Development of deep learning-based methods for rotating machinery fault diagnosis under varying speed conditions

by

Meng Rao

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mechanical Engineering
University of Alberta

Abstract

Rotating machines are widely used in industrial applications, such as driving motors in elevators and gearboxes in wind turbines. Machines in these applications often operate under varying speed conditions due to variable operation demand, ever-changing environment conditions and so on. As time goes on, machines in service would be inevitably deteriorated. When the deterioration is accumulated to a certain level, faults may occur. Faults if not detected timely and maintained properly would result in the shutdown of a machine, then economic loss and even catastrophic disasters. To avoid these unexpected consequences, fault diagnosis, whose goal is to detect the occurrence and then classify the type and the severity of a fault, is of vital importance.

Deep learning is widely used for fault diagnosis in the current big data era thanks to its automation nature and capability of processing massive data. However, performances of deep learning-based fault diagnosis are highly influenced by speed variation. Models perform well under constant speed conditions may fail under varying speed conditions. Specifically, when deep learning is adopted for fault diagnosis, we input condition monitoring data which are usually vibration signals to a deep learning model. The model performs fault feature learning and pattern recognition, and ultimately outputs diagnosis results. The problem is that speed variation induces additional features to vibration signals, and unfortunately, features induced by speed variation are often overlapped with features of faults. This increases the difficulty of learning sensitive fault features and thus impedes the fault diagnosis performances. Therefore, how to address the effects of speed variation is a key concern to facilitate deep learning-based fault diagnosis under varying speed conditions.

The objective of this research is to develop deep learning models that can address the effects of speed variation, and ultimately achieve effective fault diagnosis for rotating machinery that operated under varying speed conditions. This research includes three topics. First, considering that rotating speed is frequently required for effective fault diagnosis but sometimes is not feasible to be measured, a new deep learning model named many-to-many-to-one bi-directional long short-term memory (MMO-BiLSTM) is proposed to extract rotating speed from vibration signals. The proposed model can work like a virtual speed meter, that is, automatically output synchronized rotating speed corresponding to given vibration signals. Second, a new deep learning model named speed normalized autoencoder (SN-AE) is proposed for fault detection under varying speed conditions. The proposed model automatically removes speed variation induced amplitude modulation in vibration signals and thus achieves better fault detection performances. Given that a fault being detected, the last topic aims to classify the type and severity of this fault. An auxiliary branch named speed adaptive gate (SAG) is proposed for existing deep learning models to improve their fault classification accuracy under varying speed conditions. The proposed SAG addresses speed induced fault information imbalance and therefore yields higher fault classification accuracies. Both the second topic and the third topic require the rotating speed as an auxiliary input. The rotating speed can be measured or extracted from the first topic.

This research would promote the frontier of deep learning-based fault diagnosis, especially for varying speed conditions. The outcome of this research could serve as a good reference for engineering practitioners in industrial applications for a better maintenance. This research only considers the varying speed condition. The load is assumed constant. In the future, we will investigate the fault diagnosis of rotating machinery under varying load conditions.

Preface

This thesis is an original work by Meng Rao. Materials of this thesis have been published or submitted in journals or conference proceedings under the supervision of Dr. Ming J. Zuo and Dr. Zhigang Tian in conceptualization, reviewing, editing, project administration, funding acquisition and providing resources. Dr. Dalin Qin, Dr. Qing Li and Mr. Dongdong Wei are co-authors for some publications. Details are given below.

Materials presented in Chapter 3 have been published in a conference paper and a journal paper as follows. The co-authors Dr. Qing Li and Mr. Dongdong Wei assisted reviewing and editing the manuscripts.

- M. Rao, Q. Li, D. Wei and M. J. Zuo, "Virtual rotating speed meter: extracting machinery rotating speed from vibration signals based on deep learning and transfer learning," Proceedings of the 9th Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling, Vancouver, British Columbia, Canada, August 20-23, 2020, pp. 1-6.
- M. Rao, Q. Li, D. Wei and M. J. Zuo, "A deep bi-directional long short-term memory model for automatic rotating speed extraction from raw vibration signals," Measurement, vol. 158, pp. 107719, 2020.

Materials presented in Chapter 4 have been published in a journal paper as follows.

- M. Rao, M. J. Zuo and Z. Tian, "A speed normalized autoencoder for rotating machinery fault detection under varying speed conditions," Mechanical Systems and Signal Processing, vol. 189, pp. 110109, 2023.

Materials presented in Chapter 5 have been submitted as a journal paper for possible publication and partially published in a conference paper. In the conference paper, the co-author Dr. Dali Qin assisted reviewing and editing the manuscript.

- M. Rao, M. J. Zuo and Z. Tian, "Speed adaptive gate: Improving fault classification accuracy of deep learning models for rotating machinery under varying speed conditions," Measurement, Submitted, 2023.
- M. Rao, M. J. Zuo, Z. Tian and D. Qin, "Speed adaptive gates for deep learning-based fault classification of rotating machinery under varying speed conditions: Comparison of implementations," IEEE Global Reliability & Prognostics and System Health Management Conference, Yantai, China, October 13-16, 2022, pp. 1-7.

Acknowledgements

I would like to express my deepest gratitude to my supervisors Dr. Ming J. Zuo and Dr. Zhigang Tian, for their guidance, support, help, and encouragement during my PhD study. Under their guidance, I have gained not only academic knowledge but also critical thoughts. It is my honor to conduct research under their guidance. It would not be possible to have this thesis ready without their supervision.

I sincerely thank my supervisory committee members, Dr. Hossein Rouhani and Dr. Di Niu, for their valuable assistant during my study. Thanks Dr. Basel Alsayyed Ahmad and Dr. Mike Lipsett for being my candidacy examining committee members. Thanks Dr. Xiaodong Wang for being my final exam chair. Thanks Dr. Sharareh Taghipour from Toronto Metro University for being my external examiner. Thanks Dr. Tetsu Nakashima and Dr. Zengtao Chen for being my final examiners.

I am grateful to members in the Reliability Research Lab. Thanks for their help and support during my study. Thanks for the funding support of the China Scholarship Council and the Future Energy System under the Canada First Research Excellence Fund.

This thesis is dedicated to my grandparents and my parents. Thanks for their selfless support and care. This thesis is also dedicated to my wife and my son. They open a new journey of my life. This thesis is further dedicated to my sister. Many thanks go to her for covering my absence at home.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AE	Autoencoder
AI	Artificial intelligence
AM	Amplitude modulation
ANN	Artificial neural network
BiLSTM	Bi-directional long short-term memory
CBM	Condition based maintenance
CNN	Convolutional neural network
ELM	Extreme learning machine
FFT	Fast Fourier transform
FM	Frequency modulation
FNN	Feedforward neural network
GAN	Generative adversarial network
HI	Health indicator
IF	Instantaneous frequency
LSTM	Long short-term memory
ML	Machine learning
MMO-BiLSTM	Many-to-many-to-one bi-directional long short-term memory
MO-LSTM	Many-to-one long short-term memory
MLP	Multiple layer perceptron
ResNet	Residual network

RL	Reinforcement learning
RMS	Root mean square value
RNN	Recurrent neural network
RUL	Remaining useful life
SCADA	Supervisory control and data acquisition
SAG	Speed adaptive gate
SAG-CNN	Speed adaptively gated convolutional neural network
SAG-ResNet	Speed adaptively gated residual network
SN	Speed normalization or normalized
SN-AE	Speed normalized autoencoder
STFT	Short-time Fourier transform
SVM	Support vector machine
WT	Wavelet transform

1. Introduction

This chapter consists of three sections. Section 1.1 introduces the background, including prognostics and health management (PHM), fault diagnosis, deep learning, rotating machinery, and operation conditions of rotating machinery. Section 1.2 provides a detailed literature review of existing speed extraction methods, deep learning-based fault detection methods, and deep learning-based fault classification methods. Sections 1.3 and 1.4 provide the objectives and organization of this thesis, respectively.

1.1 Background

1.1.1 Prognostics and health management

We often have the following experience. A newly bought device, for example, a car, works well in the first few years. It runs smoothly, the noise and vibration levels are low, and the gas consumption is fair. However, as time goes on, for example, after 10 years, the car does not work as expected any more. It consumes more gas per mile, the car cabinet becomes noisier, and it even sometimes breaks down on the road. This worsening of service performance is known as performance degradation [1], [2]. Performance degradation is often experienced by rotating machines, usually due to fatigue and/or wear [3]. Fatigue occurs because rotating machines are often subjected to variable loads. Wear exists in between contacting surfaces if they have relatively micro and/or macro motions.

The performance of a machine often degrades gradually, as shown in Fig. 1.1. We acknowledge that machines might break down suddenly in some cases [4]. Sudden failures are usually induced by certain reasons such as excessive load, and often studied using statistical methods. This thesis focuses on only gradual degradation. A machine in its early commissioning stage is often in a healthy state, but over time its degradation begins and accumulates. In the healthy state, a machine performs its intended function adequately. When the degradation is accumulated to a certain level, a fault is initiated. A fault refers to an abnormal state or a defect at the component, equipment, or subsystem level (ISO 103003-226/ASTM standards). A machine with a fault is said to be in the faulty state. A machine in the faulty state can still perform its intended function, but not adequately or with reduced performance. If the fault is not dealt properly, it will continue to grow. Growing faults weaken the functioning of a machine and finally lead to the failure of the machine. Failure means that the machine is broken down and cannot perform its intended function at all.

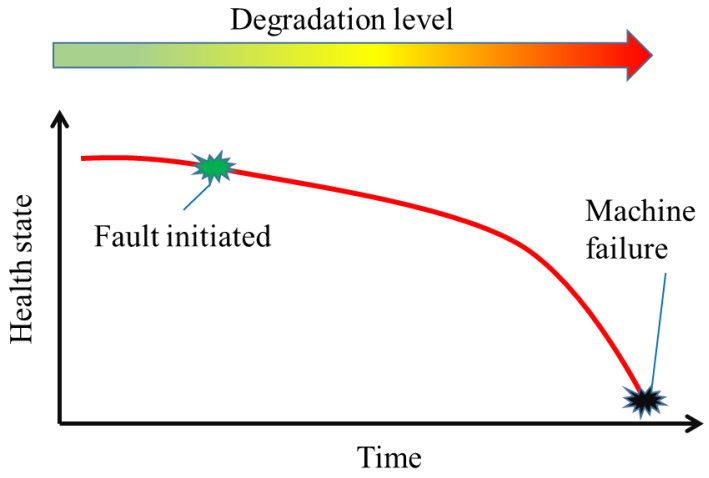


Fig. 1.1: Typical degradation process of machines [1], [2].

Failure of a machine results in unexpected shutdown of the entire machine, followed by economic loss and even catastrophic disasters [5]. For example, on April 29, 2016, a European EC225 Super Puma Helicopter crashed due to the fatigue fracture of a planet gear located in the transmission gearbox of the main rotor. All 13 people on board were killed in this disaster [6]. A very recent example is the Ohio train derailment. On February 3, 2023, a Norfolk Southern freight train carrying hazardous materials derailed in East Palestine, Ohio, United States. Tons of toxic chemicals were spilled out of the train, and polluted nearby lands, waterbodies, and even the atmosphere. More than 5,000 people had no choice but to get evacuated from their homes. Numerous fishes were killed. One reason to this disaster is the overheat failure of the wheel bearing [7]. Definitely we do not want either the economic loss or any fatality to happen. Timely and proper maintenance actions should be taken to avoid the occurrence of failures.

Existing maintenance strategies are broadly three types: run-to-break, preventive maintenance, and predictive maintenance [8]. Run-to-break is a traditional method. Machines simply run until they break down. This gives the longest operating time between shutdowns, but breakdowns are occasionally catastrophic, with severe consequences for safety, production loss, and repair loss. The loss is not only induced by the failure of the machine, but also machines connected to it. This can substantially exaggerate the loss. Preventive maintenance is carried out at regular intervals to assure a very small likelihood of failure between repairs. The advantage of this approach is that most maintenance can be planned in advance, and catastrophic failure is greatly reduced. The disadvantages are that intensive labor is required to perform regular checks and maintenance, and thus can be costly. Predictive maintenance is also referred to as condition-based maintenance (CBM), or more broadly, PHM. We predict potential failures through regular or case-sensitive condition monitoring and conduct maintenance at an optimum time. PHM is free from the

disadvantages of either run-to-break or preventive maintenance. It is being recognized as the most efficient strategy for implementing maintenance in many industries [8], and more accepted in engineering practices. A recent report shows that the PHM market will rise from USD 2.6 billion to 3.9 billion from 2019 to 2025 [9]. In this thesis, we focus on PHM.

The goal of PHM is to provide preventive solutions to improve the reliability, maintainability, safety, and affordability of machines [10]. PHM uses information extracted from condition monitoring data to assess the health state of a machine and drives maintenance operations accordingly [11]. Fig. 1.2 shows the main components constituting a typical PHM structure, from data acquisition to decision making [11], [12], [13].

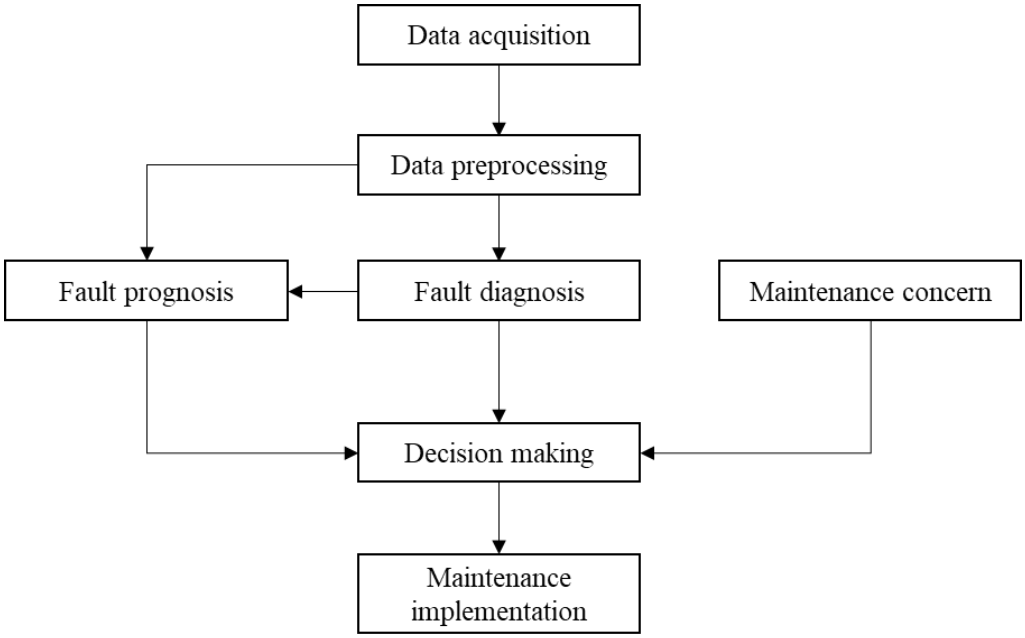


Fig. 1.2: Typical structure of PHM [11], [12], [13].

Data acquisition involves collecting and storing condition monitoring data for a system of interest. The condition monitoring data refers to the data collected with sensors installed in and/or outside

a machine, such as the vibration, temperature, acoustic emission, acoustic pressure, and oil debris. PHM requires the condition monitoring data relevant to machine health state, so that they can indicate the machine health state through successive fault diagnosis and prognosis [12]. Data preprocessing involves converting collected raw data into clean data [14]. It ensures, or at least increases the chance, that reliable and useful data is used for future analysis. Data preprocessing includes but not limited to outlier removal, denoising, segmenting, concatenating and normalization [14].

Fault diagnosis and fault prognosis are the two major phases of PHM [8]. Fault diagnosis aims to identify the occurrence of faults and pinpoints the causes of faults. Fault prognosis aims to predict the remaining useful life (RUL) of a machine based on the current health state and historical condition monitoring data. Fault diagnosis is a prior step for prognostics. Maintenance concerns refer to factors that should be considered when making maintenance decisions, such as the degree of maintenance degree, maintenance cost, availability of maintenance personnel and maintenance facility, and so on [12]. Decision making involves selecting the optimum maintenance option for the machine of interest. This is an integrated process that should comprehensively consider the results obtained from diagnosis, prognosis, and maintenance concerns. The selected maintenance option should provide a detailed guide for implementing the maintenance, such as the degree of maintenance, personnel, and logistic support. The maintenance action is then implemented accordingly [12].

In this thesis, we focus on fault diagnosis only. Other components of PHM will be studied in the future work. A further discussion of fault diagnosis is given below.

1.1.2 Fault diagnosis

As described in Section 1.1.1, the goals of fault diagnosis are to identify the occurrence of a fault, and then identify the root cause of the fault. Thus, fault diagnosis includes two specific tasks: fault detection and fault classification.

Fault detection aims at achieving the first goal, i.e., identifying whether a fault has occurred in a machine or not. If a fault could be detected before it propagates to a failure, timely maintenance strategies could be implemented so that the failure would be prevented.

Fault classification aims at achieving the second goal. Fault classification has two subtasks: identify fault types and identify fault severities. The fault type refers to what fault has occurred at what component of a machine, e.g., gear tooth pitting and bearing inner race crack. The fault severity refers to the degradation level of a fault. Determination of fault types and severity levels helps to better understand the root cause and the threat of the fault. This could be useful for making maintenance decisions. The severity level of a fault is often quantitatively measured as incipient (or minor), medium, or major [12]. The severity level of a just initiated fault is incipient or known as minor. A fault that approaches failure is regarded as a major fault. The severity level between the incipient and major is medium. The fault severity level is positively correlated with the machine performance degradation level. The more severe the fault is, the more the machine performance is degraded.

Fault diagnosis is important because it is a preliminary step for predictive maintenance. Useful diagnosis together with successive prognosis must enable proper maintenance, as shown in Fig. 1.3. An underlying request here is that faults must be successfully diagnosed before propagating to failures. Moreover, adequate time should be available for successive actions including prognosis

and maintenance to prevent the failures. Therefore, fault diagnosis should be conducted as early as possible.

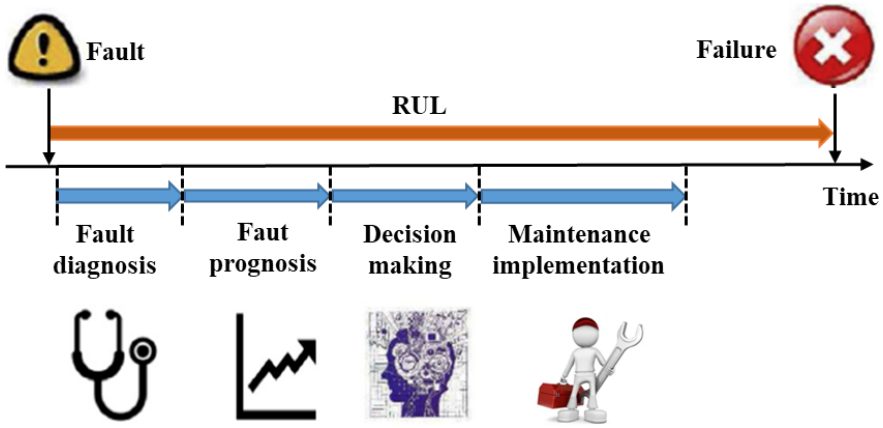


Fig. 1.3: Primary request for fault diagnosis: Enabling proper maintenance [13].

Fault diagnosis involves building maps among the information obtained in the condition monitoring data space (e.g., vibration signals) or their features and machine faults in the fault space [3]. An underlying mechanism is that the data reflects the health state of machines. Theoretically, faults can be detected because they introduce unique signatures to condition monitoring data, e.g., impulses, higher energy, higher noise, and/or higher temperature. Faults can be classified because signatures of different fault types and different fault severities are different. As an example, in gear faults, tooth wear modulates the amplitude of vibration signals, while tooth root crack induces impulses to vibration signals. Furthermore, a machine in different degradation levels excites different fault signatures. These are further reflected in different types of condition monitoring data, as shown in Fig. 1.4. As time passes, an initiated fault can introduce changes in acoustic emission, followed by vibrations, particles in the lubrication oil, noise, and heat. Because of strong sensitivity to faults and convenience of collection, the vibration signals are the most frequently

used condition monitoring data for fault diagnosis [8], [15]. In this thesis, the vibration signals are used as the primary condition monitoring data.

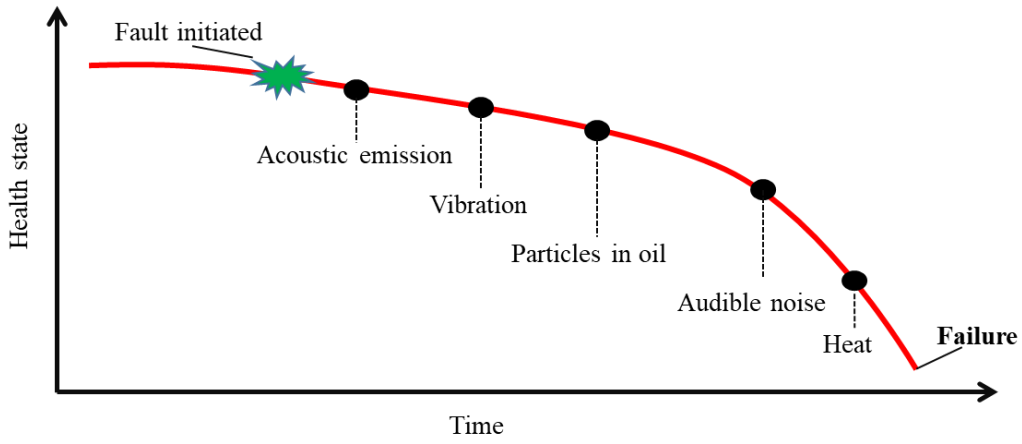


Fig. 1.4: Examples of condition monitoring data for fault diagnosis [2].

Mapping data and faults is a challenging task as the condition monitoring data including vibration signals contain not only fault information, but also other information. Essentially, vibration signals of a machine are comprehensive responses of multiple excitation sources including the information about the health state, machine operation condition, environment noise, resonance properties of machine systems, and the transmission properties between the fault location and the sensing location and so on [8], as shown in Fig. 1.5. Changes in vibrations induced by these sources are often overlapped. Sometimes the fault information is quite weak compared to other information such as environment noise. The secret of a successful fault diagnosis is then to preserve fault related information from condition monitoring data while mitigating the effects of other sources as much as possible. Among these sources, the operation conditions, especially the varying speed condition, play a crucial rule [8], and thus they are the focal point in this thesis. More information about the varying speed conditions is given in Section 1.1.5.

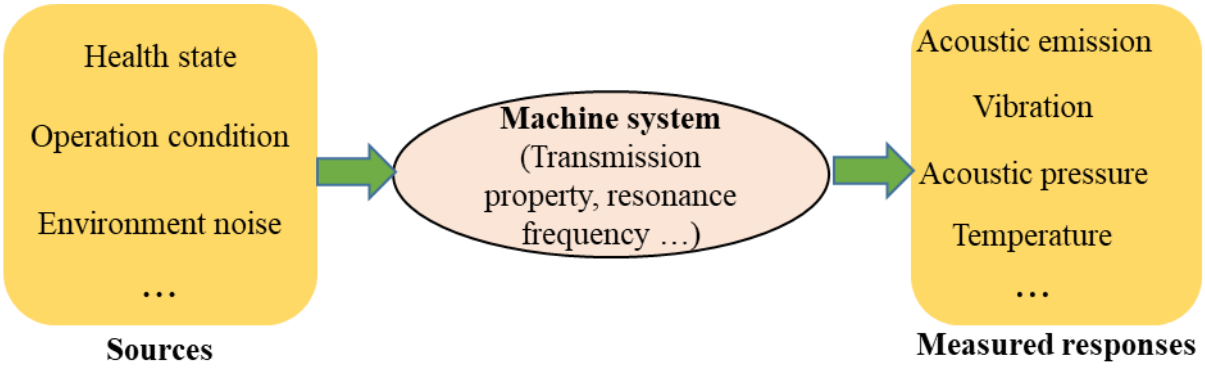


Fig. 1.5: Illustration of excitation sources of measured signals [8].

The existing fault diagnosis methods broadly have three categories: physics based, data-driven, and hybrid methods [3], [13], [16], as shown in Fig. 1.6. Descriptions of these methods are given separately in the following.

