Absolute Error ("**MAE**"), and the Huber loss ("**Huber**"). These loss functions were implemented with the libraries provided in `PyTorch`.

4.6. Initial data analysis

In order to understand the data characteristics and seek practical approaches for enhancing its anomaly detection, we firstly conducted exploratory research for the Dataset with existing machine learning approaches.

Table 4 Tasks and purpose of the initial data analysis.

Corresponding Section in this report	Task	Purpose
6.6.1	Data Visualization and statistics description	Exploratory studies to understand the characteristics of the sound data
6.6.2	Dimension reduction with PCA and *t*-SNE	Embed and visualize the data in low dimension to check if classic statistical approaches are applicable
6.6.3	Autoencoder-decoder neural network (reproductive work)	Conduct a reproductive work following the data provider's condition and acquire baseline result for our research

4.6.1. Data visualization and statistics description

Figure 14 shows frequency and log-Mel spectrogram in the time domain figures of one of the wav files of 6 dB SNR in the Dataset. A pump in normal condition operation contains high-intensity components at the frequency band of 50 Hz to 1 kHz. At high-frequency band, there are observed randomly scattered components supposed to be environment noise. In contrast, a pump in anomalous condition showed the sudden change of sound, which implies pump trip trouble.

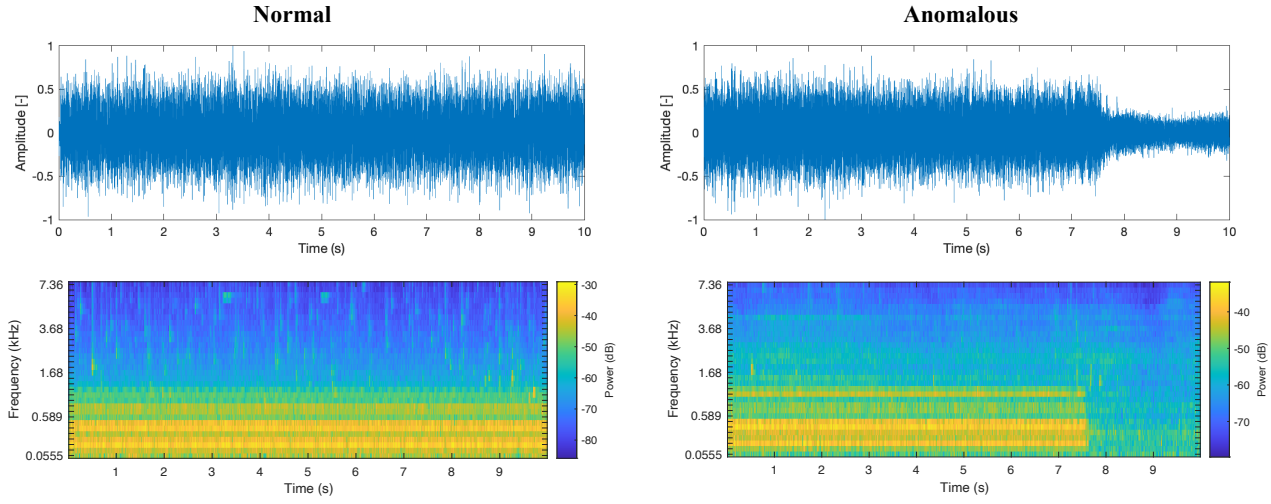

Figure 14 Examples of the normalized amplitude of the pump ID: 06 at SNR 6dB, waveform frequency in the time domain (top row) and corresponding power spectrogram (bottom row) on normal condition (left column) and anomalous condition (right column).

Likewise, **Figure 15** shows frequency and power spectrogram in the time domain figures of one of the wav files of -6 dB SNR in the Dataset. In normal condition, the frequency band in the range of 50 Hz to 1 kHz corrupted, and its boundaries become unclear compare to that of the sound data at the SNR of 6 dB. The component in the broad domain of high frequency is highlighted due to its low SNR. The anomalous condition data, in this case, shows hunching every two seconds. The anomalous condition visualized in the time-frequency waveform figure is ambiguous, but the log-Mel spectrogram seems successfully highlighted the transition of sound components, which obviously different from the corresponding normal condition.

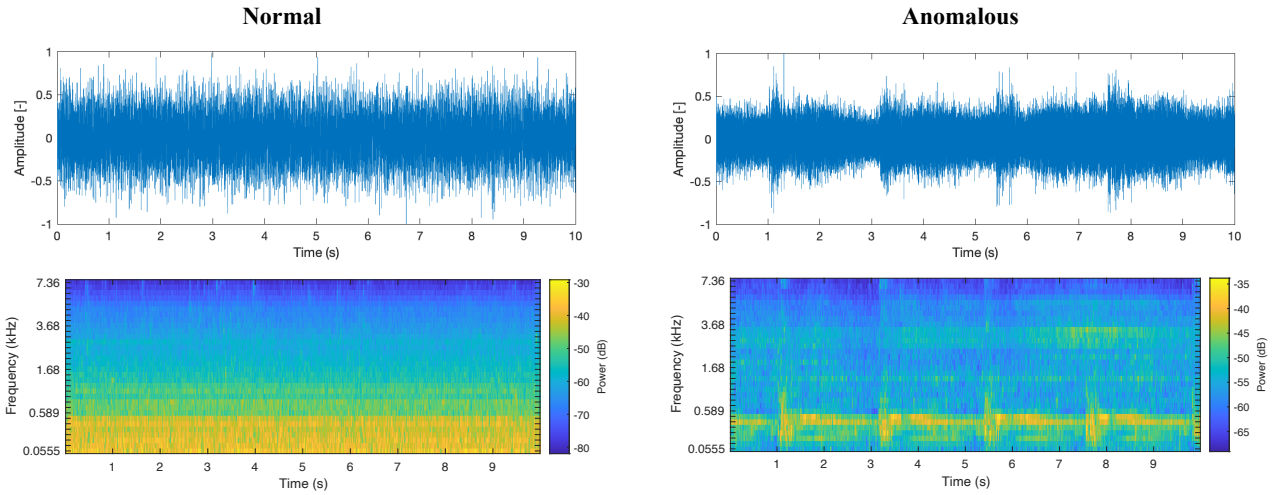

Figure 15 Examples of the normalized amplitude of the pump ID: 06 at -6 dB SNR, waveform frequency in the time domain (top row) and corresponding power spectrogram (bottom row) on normal condition (left column) and anomalous condition (right column).

It should be noted that in the Dataset, the data is labelled only normal and anomaly. No further description of these anomalous conditions was annotated to the data. Hence the anomalous condition needs to be detected as outlier data from the normal condition.