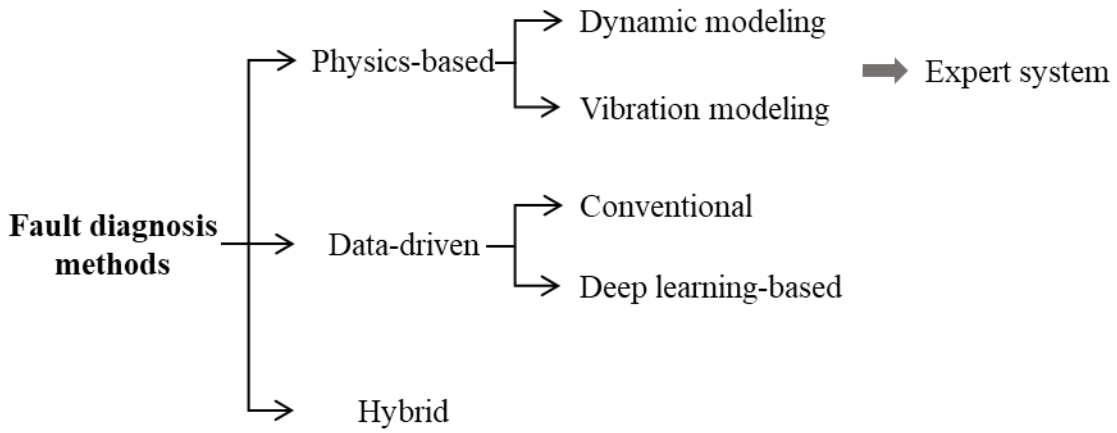


Fig. 1.6: Classification of fault diagnosis methods [16].

Physics based, also known as model based, methods follow a top-down manner [16], i.e., the map between the condition monitoring data is built from machine faults to data. The maps are built using explicit mathematical models that are derived manually. These models are built according

to physical mechanisms of faults with sufficient assumptions and simplifications. A well-built mathematical model provides theoretically expected signatures of certain faults. If we could find expected fault signatures in the collected condition monitoring data, we can conclude that the fault is diagnosed. For example, rolling element bearings generally have four fault modes, i.e., inner race fault, outer race fault, roller fault and cage fault. Each fault excites a specialized characteristic frequency in the spectrum of vibration signals, i.e., ball pass frequency inner (BPFI), ball pass frequency outer (BPFO), ball spin frequency (BSF) and fundamental train frequency (FTF) [8]. If we can find a characteristic frequency, e.g., BPFI, in the spectrum of a collected vibration, a corresponding fault, inner race fault, is diagnosed.

Physics-based methods further include dynamic modeling, vibration modeling and other physical models from the perspective of how the mathematical model is built. Dynamic modeling simplifies machines into rigid masses that are connected through massless springs as well as damping. Faults can be modeled as the changes in one or more of these three components, and/or excitation forces. Equations of motion are then derived according to the Newton's Law. The equations are often solved using numerical methods. The results are vibration signals, such as the displacement, velocity, acceleration, and their angular counterparts. The expected fault signatures are then revealed in results like accelerations. More on dynamic modeling can be found in [17], [18] and Chapter 8 of [8]. Vibration modeling builds analytical equations for vibration directly without deriving dynamic equations. The primary assumption is that the vibration of machines is the summation of infinite terms of sinusoidal signals. Expected faults can be modeled as certain changes in the amplitude and/or frequency of certain terms. More on vibration modeling can be found in [19] and Chapter 2 of [8]. The abovementioned two modeling methods are directly or indirectly based on the Newton's Law. Besides them, physical methods relying on other theorems

such as the fatigue analysis can also be used for machinery fault diagnosis. Two commonly used fatigue theories are the S-N curve and the Paris' Law. They are frequently utilized for structure health condition monitoring and fatigue life prediction [20], [21].

Physics-based methods can be developed into the "expert system". If the knowledge gained by the abovementioned mathematical models and the successive reasoning process are coded in a computer, and the computer conducts fault diagnosis automatically, it can be said that we have built an expert system [5].

Physics-based methods are easy-to-interpret and are effective if a correct and accurate model is built. However, explicit mathematical modeling might be infeasible for complex systems because it would be very difficult or even impossible to build mathematical models for such systems [3]. Advantages and disadvantages of physics-based methods are summarized in Table 1.1.

Data-driven methods follow a bottom-up manner [16], i.e., the map between the condition monitoring data and faults is built from data to machine faults. The map is automatically learned from the condition monitoring data, without explicit mathematical modeling. Data-driven based fault diagnosis often consists of five steps, i.e., data acquisition, data preprocessing, feature extraction, feature selection, and fault detection or fault classification, as shown in the left panel of Fig. 1.7. Data acquisition and data preprocessing are introduced in Section 1.1.1. They are identical to those for PHM. Here, signal processing techniques such as filtering, Fast Fourier Transform (FFT), Short-Time Fourier Transform (STFT) and Wavelet Transform (WT) are usually used to preprocess data. For the convenience of narrative, data collected from a healthy machine is known as healthy data, otherwise, it is known as faulty data.

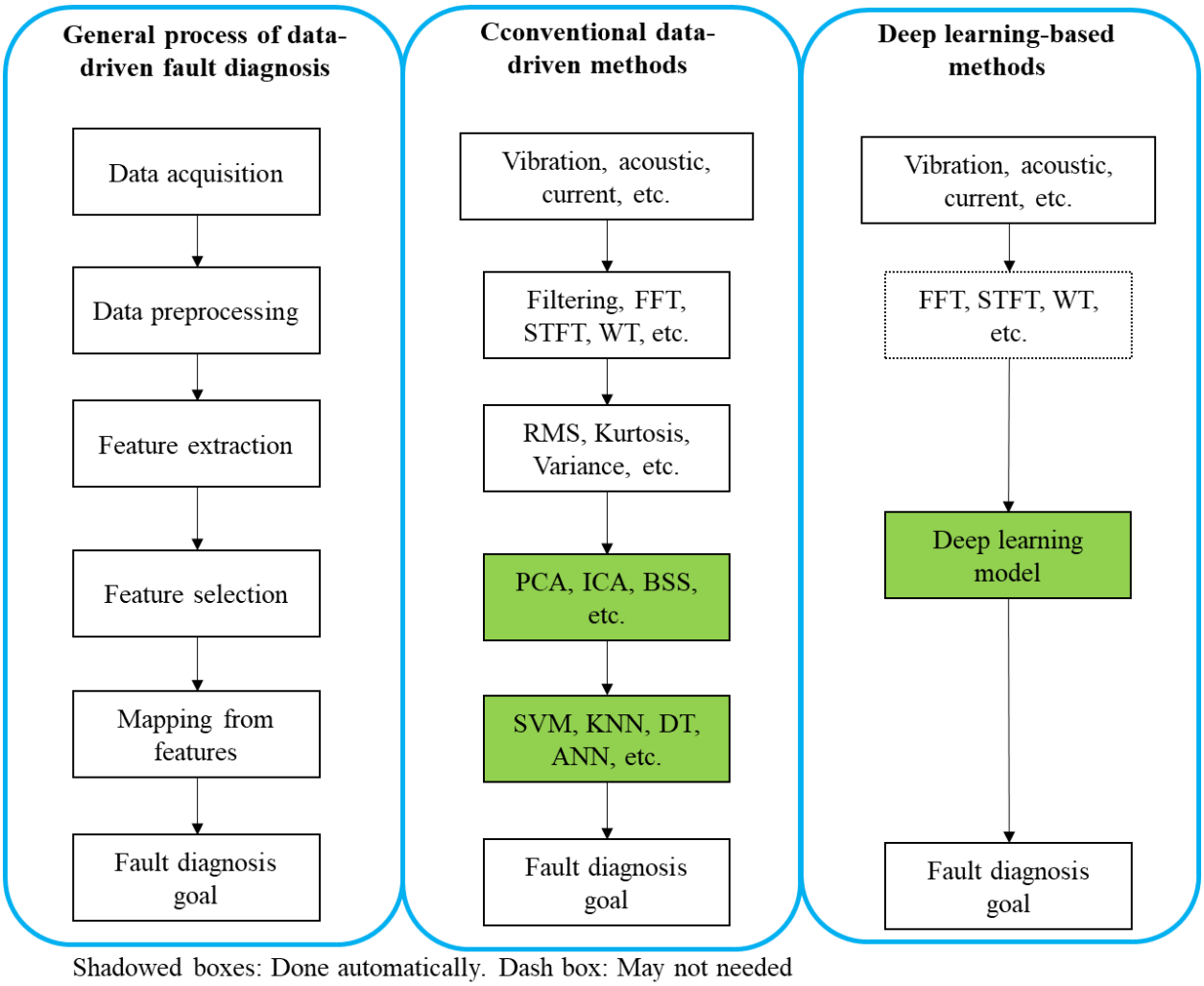


Fig. 1.7: Frameworks of data-driven fault diagnosis methods [22].

Feature extraction refers to extracting the fault-sensitive indicators from condition monitoring data. The extracted indicators are also known as features. Commonly used features include root mean square (RMS), kurtosis, and variance. Signal processing methods such as time-domain analysis, frequency-domain analysis and time-frequency domain analysis are frequently used to extract features [5]. The feature extraction process is often time consuming as case-dependent knowledge and studies are required to select appropriate signal processing tools among several possibilities

[3]. Feature selection is to select independent and better ones from all the candidate features. Feature selection is necessary as some extracted features are bad features which would deteriorate fault diagnosis performances, and more importantly, features may exist that are correlated with each other. For correlated features, only one feature is needed, otherwise, fault diagnosis would be biased to the trend indicated by these features, not mention that more computation time is required. Techniques such as the Principal Component Analysis (PCA), Independent Component Analysis (ICA); and Best Subset Selection (BSS) are often used for feature selection. The well selected features are then used for fault diagnosis algorithms to conduct fault detection or fault classification. Note that even the process for fault detection and fault classification is the same, the their exact features are often different.

Data-driven fault diagnosis methods can be further categorized as conventional data-driven methods and deep learning-based methods in terms of how the abovementioned five steps are implemented, as shown in Fig. 1.7. Conventional data-driven methods conduct feature extraction, feature selection and fault diagnosis separately. The features are handcrafted and required to be carefully selected for every single task. The fault diagnosis algorithms are often machine learning algorithms but exclude deep learning. More information about the machine learning and deep learning is given in Section 1.1.3. Machine learning algorithms like Support Vector Machine (SVM); K-Nearest Neighbors (KNN), Decision Tree (DT) and Artificial Neural Network (ANN) are often used here. Deep learning-based methods conduct feature extraction, feature selection and fault diagnosis simultaneously in a single step using deep learning algorithms. Deep learning-based methods are end-to-end methods. We simply input condition monitoring data to a deep learning model, which automatically returns fault diagnosis results.

In comparison, conventional data-driven methods need intensive expert knowledge and labor for feature extraction and feature selection, making them less favorable for massive data. Deep learning-based methods automatically fulfil these steps in a single process with limited or even without manual interference or expert knowledge. Thus, they are good for massive data. Moreover, deep learning-based methods conduct feature extraction, feature selection and fault diagnosis simultaneously. An overall optimum of all steps can be expected. Conventional data-driven methods can only obtain a local optimum for each single step, but the overall optimum is not guaranteed [16], [23]. Detailed comparisons of conventional data-driven methods and deep learning-based methods are shown in Table 1.1.

Meanwhile, with exponentially more condition monitoring data collected from increasingly complex rotating machinery, fault diagnosis has already entered the era of big data [5], [24]. In such an era, we have strong demands for intelligent methods to deal with massive data to automatically mine complex nonlinear relationships between the faults and condition monitoring data [5], [25]. Deep learning, an emerging technique that can learn hierarchical representations of raw data and any complex relationships between faults and raw data [26], [27], is ready to solve such problems. Indeed, deep learning has been successfully used in computer vision [28], natural language processing [29], autonomous driving [30], and human health diagnosis [31]. Therefore, deep learning has been attracting more and more attention in the PHM community. In this thesis, we focus on deep learning-based fault diagnosis. More information about deep learning is given in Section 2.

Hybrid methods use physics-based and data-driven methods simultaneously. Hybrid methods are welcomed because the physics-based and data-driven methods often mutually assist each other.

Firstly, physics-based methods sometimes require data-driven methods to determine their parameters. For example, Paris's Law often uses linear fitting to determine the stress intensity factor range and material parameters [21]. Secondly, data-driven methods can achieve better performances if proper physical knowledge is informed. For instance, in unsupervised learning-based fault detection, we do not have faulty data in the model development stage. The developed model knows nothing about the faulty state. However, by hand, we already know how fault signatures look like theoretically through mathematical modeling. If such knowledge is well informed to a fault detection model, its detection performance can be improved [32]. Indeed, the physics informed data-driven method is a promising research topic in the context of PHM [33] and other domains such as geophysics and molecular simulations [34]. This thesis does not cover hybrid methods.

Table 1.1: Summary of rotating machinery fault diagnosis methods [14], [16].

Methods		Advantages	Disadvantages
Physics based		1) Deterministic and precise 2) Requires little data 3) Good interpretability	1) Difficult to be implemented 2) Requires complete knowledge of system behaviors 3) Less feasible for complex system
Data-driven	Conventional	1) Simple and ease to be implemented 2) Fair interpretability 3) Feasible for complex systems	1) Requires fair amount of data 2) Requires intensive labor work and expert knowledge
	Deep learning-based	1) Simple and ease to be implemented 2) Feasible for complex systems	1) Requires massive data 2) Poor interpretability
Hybrid		1) Can be used with a lack of certain data 2) Accurate performances 3) Fair interpretability	1) High complexity of implementations 2) Requires both expert knowledge and data

1.1.3 Deep learning

Deep learning is a subsect of machine learning (ML), which is used for many but not all approaches to artificial intelligence (AI) [26], as shown in Fig. 1.8. AI refers to the simulation of human intelligence in machines. It is programmed to think like humans and mimic their actions. In the early days of AI, people coded the rules, and the computers automatically run the rules. An example is the knowledge base. People hard-coded knowledge or rules about the world in formal languages. A computer automatically makes reasoning in these formal languages using logical inference rules. The application of knowledge bases in the context of fault diagnosis is an expert system as indicated in Section 1.1.2.

Machine learning is a subset of AI that gives computer the ability to learn without explicit programming. It often refers to the automation of learning process from features. Fig. 1.9 shows that how machine learning works compared to traditional approaches, i.e., non-machine learning approaches. The machine learning "learns" rules from data and answers (also known as labels), while conventional approaches program given rules to obtain answers. Note that even we do not need to hard-code knowledge in machine learning, the features are often handcrafted. A machine learning algorithm often has a shallow structure, such as the two-layer ANN, logistic regression (LR), support vector machine (SVM), decision tree (DT), and principal component analysis (PCA). The application of such machine learning algorithms in the context of fault diagnosis is the conventional data-driven approaches as indicated in Section 1.1.2.

Deep learning is a subset of machine learning that uses a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as the input. The "deep" here simply means that the number of layers in

the model is deep. When deep learning is used for fault diagnosis, it is known as deep learning-based approaches as described in Section 1.1.2. The deep learning-based fault diagnosis is the focus of this thesis.

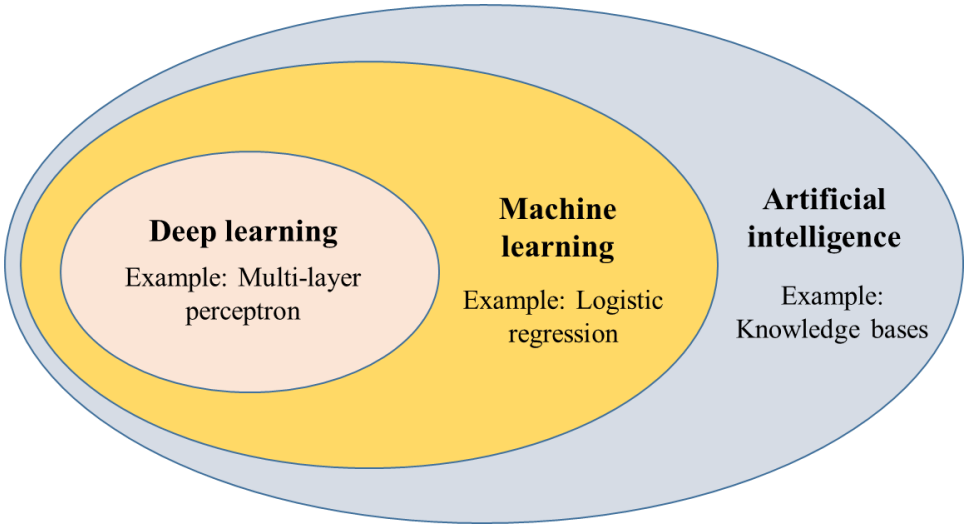


Fig. 1.8: Venn diagram showing the position of deep learning in machine learning and artificial intelligence [26].

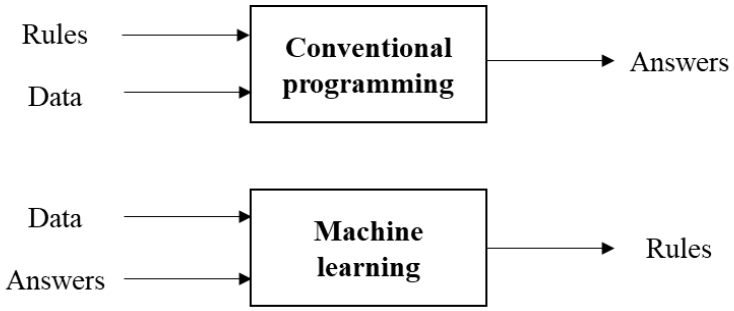


Fig. 1.9: How machine learning is different from conventional approaches [35].

Deep learning models are even not all, but pretty much artificial neural networks (ANNs) with deep layers. An exception is the graphical model, such as the deep belief network [26]. In this thesis, we utilize the ANN based deep learning models only. The ANN is briefly introduced below.

Fundamentals of deep learning including ANNs are given in Chapter 2 and are briefly introduced below.

An ANN is a network-like machine learning model that consists of inter-connected artificial neurons, or units. The neurons are often organized in layers. ANNs might have different structures. Three typical ANN structures are feedforward neural network (FNN), convolutional neural network (CNN) and recurrent neural network (RNN) [26]. In the FNN, the neurons between two successive layers are fully connected. The information flows in the forward direction and flows among the layers only. Two typical FNNs are multilayer perceptron (MLP) and autoencoder (AE). MLP is the most fundamental neural network and is often used for simple regression or classification tasks. AE is a network that tries to reconstruct its input and is often used to learn dimension-reduced representations of the input. CNN is more compact than FNN. The layers of a CNN are not fully connected. The neurons of a preceding layer are only mapped to neurons at certain locations of a successive layer. A widely used variant of CNN is residual network (ResNet). CNN is often used to process structured data such as images and sequential data. RNN allows information flowing not only among layers but also within layers. A widely used variant of RNN is the long short-term memory (LSTM) network. RNN is naturally specialized for time series learning.

Deep learning models, put simply, are neural networks with deep structures or more layers. For FNNs, a deep learning model must have at least two hidden layers. The reason for going deep is that deep models can achieve better performance in the current big data era compared to conventional machine learning methods, as shown in Fig. 1.10. The conventional machine learning methods refer to machine learning excluding deep learning.

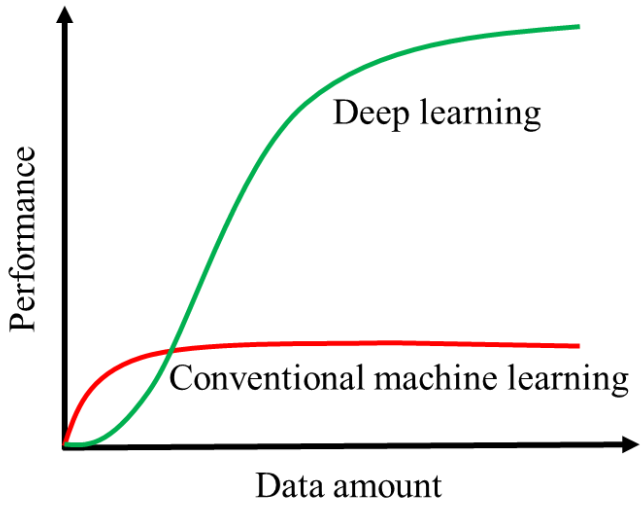


Fig. 1.10: Performance of deep learning and conventional machine learning [26].

Deep learning can be categorized into supervised learning, unsupervised learning, and reinforcement learning from the perspective of data used for training, as shown in Fig. 1.11. This categorization also works with machine learning.

Supervised learning uses labeled data to train algorithms. Labeled data contain a tag or label, referring to the exact or true value. The labeled data is in pairs (X, Y), where X is the independent variable that becomes the input of a deep learning model, and Y is the label of X that the output of the deep learning model attempts to predict. For example, the label can be whether a vibration signal indicates a fault or not, or the crack length of an aluminum plate at certain loading cycles. The label can be categorical or continuous. If the label is categorical, it can be said that the deep learning model fits a classification problem, otherwise, a regression problem. Supervised learning can achieve advanced performance, but the label might be expensive to obtain and even may not be obtained. Commonly used supervised learning models are CNN and RNN. In the context of fault diagnosis, supervised learning is often used for fault classification.

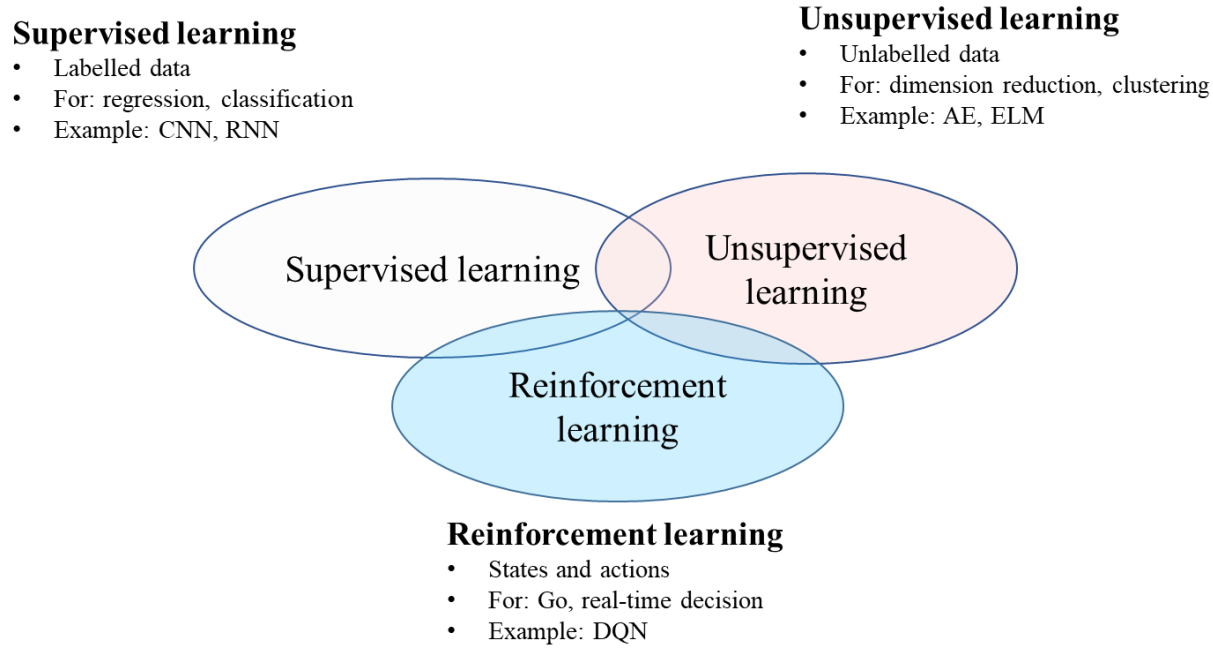


Fig. 1.11: Deep learning categories in terms of data used.

Unsupervised learning uses unlabeled data to train algorithms. Unlabeled data do not contain a tag or label. Unsupervised learning tries to make sense by extracting features or patterns on its own. It is often used to reduce the dimension of raw data, or cluster the inner patterns of raw data. Unsupervised learning does not require expensive labels, but the performance of unsupervised learning often might not satisfy the case-wise requirements in applications. Examples of unsupervised learning models include the AE and extreme learning machine (ELM). In the context of fault diagnosis, the unsupervised learning is usually used for fault detection.