4.6.2. Dimension reduction with PCA and t-SNE

Our initial data analysis utilizes the features obtained by the log-Mel spectrogram transformation and reduces the high-dimensional data to two-dimensional space for visualization. The PCA was tested using the library `scikit-Learn` (version 0.22.1). **Figure 16** shows plots of normal condition and anomalous condition data in two-dimensional space reduced using PCA from the 320-dimension features obtained by log-Mel spectrogram. The Pump sound file was 00000000.wav and the first 100 samples out of 306 samples for 10 [sec] are projected. The Pump at normal conditions and anomalous conditions at 6 dB SNR are apparently projected to different clusters in two-dimensional space. In contrast, both normal and anomalous condition sound data are distributed onto similar regions, despite some clustering. The result implies that es the data of high SNR can be conducted its anomaly detection by conventional clustering methods such as the k-mean clustering, but low SNR data needs to be scrutinized by other methods which can embrace non-linearity and reflects high-dimension information for detection.

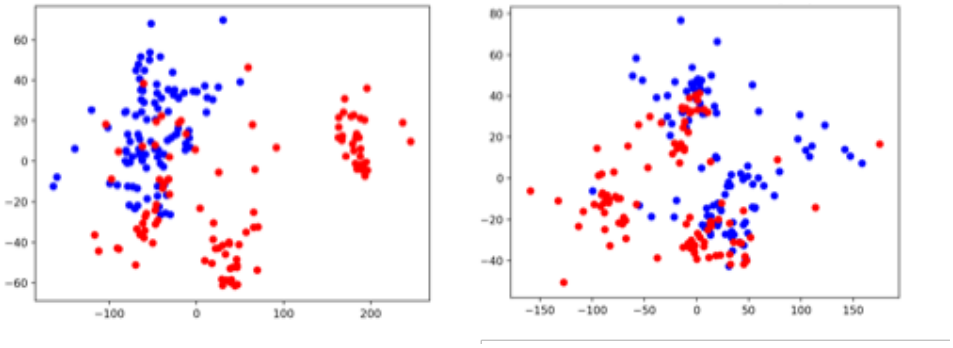

Figure 16 The Pump ID06 operation sound data at SNR of 6dB (left) and at SNR of -6 dB (right). Embedded the 320-dimension log-Mel spectrogram features into the estimated two-dimensional space by PCA. The symbols of blue and red represent the normal condition and anomalous condition, respectively.

We also applied *t*-distribution Stochastic Neighbourhood Embedding ("***t*-SNE**") to reduce the dimension [54]. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. **Figure 17** shows the plots of normal condition and anomalous condition data in two-dimensional space embedded by *t*-SNE from the 320-dimension features obtained by log-Mel spectrogram. *t*-SNE was implemented with the library in `scikit-Learn` (version 0.22.1). The data at SNR of 6 dB showed the clearer cluster than the plot of the data at SNR of -6 dB. Also, the embedded by *t*-SNE showed the clearer than the PCA shown in **Figure 16**. The data at SNR of -6 dB showed that some of anomalous condition can be demarcated, but most of the data had ambiguous boundaries against normal condition. *t*-SNE shows good performance of anomaly detection for data with high-SNR, but noisy data requires other approaches to represent the complex data structure.

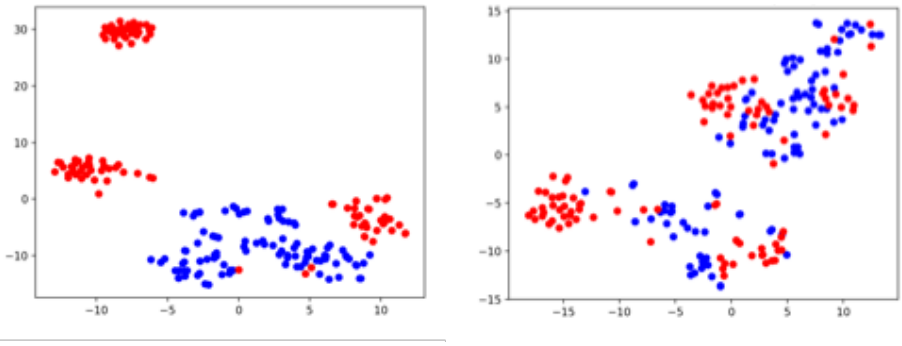

Figure 17 The Pump ID06 operation sound data at SNR of 6dB (left) and at SNR of -6 dB (right). Embedded 320-dimension log-Mel spectrogram features into the estimated two-dimensional space by *t*-SNE. The symbols of blue and red represent the normal condition and anomalous condition, respectively.

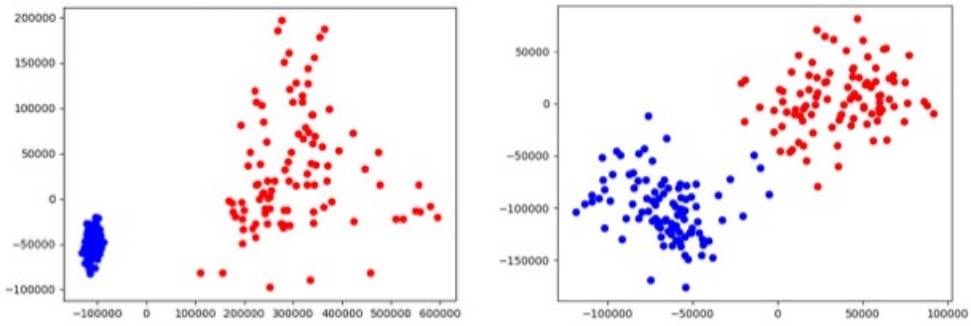

Figure 18 The Pump ID06 operation sound data at SNR of 6dB (left) and at SNR of -6 dB (right). Embedded the 2565-dimension log-Mel spectrogram features into the estimated two-dimensional space by PCA. The symbols of blue and red represent the normal condition and anomalous condition, respectively.

For the finer input of 2565 dimension vector, **Figure 18** and **Figure 19** show the embedded plot using PCA and *t*-SNE, respectively. Likewise, the Pump sound file was 00000000.wav and the first 100 samples out of 306 samples for 10 [sec] are projected. In this resolution, we can observe in both PCA and *t*-SNE that the sound data at SNR of -6dB can be segregated into two clusters. These plots indicate that the finer resolution, before combining to the 64-dimension log-Mel bin, possesses the sufficient information to separate the difference between normal and anomalous conditions. In spite of the affirmative results, it should be noted that this successful embedding was observed for the portion of the dataset. Not all of the sound data show such a clear difference and therefore we still have demand of applying the autoencoder for anomaly detection, rather than clustering methods.

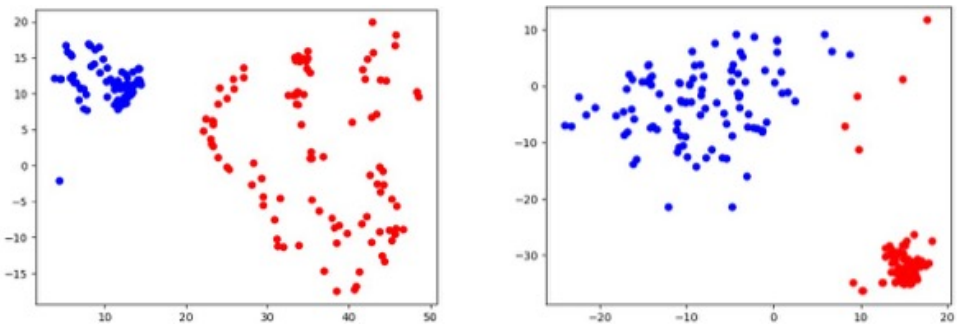

Figure 19 The Pump ID06 operation sound data at SNR of 6dB (left) and at SNR of -6 dB (right). Embedded 2565-dimension log-Mel spectrogram features into the estimated two-dimensional space by *t*-SNE. The symbols of blue and red represent the normal condition and anomalous condition, respectively.

4.6.3. Autoencoder-decoder architecture for reproductivity check

The results of the reproduce experiment and the dataset provider's baseline AUC are summarized in **Table 5** and **Figure 20**. In this Project, we replicated the pump dataset work for all machine IDs and SNRs in the architecture of the Baseline AE. The AUCs averaged over ten training runs with independent initializations, and the sample standard deviation was computed based on the ten results. The deviation was mainly derived from the random sampling of normal sound data to develop the test datasets. In each independent experiment, the sampling was conducted, and the extent of an anomaly in normal sound data was heterogeneous. Thus, the combination of normal data in the test dataset can vary the difficulty of anomaly detection, resulting in performance variation.

We confirmed that the noisy data tend to the worse AUC. In other words, our result supported the benchmark result and the trend that noisy data exacerbates the detection performance. The pump ID06 shows a sharp drop off of the AUC value as the sound contaminated more severely. It performed the AUC of 0.9281 at SNR of 6dB and performed the AUC of 0.6518 at SNR of -6dB. We mainly use the pump model ID06 for our experiments because evaluating the proposed method effect could be easier than other model IDs.

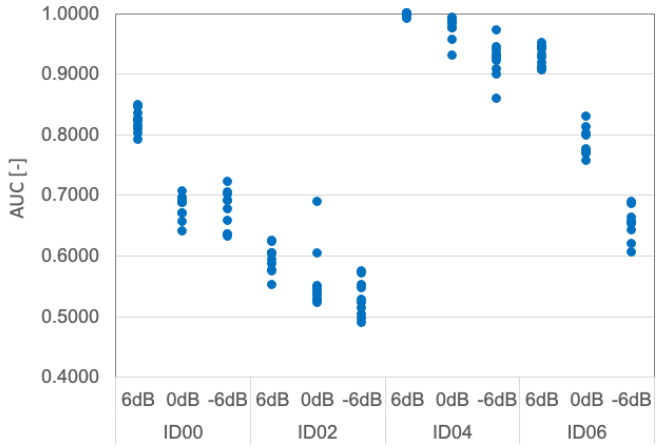

Figure 20 The results of AUC values in the reproductive work using the Baseline AE for each pump model IDs and SNRs.

Table 5 Comparison of AUC shown in **Figure 20** and the baseline AUC values presented by the Dataset provider.

Model ID	**Input SNR**	**Reproduced Results**		**Baseline** [44].
		AUC [-]	**STD DEV [-]**	**AUC [-]**
ID00	6dB	0.8212	0.0181	0.84
	0dB	0.6792	0.0199	0.65
	-6dB	0.6741	0.0328	0.58
ID02	6dB	0.5938	0.0226	0.45
	0dB	0.5576	0.0513	0.46
	-6dB	0.5293	0.0304	0.52
ID04	6dB	0.9979	0.0028	0.99
	0dB	0.9753	0.0185	0.95
	-6dB	0.9226	0.0302	0.93
ID06	6dB	0.9281	0.0168	0.94
	0dB	0.7854	0.0235	0.76
	-6dB	0.6518	0.0257	0.61

4.7. Results and discussion in our proposed approach

In this section, we summarize the result of our proposed methods and discussion.

4.7.1. The Finer-Input AE with various loss functions

Firstly, the Finer AE has applied for the pump ID06 dataset in order to check if such finer resolution shows a similar performance tendency. The dataset was not processed, and the loss function was MSE. The AUCs averaged over ten training runs with independent initializations, and the sample standard deviation was computed based on the ten results. The result is shown in **Figure 21** and **Table 6**. It was confirmed that the noisier sound data showed lower AUC values.

Interestingly, the sound at SNR of -6dB performed better than the Benchmark AE, AUC 0.6518, as reported in **4.6.3**. Meanwhile, the sound at SNR of 6dB performed 0.9008, which was slightly worse than the Benchmark AE of AUC 0.9281. Transforming to log-Mel spectrogram from a time-frequency domain acquired by STFT is supposed to have pertinent information for pump sound anomaly detection, but the information may lose when contaminated with noise.

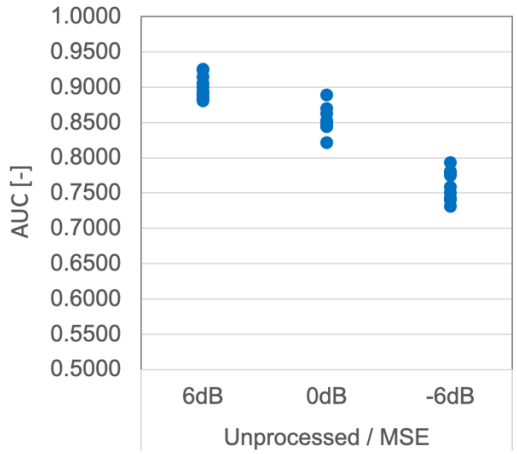

Figure 21 The results of AUC values for pump ID06 at SNR of 6dB, 0dB and -6dB tested with the Finer-Input AE. The sound data was unprocessed, and the loss function was MSE.

Table 6 The results of AUC shown in **Figure 21**.