Reinforcement learning (RL) trains an algorithm with a reward system. Given the environment states, an RL agent/algorithm tries to perform best actions to maximize the rewards. The mechanism of determining what actions should be taken is named as the policy, which will be learned/optimized. The reward is related to the goal of the RL task, for example, winning the go.

A widely used RL model is the deep Q-learning network (DQN) [26]. However, we found few studies on using RL for machinery fault diagnosis [36] and prognosis [37]. It may be promising for maintenance decision making but is out of the scope of this thesis, and thus will not be covered.

Deep learning approaches can also be classified into classic deep learning and transfer learning according to whether to transfer knowledge from one model to another, as shown in Fig. 1.12. In the classic deep learning, different learning systems learn separately from different domains. The domain can be simply interpreted as the task, such as gearbox fault classification under constant speed conditions. The knowledge from different domains is not shared. While transfer learning, given source domain(s) and a target domain, it aims to improve the learning in the target domain using the knowledge in both the source domain(s) and target domain. This process is said of transferring knowledge from the source to the target. In the context of fault diagnosis, transfer learning is promising for scenarios wherein the source domain has a large amount of labeled data, but the target domain has limited or even no labeled data [38]. Such scenarios include transferring knowledge across operation conditions or health states of a single machine [39], across machines in a fleet [40], and transferring knowledge from simulated data or experimental data to field data [41]. More on transfer learning for machinery fault diagnosis can be found in review papers [42], [43]. A prerequisite for a successful transfer learning is that the model must perform sufficiently well in the source domain. This means, the classic deep learning models must perform well inherently. In this thesis, we focus only on the classic deep learning. Transfer learning will be studied in the future. For simplicity, deep learning refers to classic deep learning unless otherwise stated in this thesis.

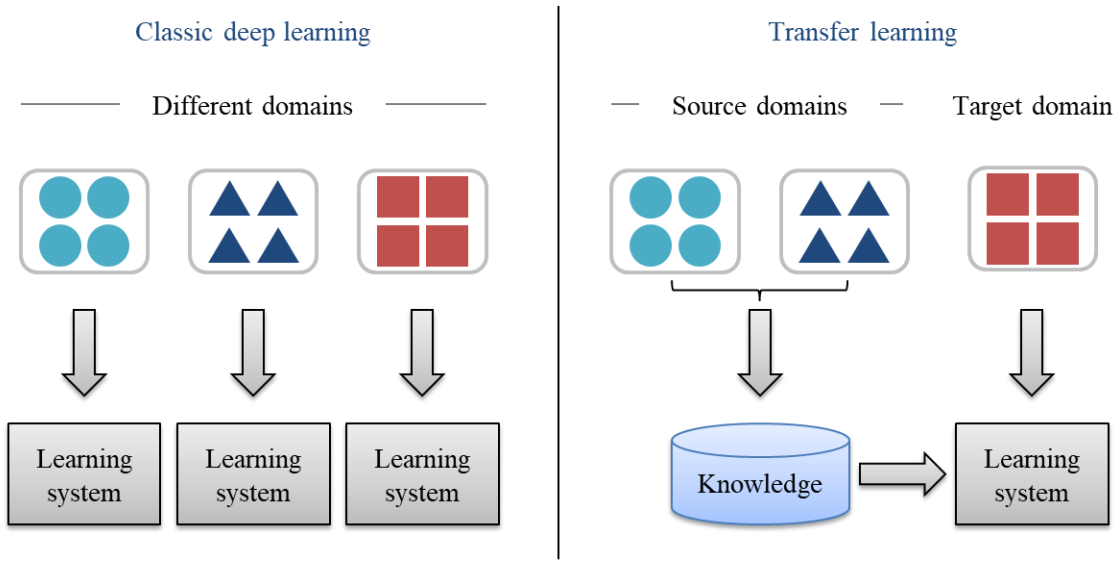


Fig. 1.12: Deep learning categories in terms of whether knowledge is transferred across domains.

As indicated in Section 1.1.2, fault diagnosis has two tasks, i.e., fault detection and fault classification. When deep learning is applied for fault detection, it is deep learning-based fault detection. Similarly, applying deep learning for fault classification is deep learning-based fault classification. Deep learning-based fault detection is usually an unsupervised learning problem. The above-mentioned AE, and the extreme learning machine (ELM) and the restricted Boltzmann machine (RBM), and their variants can be applied to fault detection. A detailed review of deep learning-based fault detection is given in Section 1.2.2. Deep learning-based fault classification is often a supervised learning problem. The abovementioned RNN and CNN, as well as FNN and ResNet, and their variants are often used for this purpose. A detailed review of deep learning-based fault classification is given in Section 1.2.3.

1.1.4 Rotating machinery

PHM has diverse application areas, such as buildings like skyscrapers and bridges [44], electronic devices such as batteries and circuits [45], and machinery. In this thesis, we focus on the machinery, more specifically, fault diagnosis of rotating machinery.

According to the type of motion, machinery can be categorized as rotating machinery and reciprocating machinery. Rotating machinery is a type of machines that are composed of at least one rotating part and certain nonrotating parts [46]. The rotating part conducts rotary operations, and the nonrotating parts are often stable. For example, a gearbox is a rotating machinery, whose rotating parts are the shafts and gears, and nonrotating parts are the housing. Rotating machinery is widely utilized in both the industry section and our daily life. To name a few, wind turbines, generators, gas turbines and pumps are used industrial applications, and gear transmission systems in watches and motors in coffee blenders belong to daily usages. Reciprocating machinery is a type of machines wherein at least a part conducts reciprocating motion such as the cam and linkage [46]. Widely used reciprocating machines include vibration shakers, reciprocating pumps, and piston engines. The reciprocating motion is often driven by a rotating machine. For instance, the cam is often driven by a motor. Fault diagnosis of reciprocating machinery is another topic of PHM [8], [47]. This is out of the scope of this thesis.

Usually, compared to other components in a machine, rotating components are more frequently subjected to faults due to their increasingly complex structures and increasingly more harsh serving conditions [5], [8]. For example, in wind turbine, the failure of the rotating parts (gearbox, generator, and yaw system) occupies 22% of wind turbine failures [48], as shown in Fig. 1.13. We acknowledge the electronic system, control system, sensors, blades, and brakes also have high

failure rates. They are out of the scope of this thesis but within the sub-field of PHM of electronic systems [45] or structure health monitoring [44]\. In this thesis, we focus on only rotating machinery.

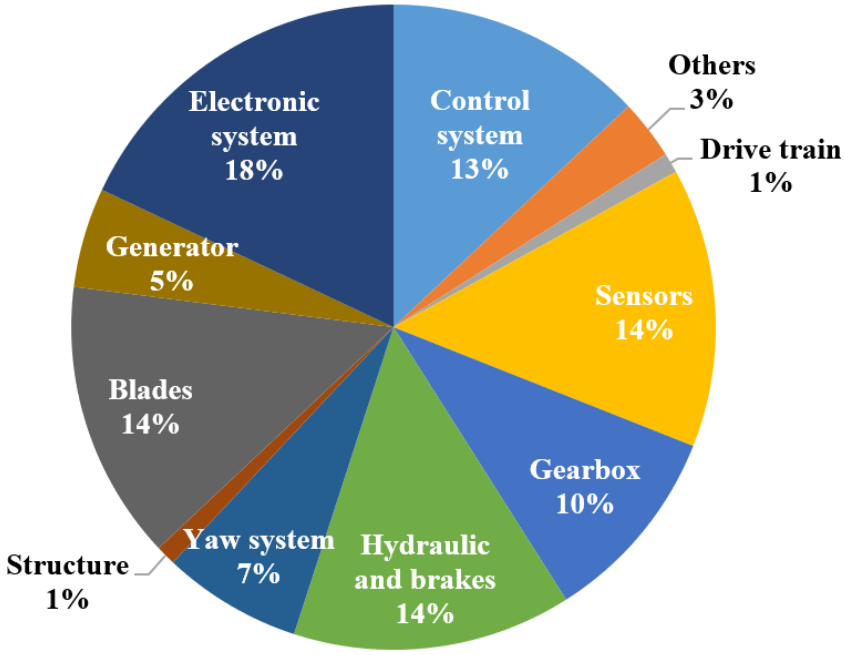


Fig. 1.13: Failure occurrence rates of main components of wind turbines [48].

1.1.5 Operation conditions of rotating machinery

1.1.5.1 Operation conditions

The operation conditions of rotating machinery refer to its rotating speed and load, or equivalently torque, subjected to it. The speed and load can be constant, or they can vary. Being constant means the speed (or load) remains unchanged over time. To vary means the speed (or load) changes over time. When the speed is constant, the load can be constant or varying, and vice versa. If both the rotating speed and load of a machine are constant, it works under stationary conditions, otherwise

it works under nonstationary conditions. In this thesis, the speed refers to the rotating speed, and may alternatively be shown as speed, rotating speed, and sometimes speed signals.

In industrial applications, there are machines working under constant speed conditions, such as asynchronous motors and generators. There are also machines that work under varying speed conditions. For example, the driving motor of an elevator needs to frequently start, run forward or reverse, and stop to allow passengers onboard, reach intended floors and allow passengers offboard. Fig. 1.14 illustrates its rotating speed profile in a whole working course. We can see that the rotating speed experiences acceleration, becomes constant and then slows down. This speed variation is repeated once a new working course is initiated.

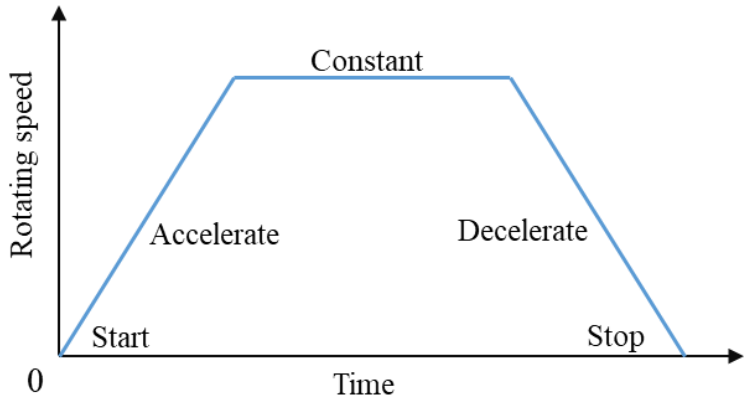


Fig. 1.14: Rotating speed profile of the driving motor of an elevator [49].

Another example of varying speed conditions is a wind turbine, powered by the wind. Because of the random nature of wind flow, the resulting rotating speed of a wind turbine is random. Fig. 1.15 shows the measured rotating speed of a wind turbine served in a Swedish wind farm for about four years. The condition monitoring data including rotating speed were collected every 12 hours. Each

measurement lasted for about 1.28 s. Each circle in the figure represents the average rotating speed of a 1.28 s long measurement. We can see that the rotating speed of the wind turbine varied randomly in between 700 rpm and 1200 rpm.

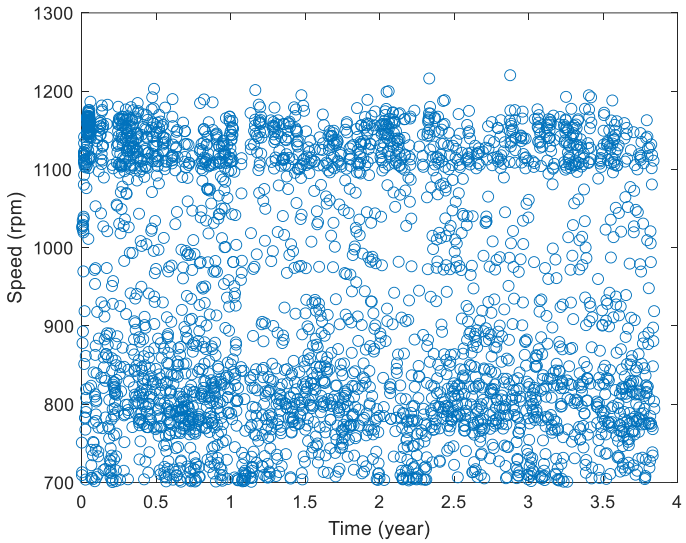


Fig. 1.15: Rotating speed of a wind turbine that served in Sweden. A circle represents the average speed of a 1.28 s-long measurement. Data credit [50].

For machines working under varying speed conditions, the load can be either constant or varying. For example, a drive conveyor system usually works under varying speed and constant load conditions. An elevator can work under varying speed and constant load conditions in a single lifting course if the passengers are unchanged. However, a wind turbine usually works under both varying speed and varying load conditions. In this thesis, we focus on the varying speed conditions. We assume that the load is constant. The varying load condition coupled with a constant or varying speed condition will be studied in the future.

1.1.5.2 Effects of rotating speed variation

Rotating speed variation introduces additional amplitude modulation (AM) and frequency modulation (FM) to vibration signals [15], [51]. The AM (FM) refers to that the instantaneous amplitude (frequency) of a signal is altered in a certain manner due to external effects. The AM and FM effects can also be introduced by faults [52]. The AM and FM effects induced by speed variation and faults are often overlapped. The overlapped effects if not distinguished, might lead to false alarms and/or missing alerts in fault diagnosis. In addition, a byproduct of FM is that speed variation leads to frequency smearing in the frequency domain of vibration signals [51]. Event frequencies no longer dominate at specific values such as the constant speed conditions but distribute in a wide range. Fault related frequencies which are critical fault signatures are mixed with fault unrelated frequencies. These exaggerate the difficulty of fault diagnosis. Detailed effects of speed variation on deep learning-based fault detection and fault classification are discussed in Chapters 4 and 5, respectively.

1.1.5.3 Methods to obtain rotating speed

For machines operated under varying speed conditions, the benefits of knowing the rotating speed are at least three-fold. First, the speed is an important condition monitoring indicator. We need to monitor the real-time rotating speed of a machine to avoid overhauling its rated speed. This is often dangerous. This is critical for wind turbines. If the rotating speed of a wind turbine reaches a certain limit, the blades should be moved away from the wind direction, or simply the blades should be held still to reduce the rotating speed of wind turbines to assure safety. Second, speed signals can facilitate fault diagnosis as they contain information related to machine health state [53]. Third, there are cases wherein the rotating speed is necessary, such as the control system of servomotors.

The instantaneous rotating speed is feedbacked to the control system to assure the moving and positioning accuracy of the servomotor.

Because the rotating speed is so important, we focus on how to obtain the speed signal in this thesis. The existing methods to obtain rotating speed are broadly categorized into two types: (1) directly installing speed sensors to measure speed [54] and (2) indirectly extracting speed from frequently used condition monitoring data such as vibration signals [55]. Direct measurement of speed is straightforward and reliable. This approach does not require much expert knowledge or complex algorithms to calculate the speed from the measured signals. Widely used sensors for speed measurement include encoders and key phasor transducers. However, sometimes it is not possible to install speed sensors due to constrains posed by the structure of target machines and/or environment space. Even when it is possible, installment of speed sensors can increase the condition monitoring cost as we need not only the speed sensors, but also data acquisition systems to read and process the sensor data [55]. Indirectly extracting speed from vibration signals is cost-effective and free-from space constraints. It is cost-effective because we do not need to buy speed sensors but instead taking advantages of existing vibration sensors. An assumption which does hold herein is that most condition monitoring systems collect vibration data. It is free-from space constrains as vibration sensors are easier to install and has limited requirements of the geometric environment. Vibration sensors are usually adhered to the static surfaces of a machine through the glue, paraffin, magnet, and sometimes screw, instead of working with rotating shafts like speed sensors. However, indirect speed extraction needs complex algorithms. The accuracy of speed extraction highly depends on the algorithm design. Literature review on rotating speed extraction is given in Section 1.2.1.

1.2 Literature review

This section provides a literature review on the methods for extracting rotating speed from vibration signals, deep learning-based fault detection and deep learning-based fault classification.

1.2.1 Extraction of rotating speed from vibration signals

Rotating speed extraction is also referred to instantaneous angular speed or frequency extraction or estimation. It refers to techniques that subtract rotating speed from condition monitoring data such as vibration, current and acoustic signals. Signals collected by speed sensors such as encoders are not included. For these signals, the speed can be easily found by counting the number of impulses or peaks in such signals. The available speed extraction methods can be broadly classified into two categories, i.e., signal processing based methods and deep learning-based methods. Detailed reviews of these methods are given below separately.

1.2.1.1 Signal processing based methods

Signal processing based speed extraction refers to using signal processing techniques to subtract speed related harmonics from vibration signals. The harmonics are either the speed or its multiples. The signal processing based speed extraction further includes phase demodulation based methods and time frequency representation based (TFR based) methods.

Phase demodulation based methods

Phase demodulation based methods determine the speed by demodulating the phase of vibration signals. The diagram of a standard phase demodulation based method is shown in Fig. 1.16 (a).

The raw vibration signal is first transformed to the frequency domain with the use of Fourier transform. The frequency spectrum is examined manually to select a proper frequency band that contains the single speed-related harmonic. A band-pass filter is then applied to filter out a signal corresponding to the selected frequency band from raw vibration data. The filtered signal must contain only a mono-component otherwise successive phases cannot be correctly calculated. Next, Hilbert transform is performed to the filtered signal to obtain its analytical form, and then the instantaneous phase and the rotating speed could be obtained [56]. Bonnardot et al. [57] used this method to extract the rotating speed of a four-stage fixed-shaft gearbox from its acceleration signals. The obtained speed was then used to resample the acceleration to the angular domain. Combet et al. [58] applied the standard phase demodulation method to extract the rotating speed of a two-stage helical reduction gearbox used in a wastewater treatment site from its acceleration signals. The extracted speed helped in the time synchronous averaging (TSA) of acceleration signals and ultimately facilitated the diagnosis of pitting faults of the gearbox.

The standard phase demodulation based method can fail when the speed fluctuates largely such that the spectra of vibration signals are smeared [55], [56]. The bandpass filter cannot return a mono-component signal, which is a must for a successful phase demodulation based speed extraction. For this case, the so-called iterative strategy, shown in Fig. 1.16 (b), can be utilized. First, a rough speed is estimated using either the above standard phase demodulation approach [59], the TFR based method (to be introduced later) [60] or others. Second, the rough speed is used for the angular resampling of the signal. The speed fluctuations in the resampled signal are mitigated. Third, another standard phase demodulation course can be applied to extract the refined speed [55].

In addition to band-pass filters, signal decomposition methods can also be utilized to obtain the mono component signal, such as the empirical mode decomposition (EMD) [61] and Hilbert vibration decomposition (HVD) [62]. To obtain the demodulated phase, the Teager Kaiser Energy Operator (TKEO) [63] can also be used in conjunction with the Hilbert transform.

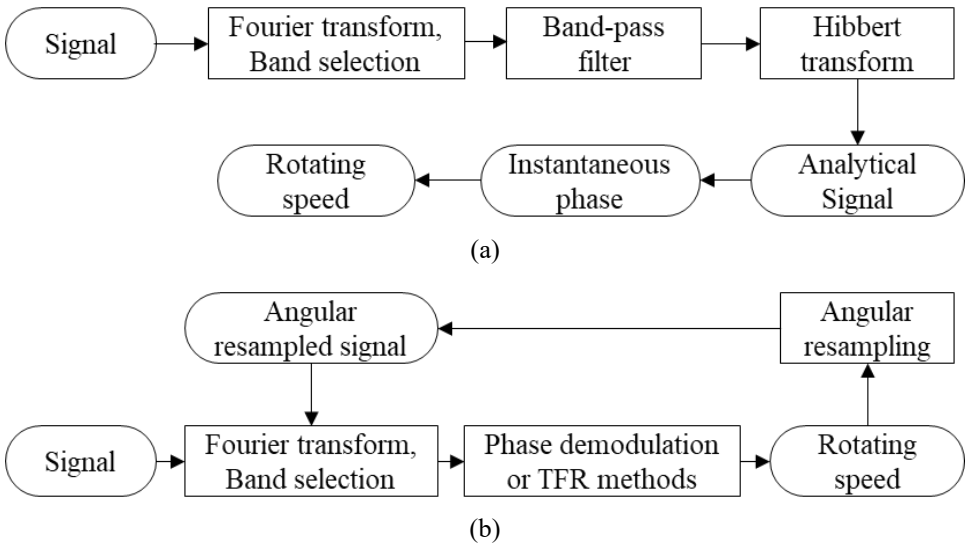


Fig. 1.16: Diagram of phase demodulation based speed extraction: (a) Standard method and (b) Iterative method [55], [56].

Time frequency representation based methods

Time frequency representation based (TFR based) methods attempt to track a harmonic component from the TFR of a vibration signal. The tracked harmonic is usually the speed or its multiples. The general process of TFR based method consists of two steps as illustrated in Fig. 1.17. First, the TFR of the raw vibration signals is generated. Second, a ridge or curve tracking method is utilized to track the harmonic component. The tracked harmonic is believed to be the speed or its multiples which is also known as the instantaneous frequency (IF). As such, efforts have been made for both steps to ensure a successful TFR based speed extraction algorithm.

For the first step, i.e., to obtain a clear TFR, Urbanek et al. [64] extracted the rotating speed of a wind turbine from the STFT of its vibration using a simple maximum tracking method. Peng et al. [65] used a Chirplet transform with a polynomial kernel for non-linear speed estimation of a rotor system. Gryllias et al. [66] utilized complex shifted Morlet wavelets to determine the instantaneous speed of a rotor system. In the second step, i.e., to track speed-related harmonic from TFR, Schmidt et al. [67] incorporated priori probabilistic knowledge about the instantaneous frequency of the target system to increase the robustness of the maxima tracking in the STFT for the speed extraction of a planetary gearbox. Iatsenko et al. [68] used an improved dynamic path optimization method to estimate the candidate path that best represents the speed component from the STFT.

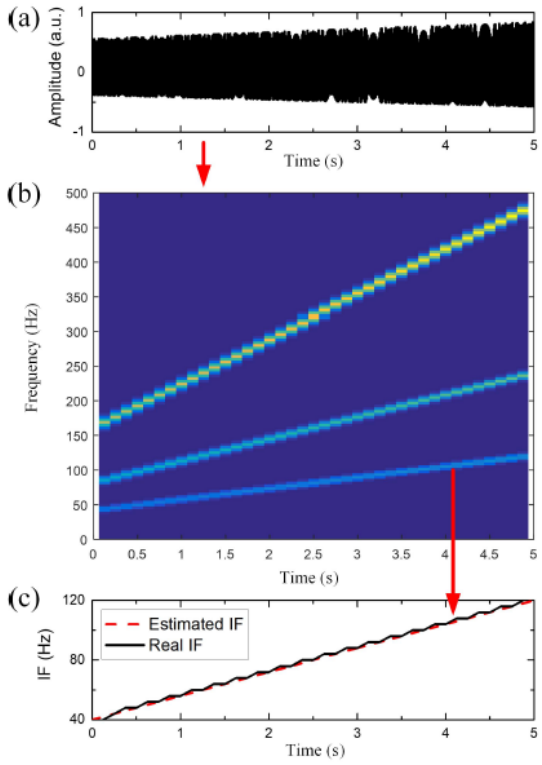


Fig. 1.17: Illustration of TFR based speed estimation: (a) Vibration signal, (b) TFR of the vibration signal and (c) Extracted speed from the TFR [69].

Either the phase demodulation based or the TFR based methods can achieve good speed extraction performances if they are carefully designed. However, both need to select case-sensitive parameters for every single piece of vibration signal to obtain either the mono-component signal or the clear TFR or the proper ridge tracking algorithm. Therefore, intensive expert knowledge and workload are required. Alternative methods free from these drawbacks are deep learning-based methods. They are reviewed in the following.

1.2.1.2 Deep learning-based methods

Speed extraction pertains to a sequence-to-sequence (seq2seq) learning problem, where a model learns from the inputs which are sequences or time series and outputs sequences too. The sequence and time series are taken equivalently and are used interchangeably in this thesis. The seq2seq learning has verified applications like speech recognition and linguistic translation. Commonly used models for seq2seq learning include CNN based models, RNN based models, and attention based models [70], [71]. Prabhavalkar et al. [72] compared the performances of these methods for speech recognition, and that found that their performances were comparative. Sutskever et al. [73] proposed a multilayered Long Short-term Memory (LSTM) to map the input sequence to the vector of a fixed dimensionality, and then another deep LSTM was used to decode the target sequence from the vector. This model showed impressive performance in translating English to French.