SNR	AUC [-]	STD DEV [-]
6dB	0.9008	0.0161
0dB	0.8526	0.0180
-6dB	0.7630	0.0210

Figure 22 shows AUC results for the various pre-processing methods and loss functions in the Finer AE neural network on the sound data of the pump ID: 06 at SNR of -6dB. The values in **Table 7** represent the mean and sample standard deviation confidence intervals based on ten independent test iterations to show a calibrated performance of the results. The hyperparameters as a coefficient for the penalized term in MSE+Lasso, MSE+Ridge and MSE+KLD were 0.01.

The proposed schematics using UKF for data preprocessing and Huber loss function showed improvement of AUC from 0.7630 to 0.8145. The data preprocessed by KF also showed an improved AUC of 0.8114. Although both the UKF-processed and KF-processed sound dataset showed better AUC when used the Huber loss function, these pre-processing showed different performance when applied MSE. The KF-processed data with MSE showed better improvement than the UKF-processed data with MSE.

Another implication was that the preprocessing affected its variance when MAE and Huber were used as a loss function. The unprocessed data with MAE and Huber loss functions showed the AUC's standard deviation of 0.0146 and 0.0279, respectively. On the other hand, the KF-processed and UKF-processed data showed the larger AUC's standard deviation of 0.0552, 0.0365, 0.0362 and 0.0366, respectively. The results implied that the data pre-processing by the adaptive filters impact anomaly

detection performance using a neural network. In our experiment, the robust estimation using the MAE and Huber loss function performed superior to these of sparsity. Hence the loss function should be designed in line with the property of the applied adaptive filters.

The result that the UKF-processed data showed better performance when used robust estimators of the MAE and the Huber loss function than using other loss functions implies that the UKF algorithm can weight the outlier data. If the outlier is weighted, the MSE can by nature perform worse compared to the unprocessed data and the KF-processed data. Instead, the MAE and the Huber loss, known as robust estimators, can work better.

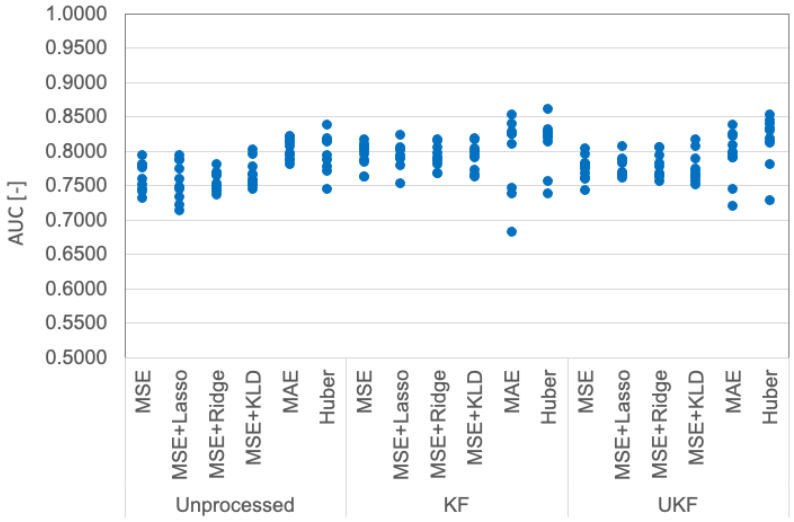

Figure 22 The summary of AUC values for the various sound data pre-processing methods and loss functions in the Finer-Input AE on the sound data of the pump ID06 at SNR of -6 dB.

Table 7 The AUCs are shown in **Figure 22**.

Pre-processing	**Loss Function**	**AUC [-]**	**STD DEV [-]**
Unprocessed	MSE	0.7630	0.0210
	MSE+Lasso	0.7556	0.0287
	MSE+Ridge	0.7547	0.0145
	MSE+KLD	0.7663	0.0191
	MAE	0.8014	0.0146
	Huber	0.7968	0.0279
KF	MSE	0.7923	0.0188
	MSE+Lasso	0.7933	0.0187
	MSE+Ridge	0.7908	0.0171
	MSE+KLD	0.7919	0.0199
	MAE	0.7964	0.0552
	Huber	0.8114	0.0365
UKF	MSE	0.7741	0.0180
	MSE+Lasso	0.7776	0.0143
	MSE+Ridge	0.7775	0.0177
	MSE+KLD	0.7750	0.0218

MAE	0.7929	0.0362
Huber	0.8145	0.0366

In order to examine the impact of input feature vector resolution, these pre-processing and loss functions of MSE and Huber were implemented by using the Baseline AE architecture. The input feature vector was 320 dimensions as stated in the reproduction experiment, based on the log-Mel spectrogram.

In contrast to the results acquired with the Finer AE architecture, it was observed that both KF-processed and the UKF-processed data with the loss function of Huber performed worse AUC compared to the unprocessed data with MSE loss function. The log-Mel spectrogram lost information necessary for anomaly detection, but the tendency was different for the pre-processing data type.

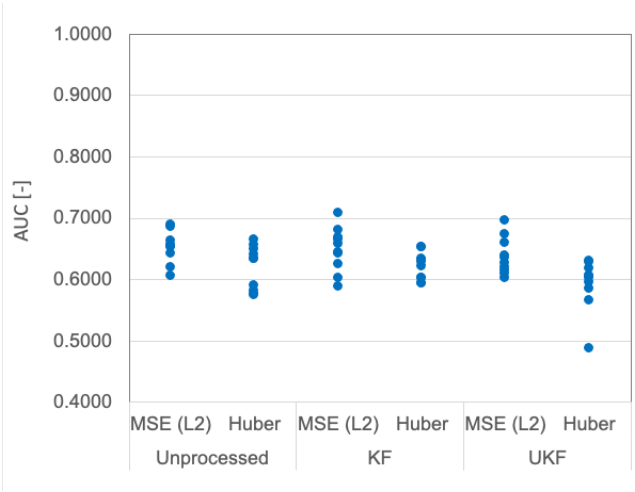

Figure 23 The results of AUCs with the Baseline AE for unprocessed, KF-processed and UKF-processed dataset. The loss functions of MSE and Huber were applied. Pump ID06 at SNR of -6 dB.

Table 8 AUCs shown in **Figure 23**.

Pre-processing	**Loss Function**	**AUC [-]**	**STD DEV [-]**
Unprocessed	MSE (L2)	0.6518	0.0257
	Huber	0.6199	0.0356
KF	MSE (L2)	0.6475	0.0359
	Huber	0.6206	0.0226
UKF	MSE (L2)	0.6370	0.0302
	Huber	0.5918	0.0414

The learning curve in the training step of the Finer-AE architecture with the Huber loss function is shown in **Figure 24**. The UKF-processed data always showed a lower reconstruction error of the training error than the unprocessed and KF-processed dataset.