The abovementioned successes of seq2seq learning inspired us to seek possibilities of extracting rotating speed from vibration signals with the use of it. Unfortunately, we failed to find direct studies supporting this idea when this thought was initiated. However, the literature of the field of driving monitoring using smartphones contains rich and closely related studies [74], [75], [76]. Suppose a mobile phone or smartphone is adhered to a car. The smartphone collects data including

acceleration, gyroscope, and global positing system (GPS) data using embedded sensors. The sensory data can be used to estimate the condition of the car or the road. Fazeen et al. [75] used patterns of acceleration signals to estimate car running conditions like safe acceleration, extreme acceleration, safe deceleration, and extreme deceleration, and road conditions like bump, pothole, and uneven, rough, and smooth surfaces. This work pertains to a classification problem. It is different from the speed extraction, a regression problem. Hsu et al. [77] used the GPS data to estimate the altitude of a car, indicating whether a car was driving on or off a viaduct. The altitude was estimated with a dynamic Bayesian network. This study is similar to speed extraction, but the GPS data is a value type data, while the vibration data used for speed extraction is waveform data.

Another relevant study is by Gu et al. [78]. They proposed an LSTM model named the many-to-one LSTM (MO-LSTM) for car running speed extraction. The MO-LSTM considered acceleration and gyroscope signals as the input and outputted corresponding car running speed. The acceleration and gyroscope signals were collected using sensors embedded in a smartphone adhered to the car front window. Their study resembles the task to extract rotating speed from vibration signals of rotating machinery. However, even the reported MO-LSTM performed well in extracting the running speed of cars, although it suffered from the following two limitations. First, the speed information was only learned in the forward-time direction, but the backward-time direction was ignored. Second, the labeled speed was only utilized at a single timepoint but the remaining $(n-1)$ points were not used in a window of length n. In this thesis, the labeled speed means the real speed used in the modeling training. The length of the window represents the time length of the LSTM model. As a result, the speed information mining ability, and thus the speed extraction accuracy of the reported MO-LSTM model, leaves room for improvement when applied to extract rotating speed from vibration signals of rotating machinery.

1.2.1.3 Summary of existing studies

As reviewed in Sections 1.2.1.1 and 1.2.1.2, signal processing based speed extraction methods need intensive expert knowledge and labor. The deep learning-based methods are free from these limitations and therefore will be used in this thesis. The reported deep learning model, MO-LSTM [78], takes raw vibration signals as the input and provides directly the car running speed as the output. It resembles the task to extract rotating speed from the vibration signals of rotating machinery, and thus can act as a baseline model. However, the reported MO-LSTM did not adequately exploit speed-related information in vibration signals, thus leaving room for improving the speed extraction accuracy. Deficiencies of the reported MO-LSTM will be addressed in this thesis as the first research topic which is defined in Section 1.3 in detail.

1.2.2 Deep learning-based fault detection under varying speed conditions

Fault detection is usually understood as an unsupervised learning problem [79]. Only healthy data is available for developing fault detection models. This is fair, especially for newly commissioned machines. For these machines, the healthy data accumulates quickly, but we do not have a chance to collect the faulty data before a fault occurs. Furthermore, operating under faulty states is often not allowed by asset managers, but instead they want a fault detection algorithm ready before the occurrence of a fault, so that once a fault emerges it could be detected right away.

Fault detection is often achieved by first constructing a health indicator (HI) using either signal processing methods, conventional data-driven methods, or deep learning methods. The HI is compared with a predefined threshold to determine whether a fault has occurred or not. If the HI is smaller than the threshold, the machine is considered as healthy, otherwise faulty. An HI is a scalar that is supposed to be able the reveal the health state of a machine. It often has a smaller

value when the machine is healthy and a larger value if the machine is faulty. In this thesis, only deep learning-based fault detection methods are reviewed. Reviews on other methods can be found in [80], [81]. Existing deep learning-based fault detection methods can be categorized as residual based and feature based according to where the HI is extracted from. The residual based methods extract the HI from residuals, while the feature based methods extract the HI from features. They are introduced separately below.

Residual based methods first build a deep learning model to reconstruct its input or predict a few steps ahead of its input. The input is often the condition monitoring data such as vibration. The reconstruction or prediction error is known as residual. The residual is then used to construct HIs. The HI is ultimately applied to fault detection. The HI here is often simply the root mean square (RMS) of the residual [82], [83], or others such as the harmonic to noise ratio [32]. Commonly used deep learning models for data reconstruction include the AE and its variants, RNN and its variants, and ELM. Reddy et al. [82] used a 11-layer deep AE for fault detection in a large flight data. The input to the AE was the raw condition monitoring data including 13 modalities such as the actuator position, load, motor current and motor temperature. The RMS of the reconstruction residual was used for fault detection. Chandra et al. [84] used an AE shaped LSTM (AE-LSTM) network to reconstruct electrocardiography (ECG) signals. The RMS of the reconstruction residual was used for anomaly detection in ECG signals. The AE-LSTM was also used for the anomaly detection of sensor data in [85]. Maya et al. [86] reported an ensemble LSTM model to reconstruct the inputted condition monitoring data. The median of the reconstruction error of a sample served as the HI for the fault detection of a plant equipment. Dervilis et al. [87] used an AE with radial basis functions to reconstruct the inputted frequency response function (FRF) data. The Euclidean distance of the reconstruction residual was used to detect possible crack faults of wind turbine

blades. Chen et al. [32] used a 3-layer LSTM to predict one step ahead of its input, which was the raw vibration data. The harmonic to noise ratio of the prediction residual was used for the fault detection of gearboxes.

Feature based methods first build a deep learning model to extract the health condition related features from condition monitoring data. The features are often the activations of a certain layer of the deep learning model. The extracted features are then used to construct the HI using PCA, one-class classifier (OCC), or others. The AE, ELM, and their variants are usually used for feature learning. Michau et al. [88], [89] used a deep ELM to learn the features of inputted data. The features, which were the activations of the bottleneck layer of ELM, were inputted to an OCC to detect the faults of a generator rotor in a power plant. The inputted data was the 320-dimentional condition monitoring data including the rotor flux, partial discharge, and end winding vibration. Mao et al. [90] used a stacked AE to extract the common features of bearings. The extracted features were then processed using a support vector data description (SVDD) model for bearing fault detection. SVDD is a variant of the support vector machine (SVM) and worked like an OCC in [90]. Chen et al. [91] trained a CNN-shaped AE in a generative-adversarial (GAN) manner to detect faults of wind turbines. The model consisted of two parts, a generator and a discriminator. The generator was used to reconstruct the inputted data which was the spectra of raw vibration. The discriminator worked like an OCC to detect the occurrence of faults.

The abovementioned methods illustrate general pipelines of deep learning-based fault detection regardless of operation conditions, but pretty much about constant speed conditions. In the following, a specific review on fault detection under varying speed conditions will be provided.

The key to fault detection of rotating machinery under varying speed conditions, in addition to extracting fault sensitive HIs, is to address the effects induced by speed variation. As both speed variation and faults introduce AM and FM effects to vibration signals, and their effects are often overlapped in HIs. Effects of speed variation if not addressed, can lead to false alarms and/or missed alerts in fault detection [8], [52], [92]. Regarding when to address the effects of speed variation, the existing deep learning-based fault detection methods under varying speed conditions can be roughly classified into three types, i.e., pre-modeling methods, in-modeling methods, and post-modeling methods. The modeling herein means building a deep learning model for fault detection. Pre-modeling methods refer to addressing the effects of speed variation in prior, which is often achieved by preprocessing vibration signals using signal processing techniques. In-modeling methods refer to addressing the effects of speed variation by the deep learning model itself. Raw vibration data is inputted to the deep learning model. Post-modeling methods refers to inputting raw vibration data to a deep learning model for processing first, and then a subsequent step is applied to address the effects of speed variation. These three types of methods are reviewed separately below.

1.2.2.1 Pre-modeling methods

Pre-modeling methods address the effects of speed variation with the raw data using signal processing methods. The preprocessed data is then used by deep learning models or other models for fault detection. The FM can be removed with the widely used computed order tracking [93], [94]. The existing studies on removing AM are relatively limited. The recent studies [52], [92] usually follow the following thought. That is, divide the vibration with a certain form of its envelope, which is believed to show the AM effect. This method, however, may remove fault

signatures too as pointed out in [52]. Not mention that we need to carefully design a proper envelop for each piece of vibration signal.

1.2.2.2 In-modeling methods

In-modeling methods address the effects induced by speed variation by a deep learning model itself. The input of the deep learning model is the raw condition monitoring data. Martin-del-Campo et al. [50] learned a set of shift-invariant dictionaries for the sparse representations of vibration signals of wind turbines using dictionary learning. The Euclidean distance between the dictionary of vibration signals of unknown health states and that of the healthy state was taken as the HI for fault detection. The vibration data was collected every 12 hours over about 46 consecutive months. Each vibration sample lasted for 1.28 s. The rotating speed of the wind turbine varied between 700 rpm and 1200 rpm in the 46 months but was almost constant for each 1.28 s long vibration sample. The shift-invariant dictionaries were claimed to be independent of speed so that the effects of speed variation were removed. However, in [50], the distance between dictionaries contains not only the mechanical health states of wind turbines, but also the approximation errors for solving the nondeterministic polynomial-time hardness (NP hard) dictionary-learning problem. These errors can lead to false alarms in the detection. In addition, the dictionary needs to be learned and updated for any newly collected vibration data. This would exponentially increase the computation load.

Instead of using dictionary learning, Liang et al. [95] reported a sparse AE based method for pump fault detection. The pump worked under slightly fluctuating speed conditions (90 – 105 rpm). The sparse AE was utilized to reconstruct 15 types of condition data such as the temperature, speed, overall vibration, and pressure. The Mahalanobis distance (MD) of the reconstruction residual was

taken as the HI to detect possible faults of the pump. Jiang et al. [96] conducted a similar study, wherein a denoising AE was used for wind turbine fault detection based on the supervisory control and data acquisition (SCADA) data. However, in [95], [96], the effects of speed variation were not specifically considered and how the effects of speed variation were addressed were not explicitly indicated. When the speed fluctuated largely, their performances may be questioned. Additionally, in their studies, the inputted data was the value-type data, not the waveform vibration. Therefore, the fault detection performance of [95], [96] over the vibration data needs further investigation.

1.2.2.3 Post-modeling methods

Post-modeling methods address the effects induced by speed variation after deep learning modeling. One pipeline involves the use of a deep learning model to learn features from raw data first, and then remove the effects of speed variation from the features. Luo et al. [97] trained a stacked AE to select impulsive vibration segments of a machine tool under different working conditions, i.e., milling, drilling and so on. A set of speed independent features, namely, the operational natural frequencies, were manually extracted from the impulsive vibrations using the so-called dynamic identification algorithm. An HI was constructed based on the similarity of these features and was further used to detect faults of machine tools. However, as reported in [97], we need to manually extract features from impulsive vibration signals, making it less preferable in the big data era due to the intensive labor for feature extraction.

Another pipeline of post-modeling is to use a speed adaptive threshold for fault detection. We do not address the effects of speed variation in either the raw data, or the deep learning model, or the constructed HI, but design a threshold that changes with speed. Zhao et al. [98] employed a denoised AE to reconstruct the SCADA data of a wind turbine. The RMS of the reconstruction

residual was taken as the HI. An adaptive threshold series was manually designed to compensate the effects of speed variation and worked with the HI to detect faults of wind turbines. This method may be exhausting as we need to design an adaptive threshold for every single HI. The problem is that we might have multiple HIs for a single machine.

1.2.2.4 Summary of existing studies

Obviously, more labor work and expert knowledge are required for the pre-modeling and post-modeling methods. As such, in this thesis, we will focus on the in-modeling methods. As reviewed above, reported in-modeling methods are mainly based the AE and its variants [95], [96]. These studies address the effects of speed variation automatically with value-typed data. While we checked their performances with the waveform-type vibration, we found the resulting HI is still affected by speed variation. The resulting fault detection performance under varying speed conditions is therefore not as good as constant speed conditions. We address this problem in this thesis in detail as the second research topic, which is defined in Section 1.3.

1.2.3 Deep learning-based fault classification under varying speed conditions

Fault classification is usually understood as a supervised learning problem [38], [99]. It requires adequate labeled data for model development. Deep learning-based fault classification simply takes raw vibration data as the input to a deep learning model, which outputs the fault types directly. To date, deep learning models, such as the AE, restricted Boltzmann machine (RBM), FNN, CNN, ResNet and RNN and their variants were widely employed for fault classification of various rotating machines like bearings, gears, rotors, motors, computer numerical control (CNC) machines, wind turbines, and compressors. To name a few, Ince et al. [100] utilized a one-dimensional 1D-CNN to classify the faults of a motor. The input data was the current signal, and

the output was the fault type, i.e., either healthy or bearing cage fault. Shao et al. [101] used a deep AE to extract fault features from acceleration signals of a fixed-shaft gearbox. These features were then utilized for fault classification of that gearbox. More examples can be found in review papers [16], [38], [79], [99], [102], [103], [104].

In addition to using raw data directly, the raw data is sometimes transformed to other domains such as the frequency domain before being processed by deep learning models to alleviate the learning difficulty. Janssens et al. [105] employed a CNN to classify the faults of bearings. The fault types to be classified included the inadequately lubricated, outer race fault, imbalance, and combined faults. The data inputted to the CNN was the discrete Fourier transform spectra of acceleration signals. Jia et al. [25] used a stacked AE to conduct fault classification of bearings and planetary gearboxes. The input data was also the spectra of raw acceleration signals.

The abovementioned methods illustrate the general pipelines of deep learning-based fault classification regardless of operation conditions. They for sure work well under constant speed conditions, but the fault classification performances cannot be guaranteed because of speed variation when applied to varying speed conditions [94]. In the following, a specialized literature review on deep learning-based fault classification under varying speed conditions will be provided. Similar to fault detection, the existing methods for rotating machinery fault classification under varying speed conditions based on deep learning are also categorized as pre-modeling methods, in-modeling methods, and post-modeling methods, according to when to address the effects of speed variation. They are reviewed separately in the below.

1.2.3.1 Pre-modeling methods

Similar to fault detection, pre-modeling methods for fault classification address the effects of speed variation in prior. Existing pre-modeling methods for rotating machinery fault classification under varying speed conditions are broadly two pipelined. One type of pre-modeling methods is to preprocess non-stationary vibration signals such that the preprocessed signals can be well classified by deep learning models that were developed for constant speed conditions. The basic idea behind is to suppress the effects of speed variation on vibration signals. For example, Rao et al. [94] and Ma et al. [106] converted nonstationary vibration signals to stationary signals using order tracking and generalized demodulation, respectively. The obtained stationary signals were then inputted to a 5-layer FNN to classify faults of bearings and a fixed-shaft gearbox [94] and a deep ResNet to classify faults of a planetary gearbox [106], respectively. Wei et al. [107] divided vibration signals using corresponding speed signals to normalize the amplitudes of vibration signals. The normalized vibration signals were processed using an 11-layer CNN to classify the faults of a rotor system.

Another type of pre-modeling methods is to use time frequency representations (TFRs) of vibration signals. The TFRs instead of the raw time series or spectra are inputted to deep learning models for fault classification. Widely used TFRs include Short Time Fourier Transform (STFT) and Wavelet Packet Transform (WPT). For instance, Du et al. [108] inputted STFT of vibration signals to a CNN for the fault classification of bearings. Diego et al. [109] inputted WPTs of vibration signals to a CNN to classify fault types of a helical gearbox. However, different TFR parameters such as WPT decomposition layers would lead to different classification results. Some of these parameters are considered better than others. To address the effects of TFR parameters, Han et al.

[110] and Yuan et al. [111] suggested ensemble CNNs to fuse multi-level WPTs. That is, vibrations signals were decomposed into multiple WPTs with different decomposition levels. Each WPT was inputted to a single CNN for feature extraction. The extracted features of all CNNs were then fused to classify the faults of a planetary gearbox [110] and a wind turbine blade [111], respectively. Zhao et al. [112] shared the same idea, but instead of fusing the features extracted with a deep leaning model as in [110], [111], they fused the WPT coefficients directly. The fused coefficients were inputted to a deep ResNet for the fault classification of a planetary gearbox.

1.2.3.2 In-modeling methods

In-modeling methods provide an end-to-end fault classification scheme, i.e., the deep learning model takes raw vibration data as the input and outputs the final fault classification results. It does not require to either preprocess the raw data or post-process the classification results. An et al. [113] reported an LSTM model based on the so-called infinitesimal method. They borrowed the idea of finite element analysis (FEA). That is, cut a vibration sample collected under varying speed conditions into many short and sequential segments. The speed in each short segment was assumed constant. These short segments were fed into an LSTM model sequentially for bearing fault classification. However, this method broke the continuity of speed variation and might lead to abrupt jumps in learned features thus misclassification, especially for largely varying speed conditions. Liu et al. [114] introduced a multi-scale kernel ResNet, which integrated three branches of ResNets with different kernel sizes. It took raw non-stationary vibration signals as the input and directly outputted the fault classification results of a motor. Even the work in [114] has shown demonstrated performances with rotating machinery fault classification, it still suffers from following drawbacks. First, effects of speed variation on fault classification performances of deep

learning models are not unfolded. Second, how to address the effects of speed variation is not specifically investigated.

1.2.3.3 Post-modeling methods

Post-modeling methods address the effects of speed variation after training a deep model. For fault classification under varying speed conditions, post-modeling methods are pretty much based on transfer learning. The knowledge learned under one or more speed conditions (source domain) is transferred to another speed condition (target domain). Effects of speed variation are either mitigated or balanced during the transfer. For fault classification, transfer learning is preferable when the source domain has a large amount of labeled data, but the target domain has no or limited labeled data.

The transfer learning is often achieved through the following three steps [115]. Firstly, a deep learning model is trained in the source domain. Then the well-trained model is transferred to the target domain in either the instance, or feature, or the parameter level. Finally, the transferred model is fine-tuned in the target domain to adapt to the new speed condition. The fine-tuned model is used for fault classification in the target domain. We acknowledge that there are transfer learning techniques such as the domain adaption which do not train a model in the source domain first and fine-tune it at post in the target domain [116]. Instead, they train a model for the source domain and the target domain in a single step. Herein they are still counted in the post-modeling methods for the purpose of simplicity. Shao et al. [117] presented a transfer learning strategy based on a CNN for rotor fault classification under different speed conditions. The CNN was trained with a massive amount of labeled data under 2000 rpm (source domain). The learned parameters were used to initialize a same CNN in the target domain. The initialized CNN was then fine-tuned with

a limited amount of labeled data under 3000 rpm (target domain). The fine-tuned CNN was applied to classify rotor faults under 3000 rpm. Cao et al. [39] reported a domain-share CNN to conduct fault classification under varying speed conditions (target domain) through transferring knowledge of constant speed conditions (source domain). The transfer was realized with a term named maximum mean difference (MMD). It measured the distribution discrepancy between the source domain and the target domain. The MMD was added to the loss function of the source domain with a certain weight and was minimized through training. The domain-share CNN obtained fair fault classification accuracies for bearings and gearboxes in [39].

1.2.3.4 Summary of existing studies

Based on literature review in the above, the advantages and disadvantages of the existing deep learning-based fault classification methods are summarized and shown in Table 1.2. This summary also holds true for the deep learning-based fault detection methods when applicable.

Pre-modeling methods address the effects of speed variation in prior. The learning difficulty of successive deep learning models is reduced. Deep learning models for constant speed conditions could be directly adopted herein. However, intensive expert knowledge is needed to design proper signal processing methods to preprocess the data to address the effects of speed variation, and heavy labor work is required to implement such signal processing work. Post-modeling methods here refer to transfer learning methods for fault classification. A prerequisite of a successful transfer learning task is that the model must perform sufficiently well in the source domain. If a model does not perform well in the source domain, it will never have a chance to perform well in the target domain. The model in the source domain is essentially the model obtained in the in-modeling methods. Therefore, improving the in-modeling methods would contribute to a better

transfer learning task. Besides, the transfer learning may be exhausted when we are asked to transfer to many speed levels. We need to conduct the transferring for every single speed level. Indeed, the transfer learning is more frequently utilized to transfer knowledge across machines in or beyond a fleet in real applications for either fault detection [40] or fault classification [38], [118]. We do not focus on transfer learning in this thesis.

Table 1.2: Summary of existing deep learning-based fault detection and fault classification methods for rotating machinery under varying speed conditions

Category	Description	Advantages	Disadvantages
Pre-modeling	1) Effects of speed variation addressed in raw data 2) Input to deep learning models is preprocessed data	1) Reduced learning difficulty	1) Intensive expert knowledge and labor work required
In-modeling	1) Effects of speed variation addressed automatically by a deep learning model 2) Input to deep learning models is raw data	1) End-to-end learning 2) Effects of speed variation are addressed automatically	1) Difficult to design the deep learning model
Post-modeling	1) Effects of speed variation are addressed in the output of deep learning models 2) Input to deep learning models is raw data	1) Reduced learning difficulty if signal processing based methods adopted 2) Promising for cases wherein the speed conditions of the source domain and the target domain are different if transfer learning is adopted	1) Intensive expert knowledge and labor work required if signal processing based methods adopted 2) Performances rely on that of the source domain if transfer learning is adopted 3) May be exhausted when there are too many speed levels to transfer

The in-modeling methods are difficult to realize because the deep learning model needs to be carefully designed to address the effects of speed variation within the model. However, it does provide an end-to-end learning scheme among these three types of learning methods. We will focus on the in-modeling methods for fault classification in this thesis. The existing in-modeling methods [114] even showed encouraging performances with rotating machinery fault classification under varying speed conditions, they did not unfold the effects of speed variation on the fault classification performances of deep learning models, and also did not specifically address the effects induced by speed variation. We address these drawbacks in this thesis in detail as the third research topic which is defined in Section 1.3.

1.3 Research objective

The overall objective of this thesis research is to develop new deep learning models or improve the existing deep learning models for effective fault diagnosis of rotating machinery operated under varying speed conditions. Machines of interest are typical rotating machines such as bearings, gearboxes, and rotors. Condition monitoring data to be used is vibration. Based on the literatures reviewed in Section 1.2, we have the following three sub-objectives:

- Develop a deep learning model to extract the rotating speed from vibration signals.
- Develop a deep learning model to conduct fault detection of rotating machinery under varying speed conditions.
- Develop a deep learning model to conduct fault classification of rotating machinery under varying speed conditions.

To achieve the three sub-objectives, we have completed three research topics. They are defined in the following.

- Topic # 1 focuses on speed extraction. In Topic #1, a deep learning model named many-to-many-to-one bi-directional long short-term memory (MMO-BiLSTM) is proposed for this purpose. Limitations of the reported MO-LSTM model [78] are addressed. The proposed model consists of two parts: the many-to-many BiLSTM part (BiLSTM part) and the many-to-one LSTM part (LSTM part). The BiLSTM part learns speed related information from vibration signals in both forward-time and backward-time directions. The final speed is successively extracted via the LSTM part from the information learned by the BiLSTM part. The performance of the proposed model is validated using an internal combustion engine dataset, a rotor system dataset, and a fixed-shaft gearbox dataset. The results show that the proposed model achieves a higher speed extraction accuracy than reported models. Details of this topic are given in Chapter 3.
- Topic #2 focuses on fault detection. In Topic #2, a deep learning model named speed normalized autoencoder (SN-AE) is proposed for rotating machinery fault detection under varying speed conditions. Limitations of reported AE models [95], [96] are addressed. The proposed SN-AE consists of two branches, i.e., an AE branch, and a speed normalization (SN) branch, The input of the SN branch is the speed signal. The output of the SN branch is to multiply the vibration to normalize its amplitude to remove the effects of speed variation. The normalized vibration is then inputted to the AE branch for fault detection. Case studies over a planetary gearbox dataset, a fixed-shaft gearbox and a bearing dataset

validate the effectiveness of the proposed SN-AE. Details of this topic are provided in Chapter 4.

- Topic #3 focuses on fault classification. In Topic #3, an auxiliary branch named speed adaptive gate (SAG) is proposed for the existing deep learning models to improve their fault classification accuracy for rotating machinery under varying speed conditions. Drawbacks of reported CNN and ResNet [114] are addressed. The proposed SAG is an auxiliary branch for existing deep learning models. It takes speed signals as the input. The output of the SAG multiplies existing deep learning models to control the information flow in these models. The SAG values change adaptively with speed, such that the fault information imbalance induced by speed variation is mitigated. Case studies with two baseline models, i.e., a CNN and a ResNet, over two experimental datasets, i.e., a planetary gearbox dataset and a fixed-shaft gearbox dataset, show the effectiveness of the proposed SAG and its superiority over the existing methods. Details of this topic are shown in Chapter 5.

Relationships among the three topics are illustrated in Fig. 1.18. Topic #1 extracts the rotating speed from vibration signals which are measured from a rotating machine. The extracted speed together with vibration signals are used for the model development of Topic #2 and Topic #3. Topic #2 focuses on fault detection. If a fault is detected, Topic #3 will identify its type and its severity through fault classification. Note the speed used in Topic #2 and Topic #3 can also be measured besides the extracted through Topic #1.

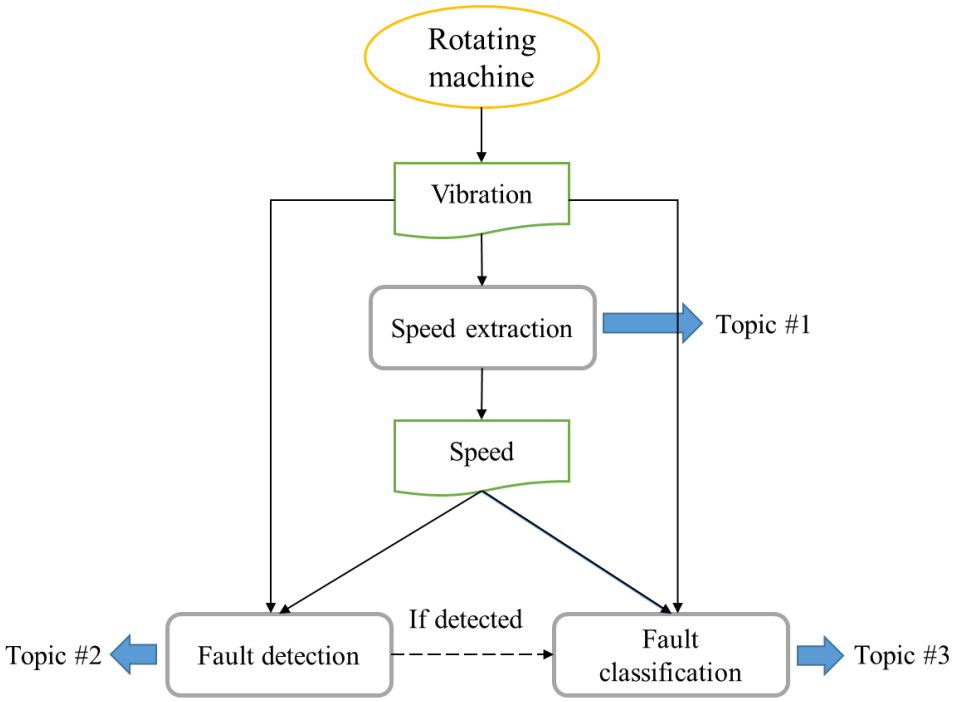


Fig. 1.18: Relationships among research topics in the presented thesis.

Primary assumptions for this thesis are listed as follows.

- The load condition of the interested rotating machinery is constant.
- The amount of vibration signals is balanced across fault types and speed.
- Adequate data is available.

The presented research work reported in this thesis promotes the frontier of deep learning-based fault diagnosis, especially for varying speed conditions. The effects of speed variation are addressed to some extent for deep learning-based fault diagnosis. Researchers in this field can take this thesis as a baseline to get promoted. Engineering practitioners may use the proposed methods in this thesis in their applications.

1.4 Thesis organization

This thesis consists of 6 chapters. Chapter 1 provides the background, literature review and objective of this thesis research. Chapter 2 describes the fundamentals of deep learning. Chapters 3 through 5 display materials regarding research Topics #1 – #3, respectively. Chapter 6 summarizes the entire thesis research and suggests future work.

This thesis is written following the paper-based template and satisfies the minimal formatting requirements of the University of Alberta.

2. Deep learning fundamentals

Chapter 1 has briefly introduced the concept of deep learning. This chapter will provide fundamentals of deep learning, including artificial neural networks, typical deep learning models and how to build a deep learning model. The introduced models in this chapter will be used in Chapters 3, 4 and 5 as needed.

2.1 Artificial neural network

As introduced in Section 1.1.3, popular deep learning models are pretty much based on artificial neural networks (ANNs). As such, this section will first provide fundamentals of ANNs, such as the structure of an ANN and how to train an ANN.

2.1.1 Structure of neural networks

An ANN is a network-like machine learning model that consists of inter-connected artificial neurons. The artificial neurons are brain-inspired systems which are intended to replicate the way that we humans learn. The connection between a pair of neurons has a connection weight. Each neuron represents a mapping between multiple inputs and a single output. The output of a neuron depends on the sum of the inputs and an activation function. Fig. 2.1 illustrates the diagram of a single neuron. Suppose it has n inputs $x = (x_1, x_2, \dots, x_n)^T$, and weights connected each input are $w = (w_1, w_2, \dots, w_n)^T$. The sum of weighted inputs is,

$$z = w^T x + b \tag{2.1}$$

where, b is a constant referring as the bias. It allows the activation function to be shifted to the left or right, to better fit the data.

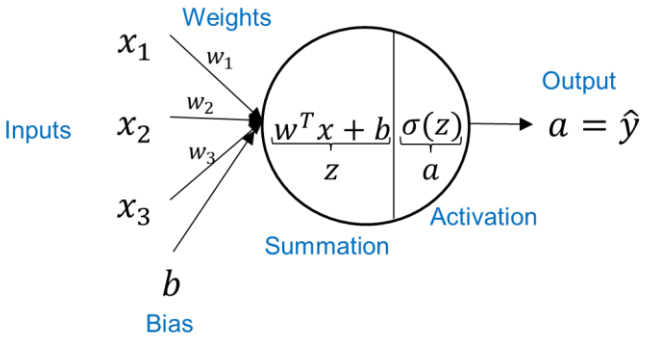


Fig. 2.1: Diagram of a single neuron.

The activation function is often a nonlinear transform of the sum z. It defines how the weighted sum of the input is transformed into an output,

$$a = \sigma(z) \tag{2.2}$$

where, σ refers to the activation function and a is the output of the neuron. The output of this neuron can be the input of successive neurons, or simply the output of the whole ANN. Usually used activation functions are the linear, rectified linear unit (ReLU) and sigmoid functions. They are mathematically represented as,

$$\sigma(z) = \begin{cases} z & \text{Linear} \\ \max(0, z) & \text{ReLU} \\ \frac{1}{1+e^{-z}} & \text{Sigmoid} \end{cases} \tag{2.3}$$

The linear activation has the least nonlinearity, the sigmoid activation has the highest nonlinearity and the ReLU is in the middle. Besides these three commonly used activation functions, there are

other types of activations such as the Gaussian, exponential linear unit (ELU) and hyperbolic tangent (tanh). See [26] for details.

Neurons in an ANN are organized in layers. Fig. 2.2 illustrates the structure of a simple but commonly used ANN, the feedforward neural network (FNN). It consists of four layers. The first layer is the input layer, the last layer is the output layer, and other layers are hidden layers. Information flows in one way only, as shown in arrow directions. Each neuron receives information only from neurons in the previous layer. The inputs to each neuron are weighted outputs of neurons in the previous layer.

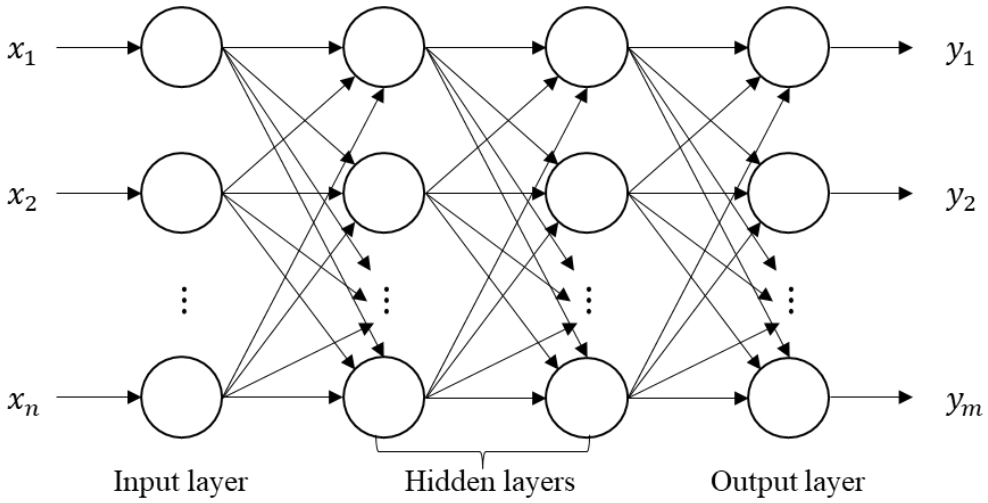


Fig. 2.2: Structure of a feedforward neural network.

The MLP can have different number of hidden layers. If an MLP contains less than two hidden layers, we say it is a shallow neural network. If an MLP has two or more hidden layers, it is deeded as a deep neural network (DNN), which is the most popular deep learning model.

2.1.2 Training of neural networks

An ANN can be considered to be a mapping from R^n to R^m, i.e., from $x = (x_1, x_2, \dots, x_n)^T$ to $y = (y_1, y_2, \dots, y_m)^T$. Here, n is the number of input variables and m is the number of output variables. For a neural network with a fixed structure, the connection weights uniquely define a specific mapping. Thus, a specific neural network can be considered as a function, though not in a mathematical form. Given the input x, the function can be written as $f(x, \theta)$, where θ refers to the collection of all weights and biases in the neural network. For simplicity, weights and biases together will be briefed as weights hereafter.

An ANN is designed and trained to make it behave in a certain way. For given inputs, $x = (x_1, x_2, \dots, x_n)^T$, it should provide outputs that are close to the expected output $y = (y_1, y_2, \dots, y_m)^T$ as much as possible. This is achieved through the optimization of the connection weights. The optimal weights are obtained with the use of data. The process of finding optimal weights with data is the so-called neural network training. The training is essentially a process of "learning" knowledge from data. The learned knowledge is stored in the neural network, specifically, reflected by the values of weights. Training a neural network requires three key gradients, i.e., dataset, loss function, and optimization method. They are introduced briefly below.

2.1.2.1 Dataset

A dataset is a set of input data and sometimes corresponding expected output data y. An entry of *x* and *y* is called a sample or an observation. A dataset often contains more than one sample. The expected output *y* is also known as true value, real value, or label. A dataset with labels provided

is a labeled dataset, otherwise an unlabeled dataset. Hereafter, we will take the labeled dataset as an example to proceed.

Prior to training, a dataset is often split into nonoverlapped subsets, including a training set and a test set, and sometime a validation set. The training set is the data used to train the neural network to find the optimal weights. The test set is to test the performance of a well-trained neural network over never seen data in the training set. The performance refers to how good a neural network achieving its intended goal. Commonly used performance metrics are the accuracy for classification tasks and the mean square error (MSE) for regression tasks. The validation set if exits is utilized before testing for the purpose of optimal hyperparameter selection. Hyperparameters are parameters that are necessary for a neural network, but cannot be learned through training, such as the number of layers, number of neurons and type of activation functions. Optimal hyperparameters are usually selected through trial and error. That is, train neural networks with different hyperparameter values, and compare their performances over the validation set, and optimal hyperparameter value is returned when the validation performance is the best.

2.1.2.2 Loss function

Given input x, a loss function measures the discrepancy between the output of a neural network $\hat{y} = f(\theta, x)$ and the corresponding label y. If the values of y are continuous, the loss function is often a quadratic function,

$$L = \frac{1}{2}(y - \hat{y})^2 \tag{2.4}$$

If the values of y are categorized, the loss function is often a cross-entropy function,

$$L = -\hat{y}\log(y) - (1 - \hat{y})\log(1 - y) \tag{2.5}$$

If a dataset contains multiple samples, the overall loss is the average loss of all samples.

Training a neural network is to minimize the loss function over the training set and returns optimal parameters θ^*. Note it is not enough to minimize the training loss only. The ultimate goal is to minimize the loss over the test set. The problem is that a small training loss does not guarantee a small test loss. This can be interpreted with the following analogy. Suppose a student is taking a course and he/she wants to get an "A". He/she works hard to learn from the textbook, lectures, and tutorials (training), and thus can solve the problems in the textbook correctly (training loss minimized). Even these bring high chances but not assure that he/she will perform well with the final exam (test loss minimized) which is necessary to be graded an "A", because of possible mental stresses, difficulty levels of the exam problems and so on. Indeed, in real applications, we care more or even only care about model performances over the test set as this illustrates how a model will perform after launched.

The ability of a model to perform well over the test set is called the generalizability. Good generalizability means that both the training loss and the test loss are small. Poor generalizability may be caused by underfitting or overfitting. Underfitting means a model does not fit the training data well. It occurs when both the training loss and the test loss are large. To mitigate underfitting, one can increase the model complexity or even design new model structures. Overfitting means a model fits the training data too much. It occurs when the training loss is sufficiently small, but the test loss is large. We can gather more training data, reduce the model complexity, or add regularizations to avoid overfitting. The regularization is defined as "any modifications to a learning algorithm that is intended to reduce its test error but not its training error" [26]. Often

used regularization methods include penalizing parameters (e.g., adding L1 norm or L2 norm to the loss function), adding noise to inputs, outputs or weights, dropout, and early-stop [26].

2.1.2.3 Optimization method

Training a neural network is essentially an optimization problem. The loss function is the target function to be optimized. However, the loss function is usually nonconvex and has multiple minima, and thus difficult to be optimized, especially when the number of layers is deep. The generic approach to minimize the loss function is the gradient decent method, which is usually organized in a back propagation manner for neural networks. The back propagation contains a two-pass procedure:

- Forward pass: The current parameters θ are fixed, and the predicted values $\hat{y} = f(\theta, x)$ are calculated.
- Backward pass: The errors $\delta = y - \hat{y}$ are computed, and back-propagated layer by layer.

The back propagated errors are then utilized to calculate the gradient $\frac{\partial L}{\partial \theta}$ for each parameter. Details of finding the gradient can found in Chapter 11 of [119]. The gradient is used to update the parameter iteratively following the gradient descent manner as follows,

$$\theta^{(r+1)} = \theta^{(r)} - \varepsilon \frac{1}{N_t} \sum_i^{N_t} \frac{\partial L_i}{\partial \theta} \tag{2.6}$$

where, ε is a hyperparameter referring to the learning rate, $\theta^{(r)}$ are the parameter values at iteration r, L_i is the loss of sample i in the training set, and N_t is the number of training samples used to update the parameters at a single iteration. In the context of deep learning, the iteration is also called the epoch. The number of samples N_t of each epoch can have multiple options. If $N_t = 1$, i.e., only a single training sample is used at a time, the optimization algorithm is called the

stochastic descent or online method. If $N_t = N$, i.e., the entire training set is used at a time, the optimization algorithm is called the batch or deterministic method. The stochastic method often returns unstable updates, while the deterministic method may overwhelm the CPU or GPU memory. A trade-off is the so-called minibatch method, which uses more than one but less than all the training samples to update the parameters at a time. Typical batch sizes range from 32 to 256, and better to be the power of 2.

The gradient descent method can be slow. Reasons can be improper initial values, improper search directions or improper learning rate. They are essentially the three key ingredients for gradient based optimization [120]. Efforts have been made to accelerate the training from all these three aspects and are briefly introduced below.

The initial parameter values are typically drawn randomly from a Gaussian or uniform distribution. Using either the Gaussian or the uniform distribution does not affect much but the scale of the initial distribution matters [26]. Good options for the initial scale include the normalized initialization and sparse initialization. The former one initializes weights of a fully connected layer with n inputs and m outputs using a scale of $\sqrt{\frac{6}{m+n}}$. The later one assures exactly k non-zero wights to avoid extremely small weights when the layers become large. One more method, named layer-wise-greedy-pretraining which was proposed by Hinton [121], can also be adopted for initialization. This method consists of two steps. The first step pretrains two successive layers to reconstruct the input of the first layer. The learned weights are to initialize the parameters in the successive finetuning step.

The search direction is related to the gradient. We can simply use the current gradient as the search direction as shown in Eq. (2.6). One modification is the momentum [26]. It preserves past directions for the current move to avoid instability induced by current gradient. A variable v is introduced to accumulate an exponentially decaying moving average of past gradients and continues to move in their directions,

$$v^{(r+1)} = \alpha v^r - \varepsilon \frac{1}{N_t} \sum_i^{N_t} \frac{\partial L_i}{\partial \theta} \tag{2.7}$$

$$\theta^{(r+1)} = \theta^{(r)} + v^{(r+1)} \tag{2.8}$$

where, $\alpha \in [0,1]$ is a hyperparameter that determines how much past gradients affect the current direction.

The learning rate has a significant impact on model performances and need to be carefully set for every single model. Usually, the learning rate is a fixed constant ranging from 0.001 to 0.1. A large learning rate can speed up the training but also bring risks of missing the optima or even leads to non-convergence. A small learning rate can somewhat assure convergence but may converge to local optima, and the convergence speed is slow. One modification is to decay the learning rate in in terms of epochs [122]. A larger learning rate is adopted at early epochs to speed up the training and decay this value at late epochs to avoid missing the optima. Another modification is to use different learning rates for different parameters. The rationale is that the loss function is often sensitive to some directions in the parameter space and insensitive to others. For this purpose, quite a few algorithms like the AdaGrad, RMSProp, and Adam were designed to adaptively assign different learning rates for different parameters [26].

2.2 Typical deep learning models

As introduced in Section 1.1.3, neural networks have three typical structures. They are the feedforward neural network (FNN), convolutional neural network (CNN) and recurrent neural network (RNN). The definition and general structure of the FNN has been introduced in Section 2.1. Here, we will not repeat the description of the FNN, but a special FNN, the so-called autoencoder (AE) will be introduced. After that, structures of the CNN and the RNN will be introduced.

2.2.1 Autoencoder

An autoencoder (AE) is a neural network that attempts to copy its input to output [26]. During this process, a new representation with lower dimensions than raw input signals can be learned. The new presentation is also referred as features extracted from the input. A typical structure of an AE consists of an input layer, a hidden layer, and an output layer, as shown in Fig. 2.3. The input layer and the hidden layer form the encoder part. The hidden layer and the output layer form the decoder part.

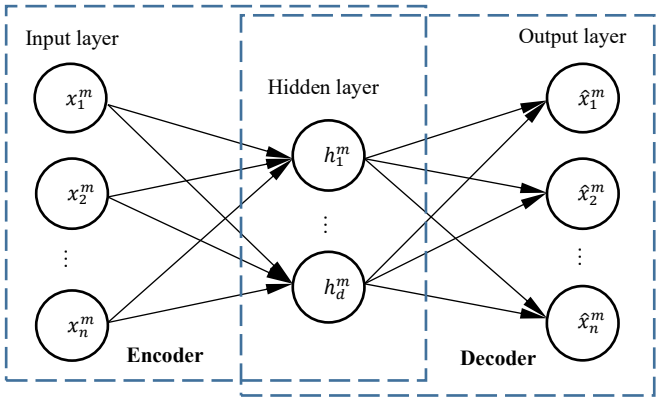


Fig. 2.3: Typical structure of an autoencoder [7].

The encoder learns features or a new representation of the input. For each measured signal x^m from a dataset $\{x^m\}_{m=1}^M$ of rotating machinery, the encoder vector h^m is defined as,

$$h^m = f_{\boldsymbol{\theta}}(x^m) \tag{2.9}$$

where, $f_{\boldsymbol{\theta}}$ is the encoder function and $\boldsymbol{\theta}$ is the weight and bias matrix in the encoder part. The decoder part reconstructs the raw data,

$$\hat{x}^m = g_{\boldsymbol{\theta}'}(h^m) \tag{2.10}$$

where, $g_{\boldsymbol{\theta}'}$ is the decoder function and $\boldsymbol{\theta}'$ is the weight and bias matrix in the decoder part. The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are determined by minimizing the following loss function,

$$L\left(\boldsymbol{\theta}, \boldsymbol{\theta}'\right) = \frac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|^2 \tag{2.11}$$

where, $\mathbf{x} = \{x^m\}_{m=1}^M$ and $\hat{\mathbf{x}} = \{\hat{x}^m\}_{m=1}^M$.

The AE is going be used in Chapter 4 for fault detection.

2.2.2 Convolutional neural network

A convolutional neural network (CNN) is a type of neural networks that uses convolution operation in at least one of its layers [26]. It is specialized for processing data with grid-like topology, like 1D time series data and 2D image data. Based on the topology of data, CNNs can be categorized as 1DCNNs and 2DCNNs. In the field of PHM, 1DCNNs are widely employed as we often encounter with 1D vibration signals. In this thesis, the term CNN refers to 1D-CNN if not specifically indicated. A CNN is usually composed of three types of layers, i.e., convolutional layers (Conv), pooling layers, and fully connected layers, as shown in Fig. 2.4(a). The convolutional layer is the layer the uses the convolution operation. Suppose we have a time series vector $\boldsymbol{x} \in \mathbf{R}^{1\times n}$ as the input, the output of the convolution layer is [26],

$$\boldsymbol{xc} = \mathbf{K} * \boldsymbol{x} + \boldsymbol{b} \tag{2.12}$$

where, * denotes the convolution operator, $\boldsymbol{b} \in \mathbf{R}^{m\times 1}$ is the bias vector, $\mathbf{K} \in \mathbf{R}^{m\times k}$ is the kernel matrix, and $\boldsymbol{xc} \in \mathbf{R}^{m\times n}$ is the feature matrix learned by the convolutional layer. Here, the integer *m* refers to the number of kernels which is also known as number of channels, and *k* refers to the size of the kernel. The process of convolution operation is illustrated in Fig. 2.4(b). The convolution operation enables sparse interactions, parameter sharing and equivariant representation, and thus brings benefits of invariance to data translation, and reduced network size [26], [123].

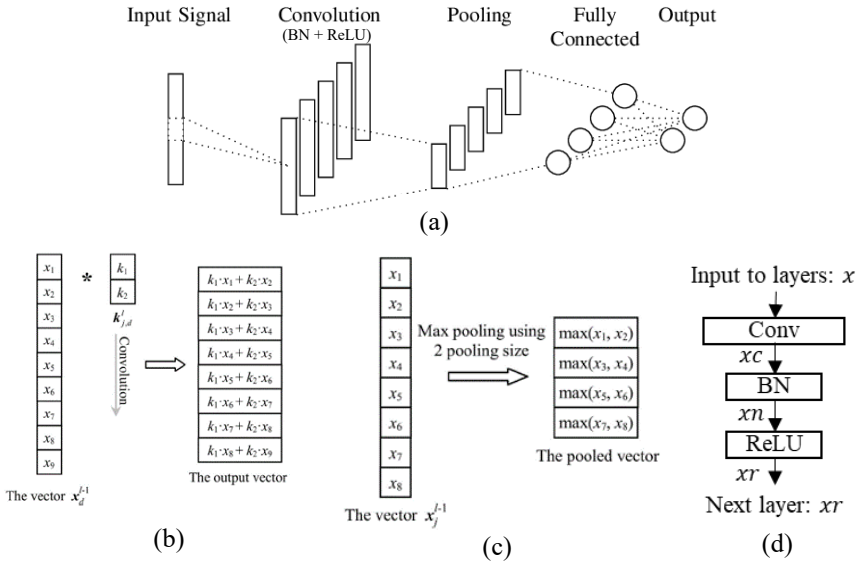


Fig. 2.4: Convolutional neural network: (a) Typical architecture, (b) Convolution operation, (c) pooling operation and (d) Basic building block [38], [123].