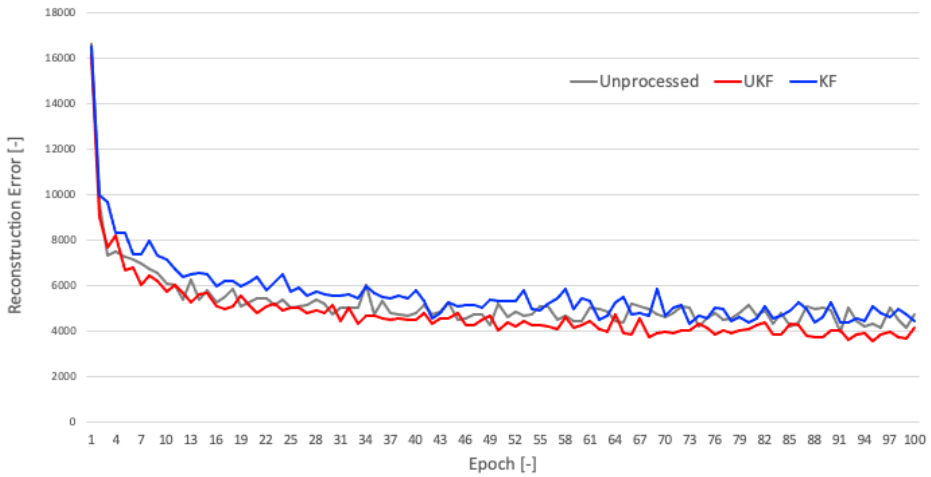

Figure 24 The learning curve of Huber loss function for unprocessed data, KF-processed and UKF-processed data.

The UKF-processed data with the Huber loss as the autoencoder loss function was tested for training epochs of 10, 50 and 100. The more training epoch showed the smaller variance and the lower AUC. This result implies that the larger epochs decreased the training loss but increased generalization error due to overfitting.

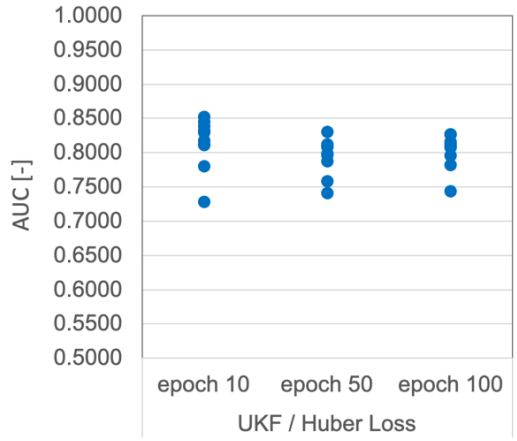

Figure 25 The results of AUC evaluated by various epochs with the Finer-Input AE. The sound data was processed with UKF, and the loss function was Huber.

Table 9 The results of AUC shown in **Figure 25**.

Epochs	AUC [-]	STD DEV [-]
10	0.8145	0.0366
50	0.7951	0.0269
100	0.8030	0.0247

4.7.2. Finer-Hidden AE

In the previous section, we acquired the implication that sparse modelling is the key to this anomaly detection. Hence, we extended the Finer-AE architecture to represent sparse information, employing

a higher dimension in its latent node. The Finer-Hidden AE, as introduced in section 4.5, has the same size of the latent node as the input node, 2565 dimensions. The Finer-Hidden AE was tested on the unprocessed data, the KF-processed data and the UKF-processed data. MSE and Huber loss function was used for comparative experiments.

Figure 26 shows the AUCs of the experiments. Compared to the Finer AE architecture results shown in **Figure 22**, the AUC of the KF and UKF processing with the Huber loss function showed better performance. This result also indicated that the pre-processing method can effect on the appropriate network architecture.

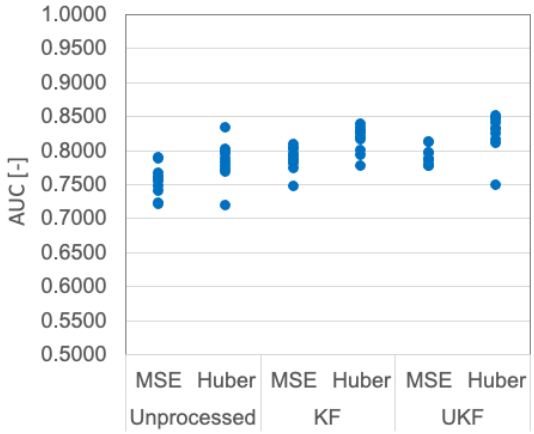

Figure 26 The results of AUC with the Finer-Hidden AE.

Table 10 The results of AUC in **Figure 26**.

Pre-processing	Loss Function	AUC [-]	STD DEV [-]
Unprocessed	MSE	0.7542	0.0233
	Huber	0.7827	0.0294
KF	MSE	0.7868	0.0177
	Huber	0.8153	0.0197
UKF	MSE	0.7903	0.0129
	Huber	0.8241	0.0300

4.7.3. Example of good and bad images

A test sample set of 102 normal-condition .wav files and the same number of anomalous-condition .wav files at SNR of -6dB was examined by the autoencoder neural network Huber loss function. The data was unprocessed. The reconstruction errors for the test sample set are shown in **Figure 27**. The horizontal axis shows normal-condition files numbered from 1 to 102 and anomalous-condition files numbered from 103 to 204. Since the architecture was trained with normal-condition sound data, anomalous-condition data cannot be reconstructed preciously and results in a relatively higher reconstruction error score.

One way to define an appropriate threshold for anomaly detection is to use an F-score, which maximizes a harmonic average of detection ratio r_1 and coverage r_2,

$$\text{F-Score} = \frac{2r_1 r_2}{r_1 + r_2}. \tag{107}$$

Based on the criteria, we can define the threshold of F-Score as 3650 for the dataset.

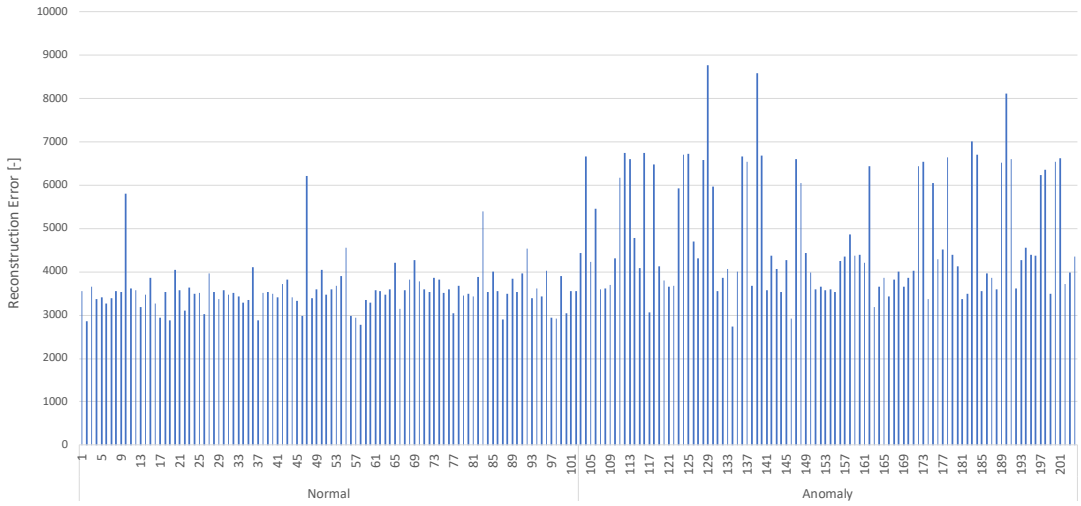

Figure 27 The results of Reconstruction Error.