A convolutional layer is often followed by a batch normalization (BN) layer and a rectified linear unit (ReLU) activation layer [123],

$$\boldsymbol{xn} = BN(\boldsymbol{xc}) \tag{2.13}$$

$$\boldsymbol{xr} = ReLU(\boldsymbol{xn}) \tag{2.14}$$

where, $\boldsymbol{xn}$ and $\boldsymbol{xr}$ are the outputs of the BN layer and the ReLU layer, respectively. Their dimensions are identical to $\boldsymbol{xc}$. The BN layer is to speed up the convergence and improve the generalization [124]. The ReLU layer is to prevent possible gradient saturation [125]. These three layers form a basic building block for CNNs, as shown in Fig. 2.4(d).

A pooling layer down-samples its input along the spatial dimensionality,

$$\boldsymbol{v} = pool(\boldsymbol{h}) \tag{2.15}$$

where, the $pool$ indicates the pooling operation, which is illustrated in Fig. 2.4(c). The pooling operation replaces the output of the pooling layer with nearby statistic of the input, such as the maximum, minimum and average [26]. The pooling layer outputs length-shortened data, and thus reduce computation load for successive layers. A fully connected layer flattens features learned by preceding layers including the convolutional layer. It is identical to the layers in an FNN.

A CNN may suffer from the performance degradation problem when it goes very deep. The residual network (ResNet), a variant of the CNN, has been developed to address this problem through shortcut connections. Fig. 2.5 shows the typical building block of a ResNet. It is composed of two branches, i.e., a residual branch and an identity branch [124]. The residual branch is shown as the two convolutional layers. The identity branch is shown the shortcut connection. The residual branch learns a non-linear mapping, the so-called residual $F(x)$, from the input x. The identity branch provides an identity mapping of the input x. The overall mapping learned by the ResNet building block is [124],

$$H(\boldsymbol{x}) = F(\boldsymbol{x}) + \boldsymbol{x} \tag{2.16}$$

The ReLU activation is then applied to $H(\boldsymbol{x})$ before going to the next layer,

$$\boldsymbol{xr} = ReLU\big(H(\boldsymbol{x})\big) \tag{2.17}$$

Note that the Eq. (2.16) requires a same dimensionality of $F(x)$ and x. If not satisfied, a linear projection or zero padding to x can be performed to match the dimension of $F(x)$ [124]. In this thesis, the linear projection will be adopted wherever needed. For simplicity, the identity will be always written as x even such actions taken.

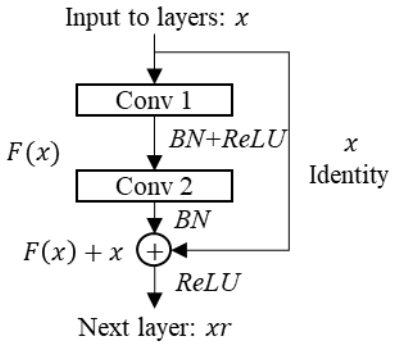


Fig. 2.5: Typical building block of the ResNet [124].

Both the CNN and the ResNet will be used for fault classification in Chapter 5.

2.2.3 Recurrent neural network

Recurrent neural networks (RNNs) are a family of neural networks for processing sequential data. In a sequence, one data point is related to previous data points. FNNs do not consider the dependency among data points. RNNs are designed to address this problem. They store the states or information of previous inputs to generate the next output of the sequence. The typical structure of RNNs is illustrated in Fig. 2.6. Visually, RNNs have connections not only among layers, but also within layers, as compared to FNNs which only have connection among layers. In Fig. 2.6, h is the hidden state, o is the output, L is the loss, U, V and W refer to weights to be learned.

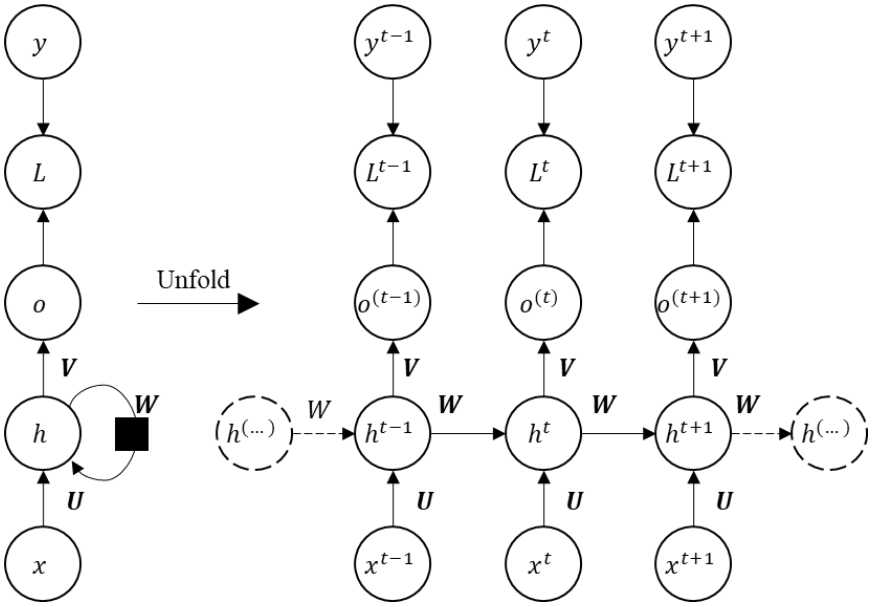


Fig. 2.6: Typical structure of a recurrent neural network [26].

An RNN usually suffers from the gradient vanishing or gradient exploding problem when dealing with long-term sequences. To address this problem, the long short-term memory (LSTM) model, which is a variant of the RNN, is developed with the use of gating mechanism [126]. An LSTM cell contains three gates as shown in Fig. 2.7. The three gates include a forget gate, an input gate, and an output gate. The three gates work with the inputs, which include the current data x_t, previous cell state c_{t-1} and previous cell output a_{t-1}, to an LSTM cell to determine the states c_t and output of the cell a_t. The state and the output of the current LSTM cell can go to a successive LSTM cell.

The forget gate determines how much information to be forgotten from the previous cell state. The forget gate is obtained as follows,

$$f_t = \sigma(W_f[a_{t-1}, x_t] + b_f) \tag{2.18}$$

where W_f is the weight matrix of the forget gate, b_f is the bias vector, and σ is the sigmoid function. The forget gate f_t is a matrix with all entries ranging from 0 to 1.

A candidate cell state for current time is calculated through a $tanh$ function,

$$\tilde{c}_t = tanh\,(W_c[a_{t-1}, x_t] + b_c \tag{2.19}$$

where W_c is the state weight matrix and b_c is the corresponding bias vector.

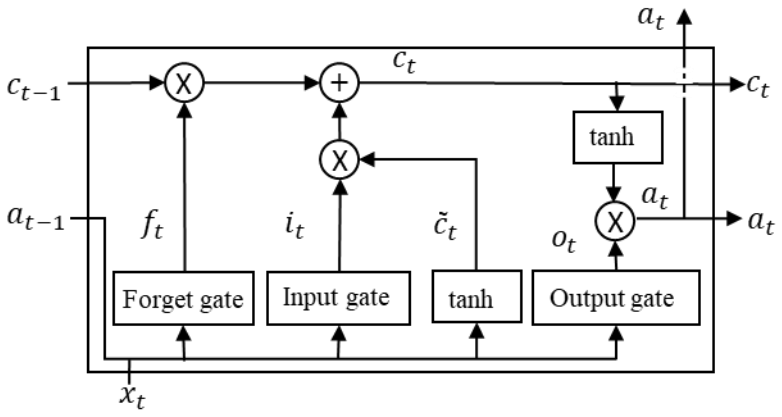


Fig. 2.7: Construction of an LSTM cell [26].

The input gate decides how much information of the candidate state to store in the cell,

$$i_t = \sigma(W_i[a_{t-1}, x_t] + b_i) \tag{2.20}$$

where W_i is the input gate weight matrix and b_f is the bias vector.

The state of the LSTM cell is then determined as,

$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \tag{2.21}$$

where the operation $\times$ means the element-wise multiplication.

The output of the LSTM cell is based on the cell state. The output gate determines how much information to output from the cell state,

$$o_t = \sigma\left(W_o[a_{t-1}, x_t] + b_o\right) \tag{2.22}$$

where W_o is the output gate weight matrix and b_o is the bias vector. The output of the LSTM cell is then determined as,

$$a_t = o_t \times \tanh\left(c_t\right) \tag{2.23}$$

An LSTM model is a sequence of LSTM cells, as shown in Fig. 2.8(a). LSTM cells at different time points share the same parameters, i.e., same weights and biases. The number of LSTM cells, n, also the number of time points, is defined as the window size. In Fig. 2.8, $x_j(j = t - n + 1, \ldots, t - 1, t)$ represents the inputted data at time j, c_j and a_j denote the cell state and activation at time j, v_j means the real (label) output at time j, and $\hat{v}_j$ means the estimated output at time j, which is equal to the activation of the corresponding LSTM cell, that is,

$$\hat{v}_j = a_j \tag{2.24}$$

where a_j is calculated with Eq. (2.23). Given the real output v_j at time j, the loss function of the LSTM model is,

$$L(\Theta) = \frac{1}{n}\sum_j \left(v_j - \hat{v}_j\right)^2 \tag{2.25}$$

where Θ denotes the parameter set of the LSTM model, which contains all the weights and biases of the model.

The LSTM models can be classified into many-to-many (MM), one-to-many (OM) and many-to-one (MO) modes according to model structures. The model shown in Fig. 2.8(a) follows an MM

mode. In this model, the outputs of LSTM cells at all time points are employed to build the loss function (Eq. (2.25)). If we only keep the input at time point $(t-n+1)$ in Fig. 2.8(a), the model becomes an OM mode. If only the output of the LSTM cell at current time t is considered, the model structure is an MO mode, as shown in Fig. 2.8 (b). The loss function of the MO mode is then changed to,

$$L(\Theta) = (v_t - \hat{v}_t)^2 \tag{2.26}$$

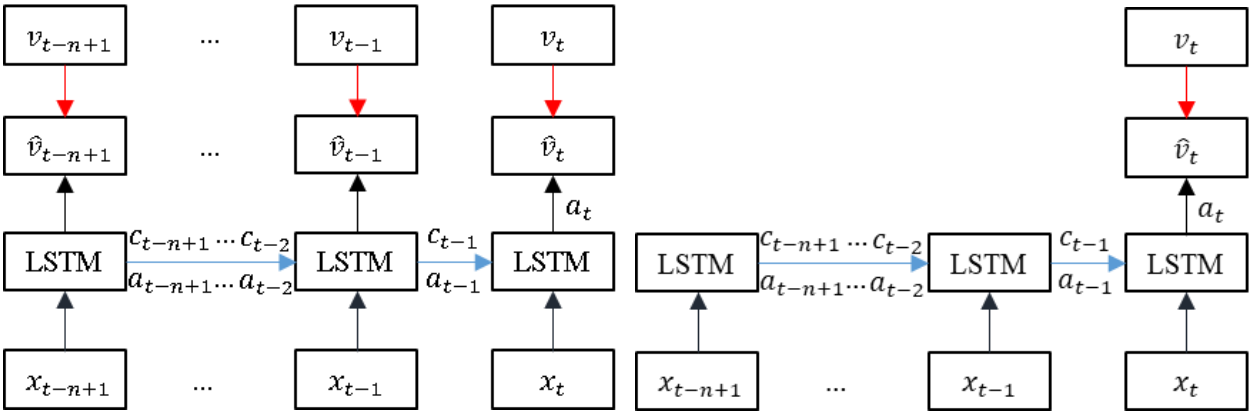


Fig. 2.8: Examples of LSTM models: (a) Many-to-many mode and (b) Many-to-one mode [26].

The LSTM model can also be classified into the unidirectional LSTM model and the bi-directional LSTM (BiLSTM) model. Models shown in Fig. 2.8 are unidirectional LSTM models. The information flows in the forward time direction only. The BiLSTM allows information flowing in both the forward time direction and the backward time direction. An example of the BiLSTM model is given in Fig. 2.9. The "LSTM F" and "LSTM B" are LSTM cells in the forward time and backward time directions, respectively. For the forward time direction, the LSTM cell state $\vec{c}_j$ and activation $\vec{a}_j$ are the same as a unidirectional LSTM model. For the backward direction, the LSTM

cell state $\overleftarrow{c}_j$ and activation $\overleftarrow{a}_j$ can also be determined using Eqs. (2.22) and (2.24) but in a reverse direction. The output of a BiLSTM model is the merge of the forward activation $\vec{a}_j$ and the backward activation $\overleftarrow{a}_j$. The merge operation can be max, min, average, multiply and concatenate.

For the simplicity of description, hereafter if not specified the LSTM model means a unidirectional LSTM model by default. The LSTM is going to be used in Chapter 3 for speed extraction.

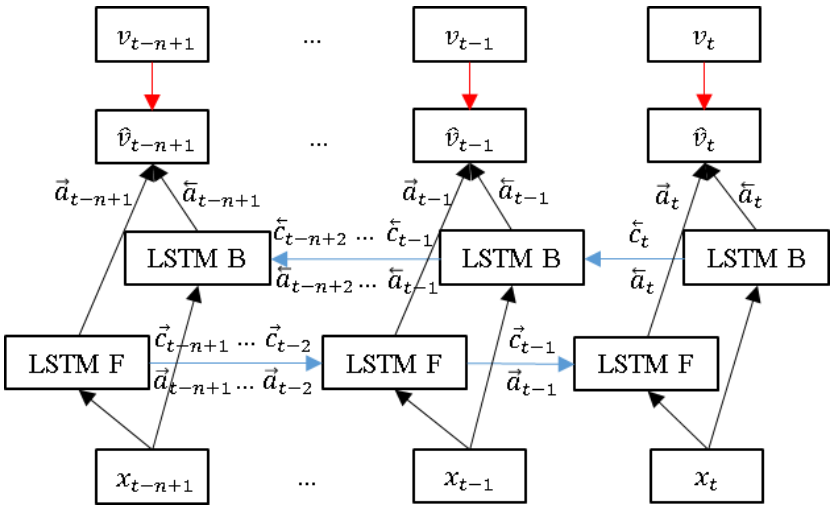


Fig. 2.9: Illustraioin of a bi-dierectional LSTM [26].

2.3 Building a deep learning model

Sections 2.1 and 2.2 provide fundamentals of deep learning models. This section will provide a general guideline of how to build a deep learning model to resolve tasks. The recipe of building a deep learning model usually consists of following steps, i.e., define the problem, collect data, choose model, preprocess data, split data, compile model, train model, evaluate model, freeze, modify or choose new model and apply the model [26], [127], as shown in Fig. 2.10.

Define the problem: Identify what is the problem to be solved. Is it a supervised learning or unsupervised learning problem? Is it a regression problem or classification problem? A good understanding of the problem will guide us to collect data and choose models.

Collect data: Collect data as much as possible. If we are working on a supervised problem, most efforts will be devoted to collecting labels.

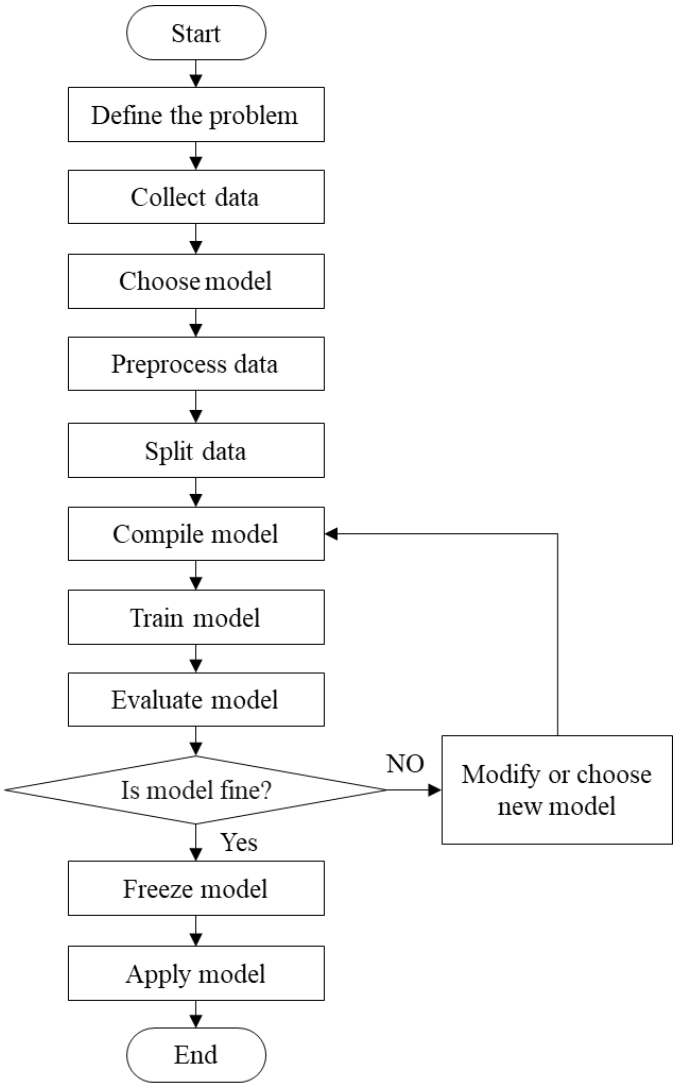


Fig. 2.10: Flowchart of building a deep learning model [127].

Choose model: Choose a model that best fits the problem and the collected data. Designing a new model is also possible but it is always encouraged to try out on-shelf models first.

Preprocess data: Select relevant data. Clean selected data. Reshape the data to fit in the model structure. Segment the data. Convert the data into other domains and so on.

Split data: Split the data into a training and a test set, and sometimes a validation set.

Compile model: Configure the loss function. Select the optimization algorithm. Assign the learning rate and more.

Train model: Train the model to learn parameters with the training set.

Evaluate model: Evaluate the trained model with the validation set if we have one. If not, the test set can be used over here.

Freeze, modify or choose new model: Check whether the evaluation of the model is successful. If yes, save the algorithm for future prediction purpose. If not, modify or choose new algorithms, again train, and evaluate the model. Repeat the process until the best model is found. Freeze the best model. Sometimes, we may need to collect more data to improve the performance of deep learning models.

Apply model: Apply the frozen model to newly collected data.

This thesis will follow the above process to design proper deep learning models for speed extraction, fault detection and fault classification for rotating machinery that operated under varying speed conditions, respectively.

3. A deep bi-directional long short-term memory model for automatic rotating speed extraction from vibration signals

This chapter focuses on rotating speed extraction from vibration signals. It is the research Topic #1 as introduced in Section 1.3. A deep bi-directional long short-term memory model is proposed for speed extraction in this chapter. The extracted speed can be used in Chapter 4 and Chapter 5 to facilitate fault detection and fault classification, respectively. Materials of this chapter have been published in a journal paper [128] and a referred conference paper [129].

3.1 Introduction

Rotating machines like wind turbines often work under varying speed conditions. The rotating speed is not only an important condition monitoring indicator for, but also facilitates the fault diagnosis of these machines [69], [130]. As reviewed in Section 1.2.1, available methods for speed extraction broadly include two types, i.e., signal processing based methods and deep learning-based methods. Comparably, signal processing based methods needs intensive expert knowledge and working load to design and implement case sensitive signal processing algorithms to obtain either the mono-component signal or the clear TFR and the ridge tracking algorithm. As such, researchers are exploring alternative methods which are free from these drawbacks, such as the deep learning approach. However, the existing deep learning model, i.e., the MO-LSTM model [78], even performs well in the car running speed extraction, but still suffers from the following

deficiencies: (1) the speed information was only learned in the forward-time direction but the backward-time direction was ignored, and (2) the labeled speed was only utilized at a single time point but remaining $(n-1)$ points were not used in a window of length n. Here the labeled speed means the real speed used in the modeling training. In this chapter, the speed measured by speed sensors will be taken as the labeled speed. The length of the window means the time length of the LSTM model. As a result, the speed information mining ability, and thus the speed extraction accuracy of the reported MO-LSTM model, leave room for improvement when applied to extract speed from complex signals, for example, the vibration signals of rotating machinery.

This chapter proposes a new many-to-many-to-one bi-directional LSTM (MMO-BiLSTM) model to overcome the above-mentioned deficiencies. Compared to the reported MO-LSTM model, the proposed model learns speed related information from vibration signals in both forward-time and backward-time directions with the use of BiLSTM and utilizes labeled speed at all time points in a window through an MM manner. We have spotted a few studies which also used BiLSTM but in other application domains. For example, Huang et al. [131] used a BiLSTM for the intensity modulating and direct detection of high-speed passive optical networks but in a MO manner. Laranjeira et al. [132] employed a BiLSTM for indoor scene recognition through an MM manner. Buoy et al. [133] used a BiLSTM for word segmentation through an MM manner. Jiang et al. [134] proposed a new model structure named global-local BiLSTM for disulfide bonding state prediction. In the global-local BiLSTM. the protein chain is first cut into consecutive short segments. Each short segment is inputted to a BiLSTM layer to learn local features. The outputs of these BiLSTMs are inputted to a successive BiLSTM layer to learn global features. This structure is like the LSTM model used in [113] for machinery fault classification. The difference is that in [113] the LSTM not the BiLSTM was utilized. However, in these works, the MM or MO BiLSTM models were

used for classification tasks, not for regression tasks as in our case. Besides, these models only employed a sole MM or MO structure. While in our proposed MMO-BiLSTM model, we first use an MM structure to utilize the labeled speed at all time points, and then use an MO structure to further improve the speed extraction performance and thus can yield stronger information learning ability.

Effectives of the proposed MMO-BiLSTM is validated with three case studies including an internal combustion engine dataset, a rotor system dataset, and a fixed-shaft gearbox dataset. Major contributions of this chapter are as follows.

(1) A new deep learning model is proposed to extract rotating speed from vibration signals.

(2) A two-stage training strategy is developed to train the proposed model.

The rest of this chapter is organized as follows. Section 3.2 presents the proposed MMO-BiLSTM model. In Section 3.3, three case studies including an internal combustion engine dataset, a rotor system dataset and a fixed-shaft gearbox dataset are displayed to verify the effectiveness of the proposed model. Discussions are provided in Section 3.4. Conclusions of this chapter are drawn in Section 3.5.

3.2 Proposed MMO-BiLSTM model

This section introduces the structure and training strategy of the proposed MMO-BiLSTM. It is based on the deep learning model of LSTM, whose fundamentals have been provided in Section 2.2.3.

3.2.1 Model structure

As described in the introduction of this chapter, the reported MO-LSTM learns information only in the forward direction. However, the backward-time direction also contains useful information. In the speed extraction task, either in the forward-time or in the backward-time direction, the extracted speed should change smoothly. Therefore, the speed at the next time point would also "influence" the speed of the current time point by suppressing abrupt jumps. Therefore, learning in both forward and backward-time directions would reveal more speed related information to improve the speed extraction accuracy. As such, we suggest using the BiLSTM for the speed extraction. Besides, the reported MO-LSTM follows a many-to-one (MO) mode which only utilizes the speed of a single time point. To use more speed information, we suggest using the many-to-many (MM) mode. In this way, the speed at all time points in a window are involved in the training process.

Given above considerations, we proposed an MMO-BiLSTM model for speed extraction in this chapter. It consists of two parts, the BiLSTM part and the LSTM part, as shown in Fig. 3.1. The BiLSTM part contains five layers. They are the Layer 0 – Layer 5, i.e., the Input → BiLSTM → BiLSTM → Average → Dense → Dense. This part assures the model to learn speed related information from vibration signals in both forward-time and backward-time directions. The LSTM part contains Layer 6 and Layer 7, namely the LSTM layer and the Output layer. The LSTM part extracts the refined speed from the information learned by the BiLSTM part in the forward-time direction. Therefore, the overall model forms a many-to-many-to-one (MMO) structure.