Among the normal-condition dataset, successfully detected normal-condition with the minimum reconstruction error of 2848 is 00000659.wav. On the other hand, wrongly detected as an anomalous condition with the highest reconstruction error of 6214 was 00000038.wav. These sound data were visually shown in **Figure 28**. The data of 00000659.wav showed momentary loud sound at 4 seconds elapsed and.

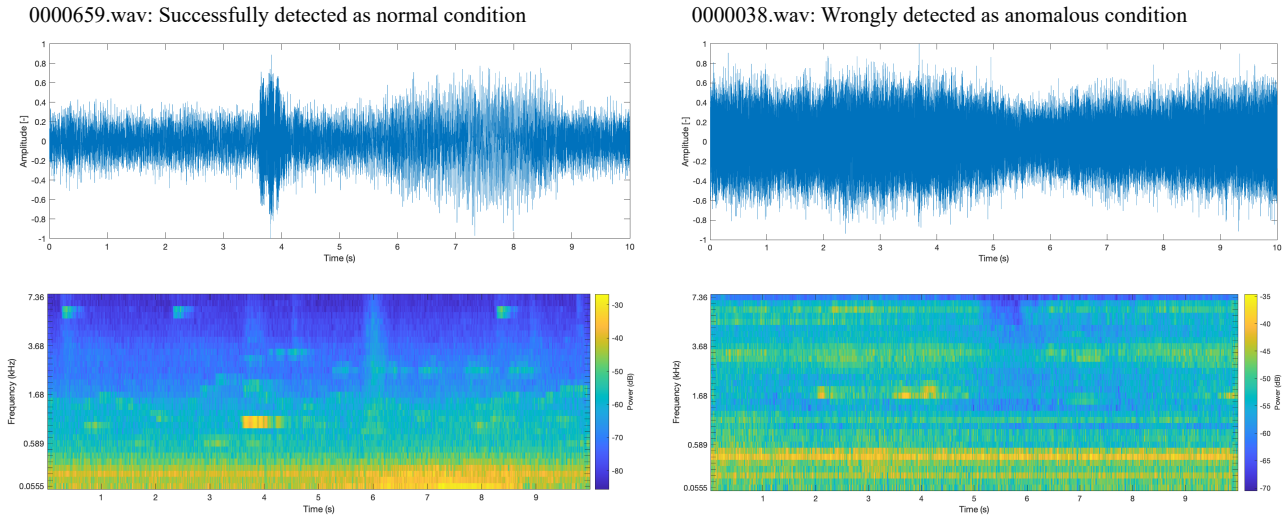

Figure 28 Examples of waveforms and power spectrum images of the normal condition data for pump ID 06 at SNR -6dB, (left) successfully detected and (right) wrongly detected.

Likewise, among the anomalous-condition dataset, successfully detected anomalous-condition with the highest reconstruction error of 6736 is 00000077.wav. The wrongly detected as an anomalous condition with the lowest reconstruction error of 2738 was 00000005.wav. In the case of 00000077.wav, somewhat periodical peaks every two seconds can be observed. This anomalous

periodical information enabled the autoencoder to detect an anomaly. In contrast, the case of 00000005.wav shows that signal information is covered with background noise.

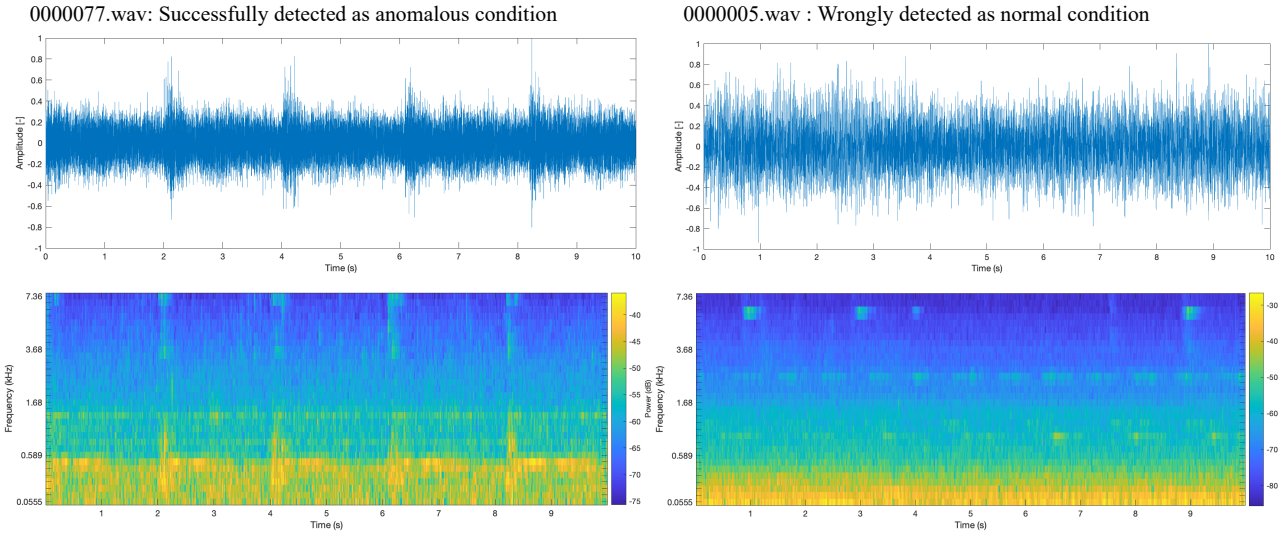

Figure 29 Examples of waveforms and power spectrum images of the anomalous condition data for pump ID 06 at SNR -6dB, (left) successfully detected and (right) wrongly detected.

4.7.4. Computation costs

Table 11 summarizes the computation time tested in our experiment environment. The model was the Finer-Input AE. The time was measured using the Python time module. The performance time returns the value (in fractional seconds) of a performance counter. It does include time elapsed during sleep and is system-wide. The processing time returns the value (in fractional seconds) of the sum of the system and user CPU time of the current process. It does not include time elapsed during sleep. The thread time returns the value (in fractional seconds) of the sum of the system and user CPU time of the current thread. It does not include time elapsed during sleep.

We executed five independent runs for all the patterns of loss functions and pre-processing and took the median of the results. We observed that the MSE and the robust estimation of the MAE and the Huber had a similar extent of computation complexity, so that we found no trade-off in the robustness and the computational efficiency.

Table 11 Computation costs.

Loss Function	Pre-Processing	Performance [sec]	Process [sec]	Thread [sec]
MSE	Unprocessed	573	539	745
	KF	560	526	735
	UKF	576	539	745
MSE+Ridge	Unprocessed	1122	1077	1322
	KF	799	757	971
	UKF	1038	1000	1207
MSE+Lasso	Unprocessed	560	518	745
	KF	586	548	756

	UKF	579	541	756
MSE+KLD	Unprocessed	759	713	1006
	KF	748	702	1004
	UKF	766	718	1033
MAE	Unprocessed	546	507	733
	KF	553	517	743
	UKF	572	537	746
Huber	Unprocessed	545	508	727
	KF	558	523	735
	UKF	568	536	737

4.7.5. Comparison experiment with ToyADMOS dataset

This section demonstrated a comparative experiment by applying our proposed methods to other sound datasets designated for anomaly detection testing. We applied our proposed approaches of the Finer-Input AE for another dataset of industrial machines [5]. We used the ToyCAR datasets in this experiment. **Table 12** summarizes the results of AUC values of our approach and the dataset provider's baseline results. For this dataset, the best combination found in our proposed approaches (UKF+Huber loss) did not perform as we demonstrated with the MIMII dataset. Instead, the combination of UKF+MSE worked better than that of Unprocessed+MSE. These results indicated that the property of the sound dataset impacts the performance. Hence, further study of the sound source and noise property is required for the actual application.

Table 12 AUC evaluated for ToyADMOS dataset by using the proposed methods.

Pre-processing	**Loss Function**	**AUC [-]**	**AUC [-] in the literature [5]**
Unprocessed	MSE	0.6810	0.874
	Huber	0.6736	N/A
KF	MSE	0.6858	N/A
	Huber	0.6657	N/A
UKF	MSE	0.7176	N/A
	Huber	0.6861	N/A

5. CONCLUSIONS

In this Project, we set the main objective to improve the accuracy in classifying normal and anomalous conditions of industrial machine based on noisy sound data. We proposed to utilize the adaptive digital filters for sound data pre-processing and to design an autoencoder architecture. The proposed approaches were demonstrated by using the real-world industrial machinery sound dataset. To our knowledge, few studies are focusing on the relationship between the data pre-processing and autoencoder architecture, and therefore we demonstrate novel insights into anomaly detection. As a result, we had the achievement of improving the anomaly detection performance. The key takeaways are summarized as follows:

1. The proposed combination of the UKF as the sound data pre-processing and the Huber loss function as the loss function of the autoencoder with the finer input vector proved the improvement of the AUC value for the noisy data of the pump (ID: 06 at SNR -6dB) to 0.8241 (±0.0300), which is by 0.0699 improvements from the baseline result;

2. Our results indicated that the robust estimation is suitable for extracting meaningful information from the noisy sound data we used in this Project. Indeed, the MAE and the Huber loss function outperformed compared to other tested loss functions; and

3. We observed that the optimal type of the autoencoder architecture, such as the input vector's dimension and the loss function, is affected by adaptive digital filters for the data pre-processing. The results experimentally implied that the architecture of the autoencoder should be designed and optimized as a single system in conjunction with the design of the adaptive digital filters for anomaly detection in noisy sound data.

It could be interesting to explore other adaptive digital filters such as particle filter in future work. It accommodates non-gaussian noise, and therefore it may help understand details in the mechanism of how the filtering effect the training process of an autoencoder.

6. REFERENCES

1. CHANDOLA, Varun *et al*. Anomaly detection: a survey. *ACM Computing Surveys*. vol. 41, no. 3, article 15, p. 1–14, 2009.
2. CHALAPATHY, Raghavendra and Sanjay CHAWLA. Deep learning for anomaly detection: a survey. 2019. arXiv:1901.03407v2.
3. KAWAGUCHI, Yohei and Takashi ENDO. How can we detect anomalies from subsampled audio signals?. *2017 IEEE International Workshop On Machine Learning For Signal Processing*, Tokyo: IEEE, 2017.
4. KAWAGUCHI, Yohei. Anomaly detection based on feature reconstruction from subsampled audio signals. *26th European Signal Processing Conference (EUSIPCO)*. p. 2538-2542, 2018.
5. KOIZUMI, Yuma *et al*., Unsupervised Detection of Anomalous Sound Based on Deep Learning and the Neyman–Pearson Lemma, *IEEE/ACM Transactions On Audio, Speech, And Language Processing*, IEEE, vol. 27, no. 1, p. 212 – 224, 2019.
6. ROSENBLATT, Frank. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Phycological Review*, Vol. 65, No. 6, 1958, p. 386 – 408.
7. FUKUSHIMA, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*. vol. 36, p. 193 – 202, 1980.
8. RUMELHART, David. Learning representations by back-propagating errors, *Nature*, vol. 323, p. 533 – 536, 1986.
9. LECUN, Yann *et al*., Backpropagation applied to handwriting zip code recognition, *Neural Communication*, vol. 1, p. 541 – 551, 1989.
10. GLOROT, Xavier. Antoine BORDES and Yoshua BENGIO. Deep sparse rectifier neural networks. *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS) 2011*. 2011. p. 315 – 323.
11. NAIR, Vinod and Geoffrey E. HINTON. Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning*. Haifa, Israel, 2008.
12. DUCHI, John, Elad HAZAN, and Yoram SINGER. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*. vol. 12, p. 2121 – 2159, 2011.
13. KINGMA, D. P. and J. BA. Adam: A method for stochastic optimization, *International Conference of Learning Representations 2015*, arXiv preprint arXiv:1412.6980, 2014.
14. LECUN, Yann *et al*., Gradient-based learning applied to document recognition, *Proceedings of the IEEE*. IEEE, vol. 86, issue 11, p. 2278 – 2324, 1998.
15. KRIZHEVSKY, Alex, Ilya SUTSKEVER, and Geoffrey E. HINTON. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*. vol. 25, no. 2, 2012.
16. GOODFELLOW, Ian J. *et al*. Generative adversarial nets, arXiv:1406.2661v1.
17. RADFORD, Alec, Luke METZ and Soumith CHINTALA, Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks, *CoRR 2015*, 2015.
18. MIKOLOV, Tomas *et al*. Recurrent neural network based language model. *Interspeach 2010*, ISCA, Chiba, Japan, p.1045-1048, 2010.