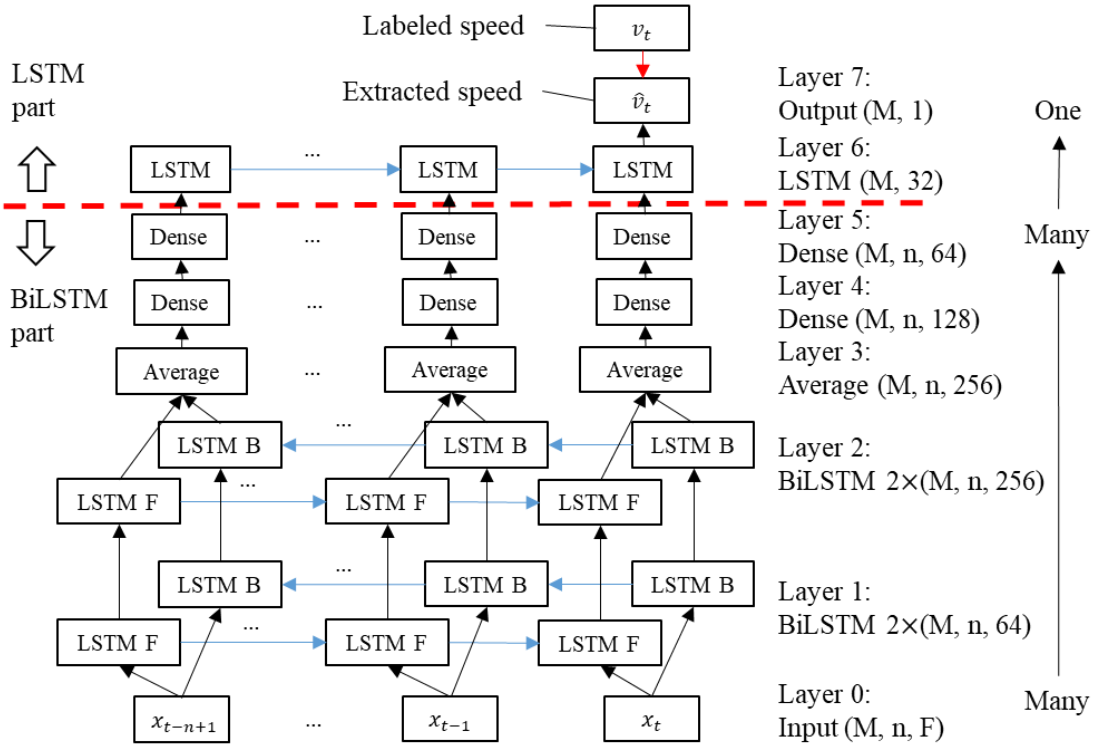


Fig. 3.1: Structure of the proposed MMO-BiLSTM model.

The input of the model are the multi-channel vibration signal sequences $\{x_j\}^s$ $(j = t - n + 1, \dots, t-1, t; s = 1,2, \dots, F)$. Here, F represents the number of channels of vibration signals, and v_t is the measured speed, which is taken as the real or labeled value in this model. The output of the model at time t is the extracted speed $\hat{v}_t$. The output dimension of each layer is also shown in Fig. 3.1. For example, the output dimension of Layer 4 is (M, n, 256), where the first entry (M) is the number of data samples, the second entry (n) is the number of time points within a window, i.e., the window size, and the third entry (256) is the number of the neurons of this layer. The output dimension of Layer 6 is (M, 32), which means the output of this layer is in 2D. For 2D outputs, the first entry also means the number of data samples and the second entry is the number of neurons of this layer.

3.2.2 Training strategy

A two-stage training strategy is proposed to train the proposed MMO-BiLSTM model, as shown in Fig. 3.2. In the first stage, the BiLSTM part is pre-trained via a supervised learning manner. The supervised pre-training is different from the traditional unsupervised pre-training [135], [121], [136], [137], which intends to obtain dimension-reduced features. Instead, the proposed supervised training forces the model to learn speed related information with an unchanged dimension. Besides, the pre-training process is an MM learning process, which means the labeled speed at all time points in a window will be utilized.

In the second stage, the weights and biases of the BiLSTM part will firstly be initialized with the corresponding weights and biases that learned from the first stage. Then the BiLSTM part will be fine-tuned, and the LSTM part will be trained simultaneously to obtain the refined speed via a supervised and MO manner. The way that the LSTM part extracts speed from the information learned by the BiLSTM part mimics the process of predicting a variable with its historic data, which is an effective way to conduct prediction and has been successfully used in wind speed prediction [138].

Compared to the reported MO-LSTM model, where the MO mode and unidirectional LSTM are used, the BiLSTM, MMO and a two-stage training strategy are employed in the proposed MMO-BiLSTM model. The MM enables the proposed model to utilize all the speed labels in a window, while the reported MO-LSTM model utilizes the speed label at only a single time point in a window due to MO. The BiLSTM enables the proposed MMO-BiLSTM model to learn speed related information in both forward-time and backward-time directions, while the reported MO-

LSTM model learns information only in the forward-time direction as it uses LSTM. Therefore, it is promising that our proposed model will have stronger speed information mining ability.

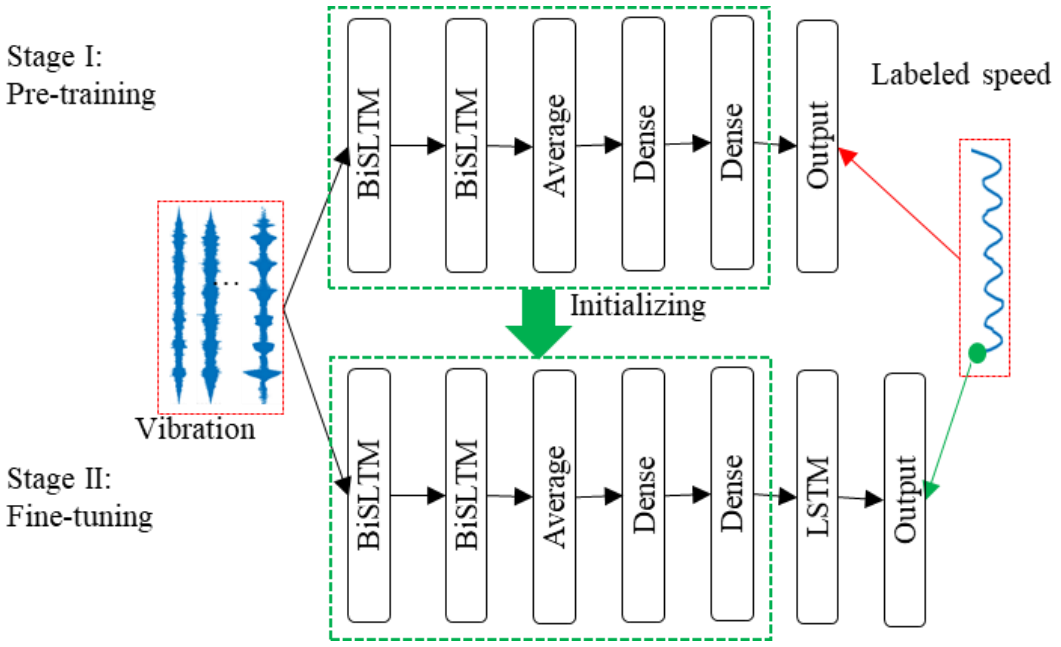


Fig. 3.2: Two-stage training strategy for the proposed model.

The mean absolute percentage error (MAPE) will be used to evaluate the speed extraction accuracy of the proposed model. The definition of the MAPE is as follows,

$$MAPE = \frac{1}{T}\sum_{i}^{T}\left|\frac{v^i - \hat{v}^i}{v^i}\right| \tag{3.1}$$

where $\hat{v}$ denotes the extracted speed by the model, v denotes the labeled speed, i.e., the measured shaft rotating speed and T is the number of the time points of the speed. The CPU time per epoch for model training will be employed as another performance metric. It measures the training speed of a model.

3.3 Case studies

This section presents three case studies to extract the rotating speed of three commonly used rotating machines separately. They are an internal combustion engine, a rotor system, and a fixed-shaft gearbox.

3.3.1 Case study 1: Internal combustion engine dataset

3.3.1.1 Dataset description

The engine dataset was collected from a 4-cylinder internal combustion engine by the author and colleagues in 2016 at Chongqing University, Chongqing, China. The schematic of the experiment setup is shown in Fig. 3.3. Five acoustic sensors named S1-S5 were installed 1 m away from the surfaces of the engine to collect acoustic pressure signals. The acoustic pressure signals have the same nature as vibration thus will be taken as vibration signals in this chapter. The speed of the output shaft of the engine was measured with a speed sensor named T1 which was an encoder. The sampling frequency was 51200 Hz.

The rotating speed of the engine was controlled by a dynamometer via a manual speed control knob. The designed speed profiles include constant (between 700 rpm and 2400 rpm with a step size of 100 rpm), running up (700 rpm to 2400 rpm) and running down (2400 rpm to 700 rpm). For each speed profile, two tests were conducted. Each test lasted for 10 s for the constant speed cases, and about 80 s for the varying speed cases. Note that the measured speed is slightly different from the designed speed profile due to randomness of operating environment and machine capacity. The measured speed of two tests of a single speed profile is also different as the speed was controlled manually.

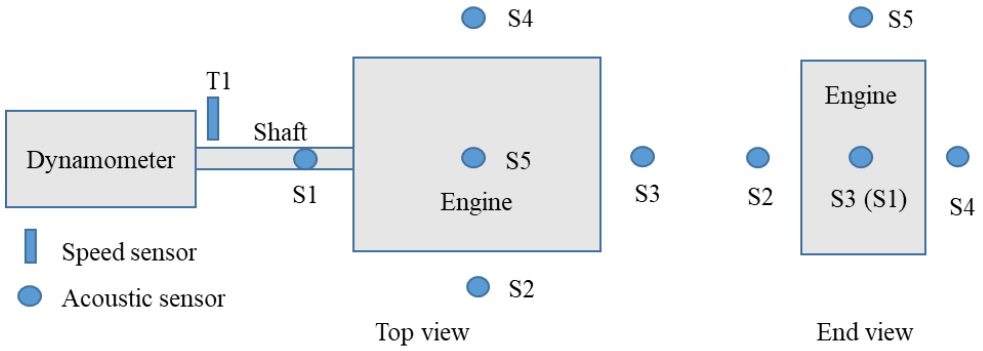


Fig. 3.3: Schematic of the internal combustion engine experiment rig.

The data is preprocessed to make them proper for an LSTM model before the implementation of speed extraction. The vibration signals and the speed signals are preprocessed synchronously. Steps to preprocess the data are listed in the following.

Step 1: Low-pass filter and down-sample data to a new sampling rate of 256 Hz. The new sampling rate assures that up to third order harmonics of the highest speed are contained in the down-sampled signals.

Step 2: Segment the raw data into segments with a length of 2 s, or equivalently, 512 data points. For the engine dataset, we get 300 segments.

Step 3: Randomly split down-sampled segments into a training set and a test set by a ratio of 8:2.

Step 4: Shift each segment with a window size of n and overlapping of $(n-1)$. We get (513-n) shifted samples from each segment. As a result, 300 segments × (513-n) samples/segment = 300 × (513-n) samples are obtained in total. Generally, the longer the window size is, the higher the accuracy the LSTM model will achieve [139]. But a larger the window size needs substantially more CPU time to train the model. In this chapter, the window size is set as $n = 80$. Then 129900

samples are obtained for the engine dataset. The shifted vibration samples will be the input to the LSTM models, and the shifted speed samples will serve as the labeled speed.

The training set will be used to train the models to obtain the optimized weights and biases. When the model is trained, the test set will be applied to evaluate the performance of the model. To further check the performance of the model, the trained model will be used to extract speed from newly obtained vibration data. This process mimics the real applications that using the developed model for speed extraction from newly collected vibration data as a machine keeps operating. For the engine dataset, we have two new applications, i.e., the application 1 and application 2. Each application corresponds to a varying speed profile, either running up or running down. The application data will be preprocessed with Step 1 and Step 4.

3.3.1.2 Model setting

The proposed model was programmed with Python in the framework of Tensorflow, and trained in Google Colaboratory [140] with one GPU used. The Adam [141] is taken as the optimization method. Usually, hyperparameters of deep learning models are determined manually or by parameter searching methods such as the grid search, random search, and model based hyperparameter optimization [26]. In this chapter, the hyperparameters are determined manually after many trials. For example, for the learning rate α, we tried four values, i.e., 0.0001, 0.001, 0.01 and 0.1. The one with the lowest training error and relatively fast convergent speed is chosen. As a result, we get the following hyperparameters: the learning rate is $\alpha = 0.001$; the exponential decay rate for the first moment estimates is $\beta_1 = 0.9$; the exponential decay rate for the second moment estimates is $\beta_2 = 0.999$; and the batch size is 512. The maximum epoch for pre-training is 300 and for fine-tuning is 100. The specific epochs are determined by an early stopping rule.

Early stopping is a commonly used strategy to avoid overfitting in the training [26]. It stops the training process when the validation error begins to rise with some patience such as after certain epochs. The patience is set as 20 epochs in this chapter. We have implemented the reported MO-LSTM model as a benchmark under the same computing environment described in this paragraph. The hyperparameter values of the reported MO-LSTM model are selected in the same way described in this paragraph. After selection, same hyperparameters listed above are used for the MO-LSTM except that the maximum epoch for training is set to be 400 as the reported MO-LSTM uses only one-stage training.

3.3.1.3 Results

For the engine, we intend to extract the engine rotating speed from the vibration signals measured by five acoustic sensors. The speed extraction performance of the reported MO-LSTM model and the proposed MMO-BiLSTM model is shown in Table 3.1. The speed extraction error, namely, the MAPE of the proposed model is substantially decreased compared to the reported model in the test set (from 3.15% to 1.03%), application 1 (from 4.38% to 1.27%) and application 2 (from 5.68% to 1.29%). The relative reduction percentages are 67.30%, 71.01% and 77.29%, respectively. The test set measures the performance of the model with existing data, while the two applications measure the performance when applying the model to newly collected data. The way to use both the test set and applications mimics the process of how we employ deep learning models in real applications, that is, train a model and test it with historical data and apply it to new data.

The extracted speed of the two applications is shown in Fig. 3.4. Each application represents a typical speed profile of the engine. The application 1 corresponds to the running down and the application 2 corresponds to the running up speed profile. In Fig. 3.4, some parts of the speed are

zoomed in for a closer observation. The zoomed in parts are named (I) for high-speed operation and (II) for low-speed operation. The extracted speed curves by the proposed MMO-BiLSTM model are closer to the real speed than the reported MO-LSTM model, which means the proposed MMO-BiLSTM model has stronger speed mining ability than the reported MO-LSTM model.

The Fig. 3.4 also illustrates that the accuracy of the extracted speed in low-speed (less than 1000 rpm) operation is lower than that of the high-speed operation, especially for the reported MO-LSTM model. This may be because that higher speed brings higher energy to the machine, thus more speed-related energy will be contained in the machine's vibration. Given the ambient environment (i.e., fixed environment noise), the SNR of the vibration signal would then increase with the increase of speed. For a deep learning model, it is easier to learn from cleaner (higher SNR) data. As a result, the speed extraction accuracy of both the reported MO-LSTM model and the proposed MMO-BiLSTM model for low-speed operation is lower than that for high-speed operation. However, for the low-speed operation (see zoomed in parts (II) in Fig. 3.4), the proposed MMO-BiLSTM model can still track the real speed relatively well, but the reported MO-LSTM model cannot. This means the proposed model can work well even with low SNR vibration signals.

Table 3.1: Speed extraction results of the engine dataset.

Model		Reported MO-LSTM	Proposed MMO-BiLSTM	Relative reduction
CPU time (s)		8	24/21*	-
MAPE	Test set	3.15%	1.03%	67.30%
	Application 1	4.38%	1.27%	71.01%
	Application 2	5.68%	1.29%	77.29%

* Means the CPU time of the pre-training and fine-tuning

From Fig. 3.4, we can observe slight fluctuations in the extracted speed curves either by the reported MO-LSTM or the proposed MMO-BiLSTM compared to the real speed curves. This may

be caused by noise in the vibration signals. Noise induces fluctuations to the input (vibration) of the models. Consequently, fluctuations occur in the output (extracted speed) of the models. The fluctuations can be eased by some curve smoothing strategies, like weighted moving average [142].

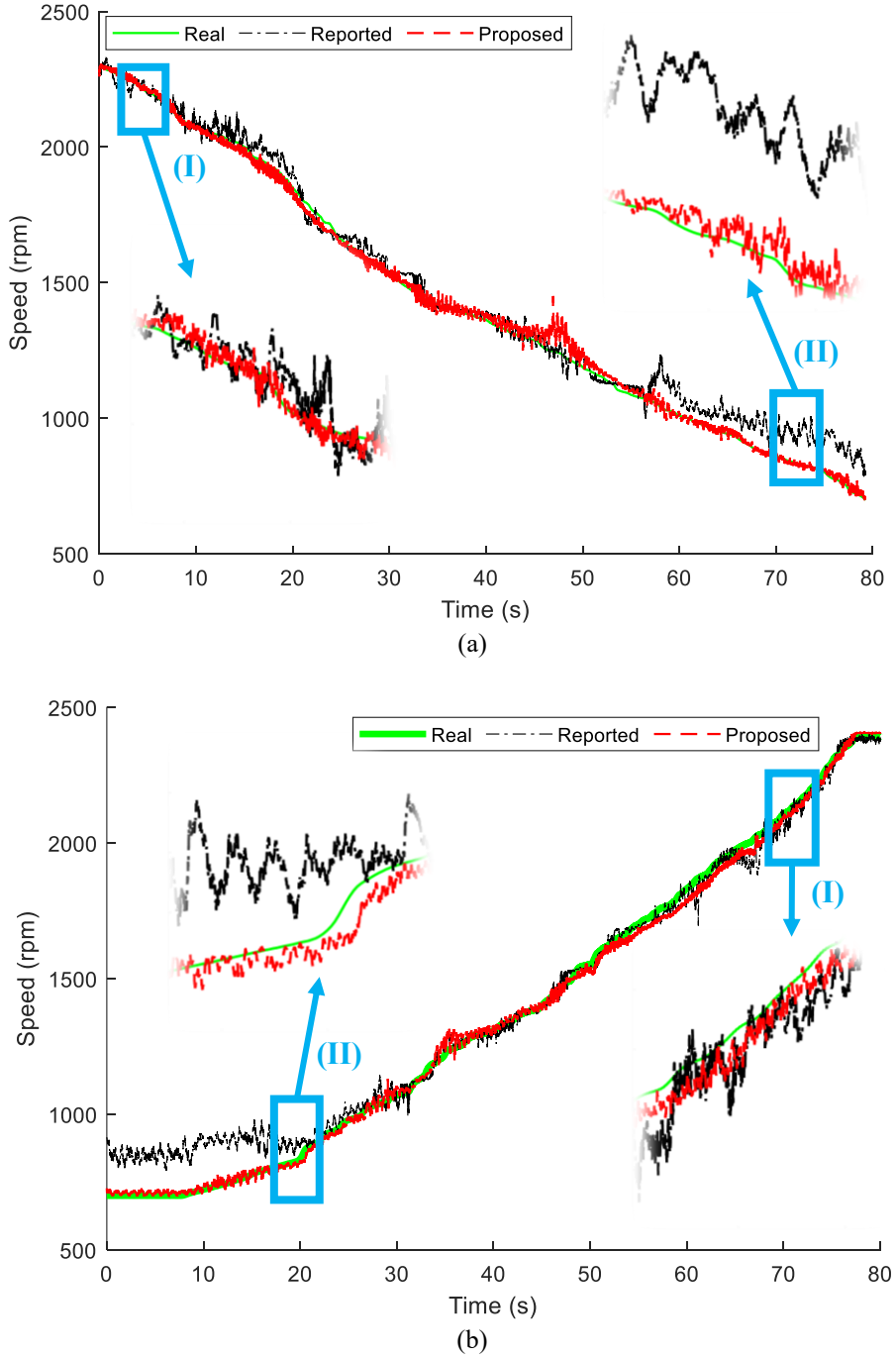


Fig. 3.4: Speed extraction results of the engine dataset: (a) Application 1 and (b) Application 2.

Table 1 also shows that the proposed MMO-BiLSTM model needs more time to train per epoch than the reported MO-LSTM model. For the reported MO-LSTM model, the CPU time per epoch is 8 s, while the proposed model needs 24 s (pre-training) or 21 s (fine-tuning) per epoch. The reason is that the proposed model has more trainable parameters (734,465 parameters) than that of the reported model (394,945 parameters).

3.3.2 Case study 2: Rotor system dataset

3.3.2.1 Data description

The rotor system dataset was collected by author's colleagues in 2017 using a rotor system simulator from the University of Electronic Science and Technology of China, Chengdu, China [143]. The experimental setup of the rotor system and the schematic of sensor locations are shown in Fig. 3.5. Two accelerometers were installed on the bearing housing to the right side to measure the vertical and horizontal accelerations. Two displacement transducers were placed between the flange and weight plate to measure the displacement in vertical and horizontal directions. One tachometer was installed near the motor to measure the rotating speed of the shaft. The sampling frequency was 10240 Hz.

The rotating speed of the rotor was controlled by an automatic control system. Two speed profiles ranging between 300 rpm and 2000 rpm were designed. Both speed profiles contain linear up, sinusoidal and linear down parts, but the amplitudes of the sinusoidal parts are different. For each speed profile, 40 repeating tests were conducted. Each test lasted for about 14 s. The rotor system dataset is preprocessed in the same way as the engine dataset as in Case 1. After preprocessing, we get 242480 samples, and two more applications, i.e., application 1 and application 2.

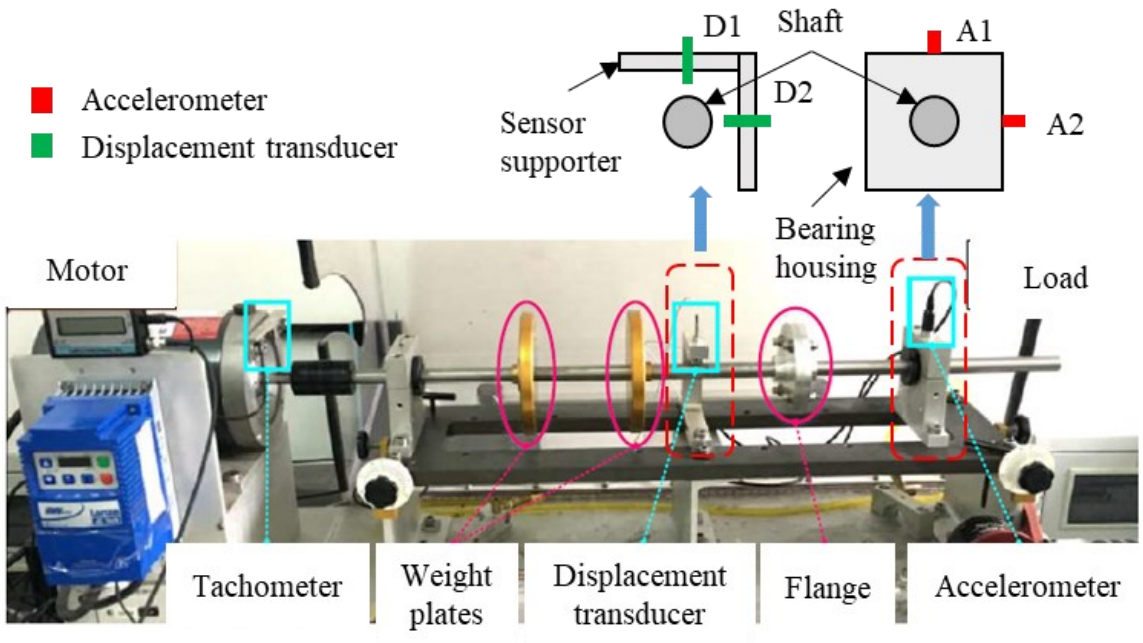


Fig. 3.5: Experimental setup of the rotor system.