19. GERS, A. Felix, Nicol N. SCHRAUDOLPH and Jurgen SCHMIDHUBER. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*. vol. 3, p. 115 – 143, 2002.
20. SCHÖLKOPF, Bernhard *et al.* Support vector method for novelty detection, *Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99)*, CO, USA., p. 582-588, 2000.
21. HU, Qiao *et al*. Fault diagnosis of rotating machinery based on improved wavelet package transform and SVMs ensemble. *Mechanical Systems and Signal Processing*. vol. 21, p. 688-705, 2007.
22. BREUNIG, M. Markus *et al*., LOF: Identifying density-based local outliers. *International conference on management of data*. TX, USA, p. 1–12, 2000.
23. GU, Xiaoyi, Leman AKOGLU, and Alessandro RINALDO. Statistical analysis of nearest neighbour methods for anomaly detection. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. Vancouver, Canada, 2019.
24. SAKURADA, Mayu and Takehisa YAIRI. Anomaly detection using autoencoders with nonlinear dimensionality reduction. *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. QLD, Australia, p. 4 – 11, 2014,
25. SCHLEGL, Thomas *et al*. Unsupervised anomaly detection with generative adversarial networks to guide maker discovery, *the proceedings of Information Processing in Medical Imaging 2017*, NC, USA, pp. 146 – 157, 2017.
26. RUFF, Lukas *et al*. Deep one-class classification, *Proceedings of the 35th International Conference on Machine Learning*, . Stockholm, Sweden, p. 4393 – 4402, 2018.
27. MARCHIE, Erik *et al*. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks, *International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2015*, p.1996 – 2000, 2015.
28. HAYKIN, Simon. *Adaptive Filter Theory*. Essex: PEASON, 5 edition, 2001. ISBN 978-0-273-76048-3.
29. KALMAN, Rudolf E. New approach to linear filtering and prediction problem, Journal of Basic Engineering, *Transactions of the ASME- Journal of Basic Engineering*. vol. 82D, no. 1, p. 35-45, 1960.
30. ADACHI, Shuichi and Ichiro MARUTA. *カルマンフィルタの基礎 (Fundamentals of Kalman Filter)*. Tokyo: Tokyo Denki University Press, 2012. ISBN 978-4-501-32890-0.
31. JULIER, Simon *et al*. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transaction on Automatic Control*. vol. 45, no. 3, 2000.
32. DONOHO, L. David and P. B. STARK. Uncertainty principle and signal recover. *SIAM J. Appl. Math*. vol.49(3), p. 906-931, 1989.
33. OLSHAUSEN, B. and F. J. DAVID. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1 ?, *Vision Research,* Vol.37, no. 23, p. 3311-3325, 1997.
34. KULLBACK, S. and R. LEIBLER, R. A. On Information and Sufficiency, *Annals of Mathematical Statistics,* Vol.22, No. 1, p. 79-86, 1951.
35. OKATANI, Takayuki. *深層学習(Deep Learning)*, Tokyo: Kodansha, 2015. ISBN 978-4-06-152902-1.

36. ARPIT, Devansh *et al*. Why regularization auto-encoders learn sparse representation?, *Proceedings of the 33rd International Conference on Machine Learning,* New York, NY, USA,2016.
37. VINCENT, Pascal *et al*. Extracting and composing robust features with denoising autoencoders, *Proceedings of the 25th international conference on Machine Learning*, 1096-1103, 2008.
38. RIFAI, Salah *et al*. Contractive Auto-Encoder: Explicit Invariance During Feature Extraction, *Proceedings of the 28th international conference on Machine Learning*, 833-840, 2011.
39. MARONNA, A. Ricardo *et al*. Robust statistics: theory and methods, New York: Wiley, 2006. ISBN 978-1-119-21467-0.
40. FUJISAWA, Hiromitsu and Shinto EGUCHI. Robust parameter estimation with a small bias against heavy contamination. *Journal of Multivariate Analysis*. vol.99, p.2053-2081, 2008.
41. GEMMEKE, F. Jort *et al*. Audio set: an ontology and human-labelled dataset for audio events. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. LA, USA, 2017.
42. DEKKERS, Gert *et al*. the SINS database for detecting daily activities in a home environment using an acoustic sensor network. *Detection and Classification of Acoustic Scenes and Events*. Munich, Germany, 2017.
43. DONGO, Yul and Dong Yun IL. Residual error based anomaly detection using auto-encoder in SMD machine sound, *Sensors*, vol. 18, no. 5, 2018.
44. PUROHIT, Harsh *et al*. MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection. *Detection and Classification of Acoustic Scenes and Events 2019*. New York, USA, p. 209 – 213, 2019.
45. BISHOP, M. Christopher. *Pattern Recognition and Machine Learning*, Cambridge: Springer, 2006. ISBN 978-0387-31073-2.
46. TOMIOKA, Ryota. *スパース性に基づく機械学習 (Machine learning with sparsity inducing regularizations)*. Tokyo: Kodansha, 2015. ISBN 978-4-06-152910-6.
47. BALDI, Pierre and Kurt HORNIK. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Network*, vol. 2, issue 1, p. 53-58, 1989.
48. SEKIHARA, Kensuke. *統計的信号処理(Introduction to Statistical Signal Processing)*. Tokyo: Kyoritsu Publishing. 2011. ISBN 978-4-320-08567-1.
49. KATAYAMA, Tohru. *非線形カルマンフィルタ(non-linear Kalman filter)*. Tokyo: Asakura Co. Ltd., 2011. ISBN 978-4-254-20148-2.
50. WAN, A. Eric. The unscented Kalman filter for nonlinear estimation. *Adaptive System for Signal Processing, Communications, and Control*. IEEE, 2000.
51. Scikit-learn website [online, viewed 4 January 2021]. Available from: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html.
52. FAWCETT, Tom. An introduction to ROC analysis, *Pattern Recognition Letters*. vol.27, p. 861-874, 2006.
53. MathWorks, Inc. *MATLAB website: State Estimation* [online, viewed 4 January 2021] Available from: https://jp.mathworks.com/help/control/state-estimation.html?lang=en.
54. MAATEN, Laurens van der and Geoffrey HINTON, Visualizing Data using t-SNE, *Journal of Machine Learning Research*, Vol. 9, pp. 2579 – 2605, 2008.