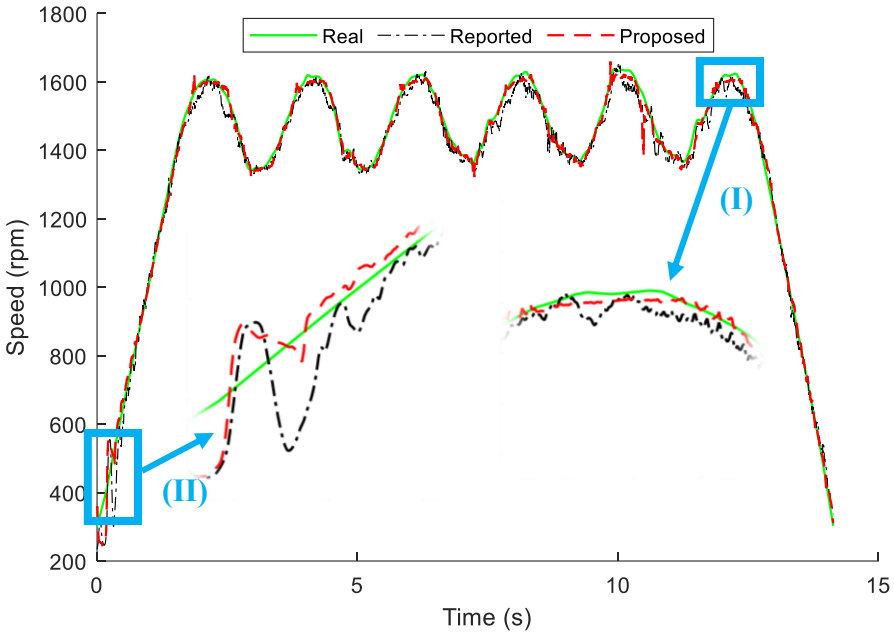
3.3.2.2 Results

For the rotor system, we intend to extract the rotating speed of the rotor shaft from four-channel vibration signals including two channels of accelerations and two channels of displacements. The model parameters are the same as the engine case. The speed extraction results of the proposed model and the reported model are shown in Table 3.2. The extracted speed of the two applications is shown in Fig. 3.6. Detailed observations are given below.

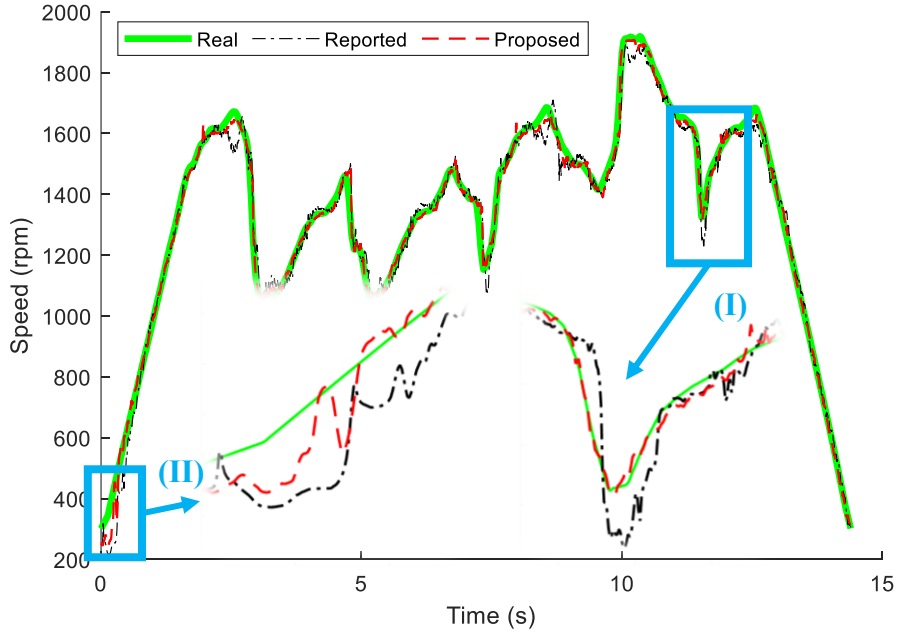
Table 3.2: Speed extraction results of the rotor system dataset.

Model		Reported MO-LSTM	Proposed MMO-BiLSTM	Relative reduction
CPU time (s)		28	53/61*	-
MAPE	Test set	1.50%	0.69%	54.00%
	Application 1	2.21%	1.27%	42.53%
	Application 2	2.97%	1.50%	49.50%

* Means the CPU time of the pre-training and fine-tuning



(a)



(b)

Fig. 3.6: Speed extraction results of the rotor system dataset: (a) Application 1 and (b) Application 2.

The MAPEs of the proposed MMO-BiLSTM model are 0.69%, 1.27% and 1.50% for the test set, application 1 and application 2, respectively. These values are all smaller than their counterparts

of the reported MO-LSTM model, i.e., 1.50%, 2.21% and 2.97%, respectively. Compared to the reported MO-LSTM model, the speed extraction errors of the proposed MMO-BiLSTM model are reduced by 54.00%, 42.53% and 49.50%, respectively. But the proposed model costumes about double time per epoch of the reported MO-LSTM model to train.

For a closer observation of Fig. 3.6, some parts of it are zoomed in. See those named (I) and (II). The two applications represent two typical speed profiles of the rotor system. Both the reported model and the proposed model can track the speed trend well, but the proposed model tracks the real speed closer than the reported model. From Fig. 3.6, we can also observe that the speed extraction accuracy of both the reported MO-LSTM and the proposed MMO-BiLSTM for the low-speed (less than 400 rpm) operation is lower than that for the high-speed operation. But the proposed MMO-BiLSTM model tracks the real speed better in low-speed operation than the reported MO-LSTM model (see zoomed in parts (II) in Fig. 3.6). These observations are like the engine dataset as seen in Case 1.

3.3.3 Case study 3: Fixed-shaft gearbox dataset

3.3.3.1 Data description

The fixed-shaft gearbox dataset [144] was collected at the University of Alberta, Edmonton, Alberta, Canada in 2018 by the author and author's colleagues. The test rig is shown in Fig. 3.7(a). It consists of a drive motor, a bevel gearbox, a 1st stage planetary gearbox, a 2nd stage planetary gearbox, a 1st stage speed-up fixed-shat gearbox, a 2nd stage speed-up fixed-shaft gearbox, and a driven motor. The rotating speed of the gearbox was controlled by a variable frequency drive. The gearbox of interest is the 2nd speed-up gearbox (marked in a rectangle). The fixed-shaft gearbox has three shafts, i.e., an input shaft, a middle shaft, and an output shaft, as shown in Fig. 3.7(b).

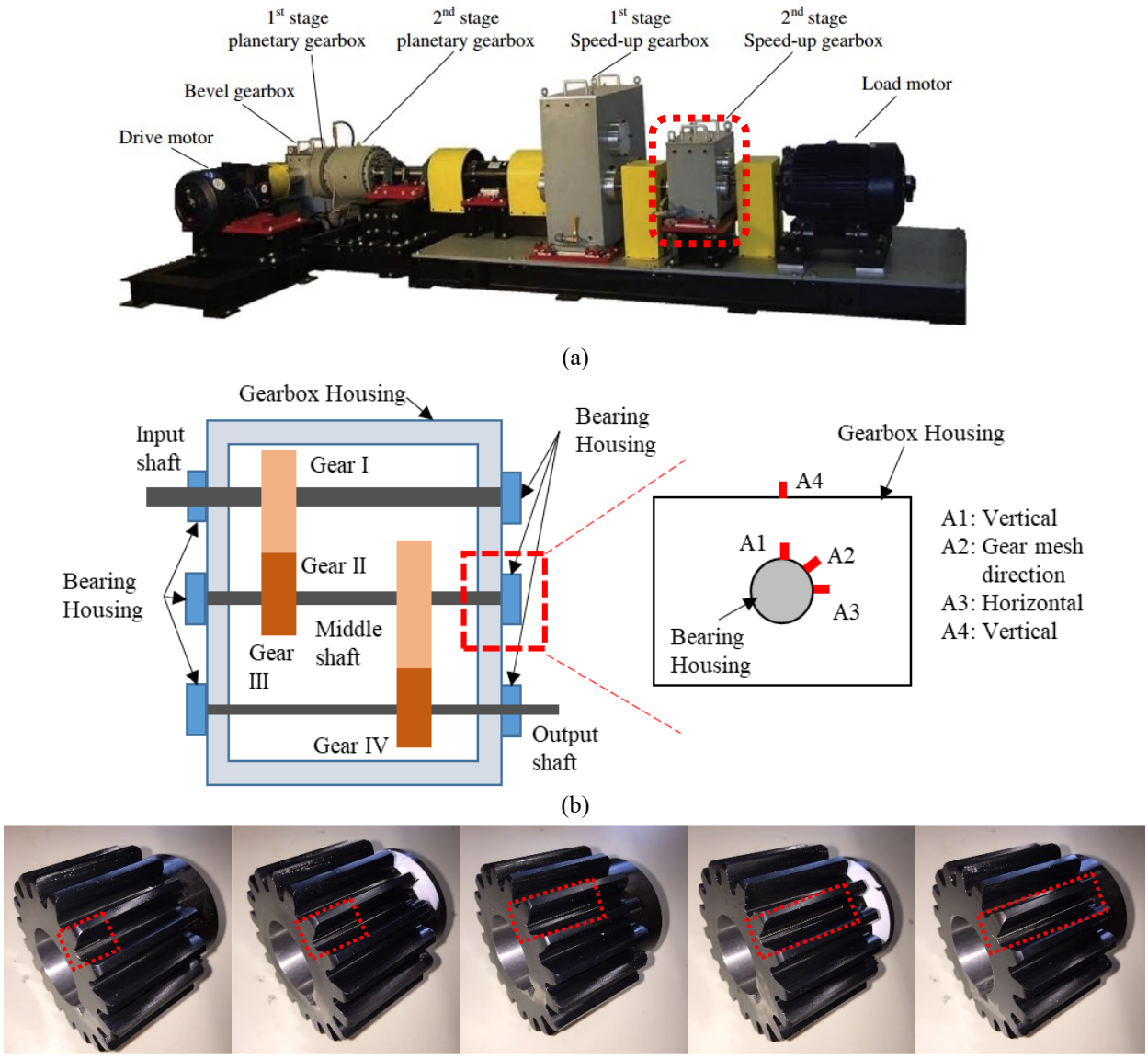


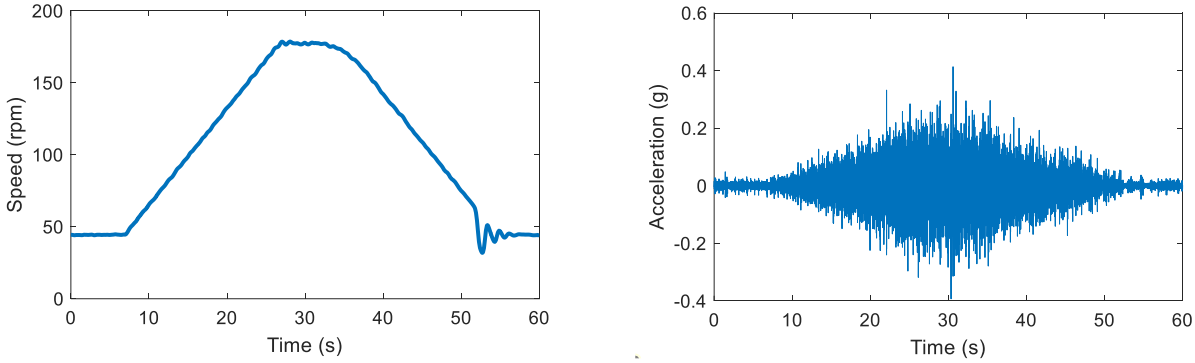
Fig. 3.7: Experimental setup for the fixed-shaft gearbox dataset: (a) Test rig, (b) Schematic of the fixed-shaft gearbox and sensor locations and (c) Simulated faulty gears with crack severities increasing from left to right.

Four accelerometers were installed to collect the vibration data. Three of them were mounted on the bearing housing of the middle shaft in the horizontal, vertical and gear meshing directions, and one was mounted on the gearbox top cover. An encoder was installed on the shaft of the load motor to pick up its rotating speed. The output shaft of the gearbox was connected to the load motor shaft

via a coupler. The speed of the output shaft thus equaled to the motor speed. The speed of the middle shaft was then calculated according to the transmission ratio between the middle shaft and the output shaft.

Data with different health states under different speed conditions was collected. The health states include the healthy state and the faulty states with five severity levels of tooth root cracks, as shown in Fig. 3.7(c). The crack was manually induced. The speed varies continuously, i.e., running up to 180 rpm and then going down. For each case, 5 repeating tests were conducted. Each test lasted about 60 s. The sampling frequency was 25600 Hz. An example of the collected vibration signal and its corresponding speed signal is given in Fig. 3.8.

Only healthy data is used in this chapter. Faulty data is going to be used in Chapter 4 and Chapter 5. All the four channels of vibration will be used for speed extraction. The gearbox dataset is preprocessed in the same way as the engine dataset and the rotor system dataset. After preprocessing, 255037 samples are obtained. Also, two applications, i.e., application 1 and application 2, will be applied.



(a) Speed (b) Acceleration

Fig. 3.8: Example of collected data of the fixed-shaft gearbox: (a) Speed and (b) Acceleration.

3.3.3.2 Results

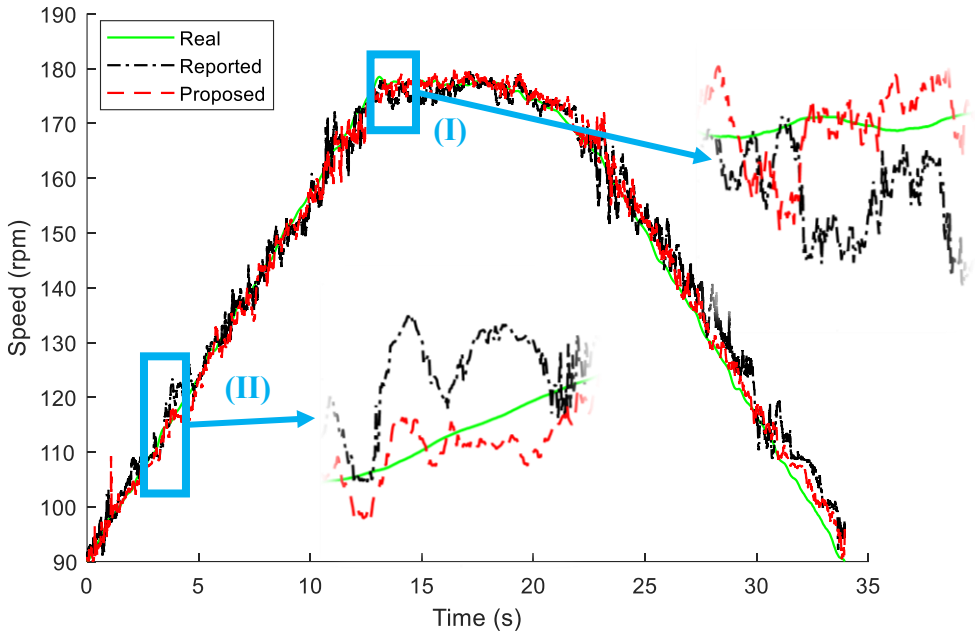
For the fixed-shaft gearbox dataset, the goal is to extract the rotating speed of the middle shaft from four-channel vibration signals. Table 3.3 displays the speed extraction results of the reported model and the proposed model with the gearbox dataset. The MAPEs of the proposed MMO-BiLSTM model are 2.03%, 2.05% and 1.97% with the test set, application 1 and application 2, respectively. These values yield corresponding reductions of 25.61%, 37.56% and 29.44 % over the speed extraction errors of the reported MO-LSTM model. Still, the proposed model consumes about double time per epoch to train.

Table 3.3: Speed extraction results of the fixed-shaft gearbox dataset.

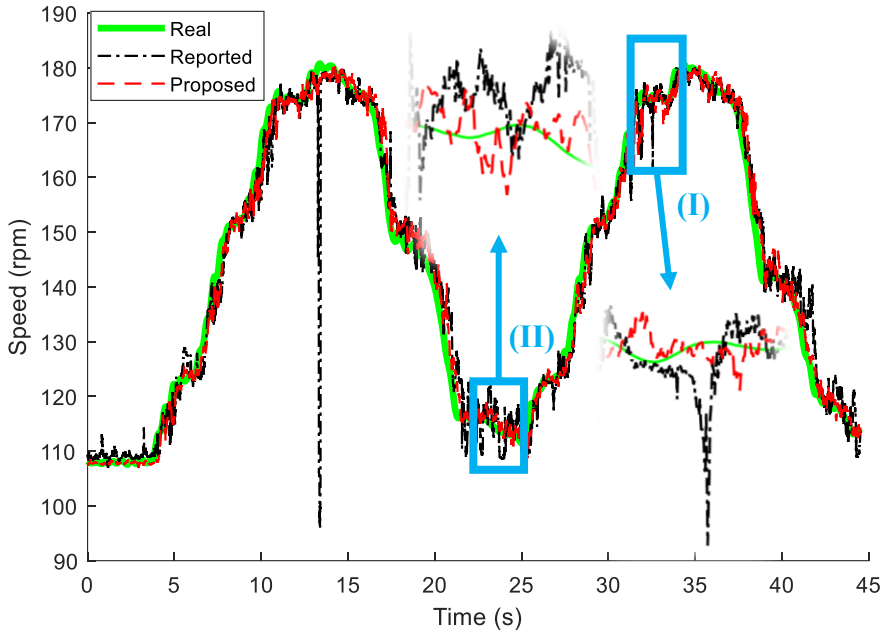
Model		Reported MO-LSTM	Proposed MMO-BiLSTM	Relative reduction
CPU time (s)		30	53/60*	-
MAPE	Test set	2.03%	1.51%	25.61%
	Application 1	2.05%	1.28%	37.56%
	Application 2	1.97%	1.39%	29.44%

* Means the CPU time of the pre-training and fine-tuning

Different from the engine and the rotor system, the fixed-shaft gearbox operates in a relatively narrow speed range of 100 rpm – 180 rpm, but with heavy load applied. As a result, some unexpected shocks would happen in the gear meshing process. Fig. 3.9 illustrated the extracted speed curves of the two applications. Some parts of this figure are zoomed in which are named (I) and (II) for a closer observation. It can be observed especially from Fig. 3.9 (b) that some unexpected jumps occur in the extracted speed by the reported model, but smoother speed that closes to the real speed curves are tracked by the proposed model. This means our proposed MMO-BiLSTM model outperforms the reported MO-LSTM model with the fixed-shaft gearbox dataset.



(a)



(b)

Fig. 3.9: Speed extraction results of the fixed-shaft gearbox dataset: (a) Application 1; and (b) Application 2.

Also different from the engine dataset and the rotor system dataset, the speed extraction accuracy of the gearbox dataset is similar in the low-speed operation and in the high-speed operation. The

reason may be that the gearbox operates in a relatively narrow speed range (max speed/min speed = 1.8), while the engine and the rotor system operate in a quite wide speed range (max speed/min speed = 3.4 and 6.7, respectively). For the gearbox, the SNR difference caused by speed variation in such a narrow range may be not large enough to influence the speed extraction accuracy.

Overall, the speed extraction results of the proposed MMO-BiLSTM model with the three typical rotating machines imply that, the proposed model can successfully extract rotating speed from vibration signals of different rotating machines and outperforms the reported MO-LSTM model. These machines can operate under different speed and load conditions. Therefore, we believe our proposed deep learning model is effective to extract rotating speed from vibration signals and can be applied to other rotating machines.

3.4 Discussion

In the proposed and the reported models, several model structures, namely, LSTM, BiLSTM, MM, MO and MMO, and a two-stage training strategy (TS) are used. To understand why the proposed MMO-BiLSTM model performs better than the reported MO-LSTM model, separate contributions of these model structures and the training strategy to the speed extraction accuracy are evaluated with the above three datasets. Models formed by feasible combinations of these structures and the training strategy are listed in Table 3.4, wherein the LSTM+MO is exactly the reported MO-LSTM model and the BiLSTM+MM+MO+TS is the proposed MMO-BiLSTM model. Note that in Table 3.4, LSTM-M (BiLSTM-M) and LSTM-O (BiLSTM-O) mean the LSTM (BiLSTM) model pertaining to a MM structure and a MO structure, respectively. The second layer of LSTM+MO

and LSTM+MM+MO is a Dense layer while others are not. As according to our trials, with this Dense layer kept the LSTM+MO and the LSTM+MM+MO perform better for speed extraction.

Table 3.4: Feasible models composed of different structures.

Model	Layers
LSTM + MO (Reported MO-LSTM)	Input → Dense → LSTM-M → LSTM-O → Dense → Dense → Output
LSTM + MM	Input → LSTM-M → LSTM-M → Dense → Dense → Output
BiLSTM + MO	Input → BiLSTM-M → BiLSTM-O → Dense → Dense → Output
BiLSTM + MM	Input → BiLSTM-M → BiLSTM-M → Average → Dense → Dense → Output
LSTM + MM + MO	Input → Dense → LSTM-M → LSTM-M → Dense → Dense → LSTM-O → Output
BiLSTM + MM + MO	Input → BiLSTM-M → BiLSTM-M → Average → Dense → Dense → LSTM-O → Output
LSTM + MM + MO + TS	Pre-training: Input → LSTM-M → LSTM-M → Dense → Dense → Output Fine-tuning: Input → LSTM-M → LSTM-M → Dense → Dense → LSTM-O → Output
BiLSTM + MM + MO + TS (Proposed MMO-BiLSTM)	Pre-training: Input → BiLSTM-M → BiLSTM-M → Average → Dense → Dense → Output Fine-tuning: Input → BiLSTM-M → BiLSTM-M → Average → Dense → Dense → LSTM-O → Output

The speed extraction performance of these models with the engine dataset are shown in Table 3.5. The results with the rest two datasets, namely, the rotor system dataset and the fixed-shaft gearbox dataset illustrate the same trend as the engine dataset thus not shown. Table 5 shows that,

(1) All the models achieve higher speed extraction accuracy than the reported MO-LSTM (LSTM+MO) model.

(2) When the MM is employed, the accuracy is substantially improved than MO no matter the LSTM or BiLSTM is used. The MM consumes similar CPU time as the MO per epoch.

(3) The BiLSTM achieves higher accuracy than LSTM with either MM or MO used, but the BiLSTM consumes more CPU time than LSTM per epoch.

(4) Without TS, the MM+MO (i.e., MMO) structure performs relatively poor with either LSTM or BiLSTM used. When the TS is employed, the speed extraction accuracy is substantially improved, especially for the BiLSTM case.

(5) Overall, the BiLSTM + MM + MO + TS, i.e., the proposed MMO-BiLSTM model performs the best in terms of the speed extraction accuracy.

Given the comparison results of these structures, we would like to suggest that if one has sufficient computation resources and prioritizes the speed extraction accuracy, the proposed MMO-BiLSTM model is preferred. If we want a trade-off between the accuracy and the training time consumption, only using the MM is preferred. Only Using BiLSTM is not suggested as it consumes much more training time than the MM but does not bring significant gain in the accuracy compared to the MM.

Table 3.5: Speed extraction performances of different models with the engine dataset.

Model	CPU time (s)	MAPE (%)		
		Test set	Application 1	Application 2
LSTM + MO (Reported MO-LSTM)	8	3.44	4.33	6.65
LSTM + MM	9	1.70	1.27	2.12
BiLSTM + MO	23	1.99	1.99	3.40
BiLSTM + MM	24	1.05	1.22	1.92
LSTM + MM + MO	12.5	2.71	3.51	2.75
BiLSTM + MM + MO	21	2.28	2.93	3.55
LSTM + MM + MO + TS	9/12.5*	2.80	1.73	2.01
BiLSTM + MM + MO + TS (Proposed MMO-BiLSTM)	24/21*	1.03	1.28	1.29

* Means the CPU time of the pre-training and fine-tuning

3.5 Summary and conclusion

This chapter presents a deep learning model, i.e., the MMO-BiLSTM, to automatically extract a machine's rotating speed from its vibration signals. The performance of the proposed MMO-BiLSTM is evaluated with three typical rotating machines, including an internal combustion engine, a rotor system, and a fixed-shaft gearbox, and compared with the reported MO-LSTM model. From the results, we can conclude that,

(1) The proposed MMO-BiLSTM can successfully extract the rotating speed of rotating machines directly from their vibration signals and achieves higher accuracy than the reported MO-LSTM model. Therefore, it is promising that the proposed model could work like a virtual speed sensor to automatically "measure" the speed from vibration signals.

(2) For the speed extraction task, the MM structure outperforms the MO structure, and the BiLSTM performs better than the LSTM. The MMO with BiLSTM performs the best when the proposed two-stage training strategy is taken.

In this chapter, the proposed model is trained with the historical vibration and speed data of a machine and can only extract the speed of the same machine. In the future, we will investigate to train the proposed model with historical data of one machine (e.g., motor) but apply it to another similar machine (e.g., rotor system) by using techniques like transfer learning [115], [145]. This could be very useful for those machines that speed information is in demand, but the speed sensors are difficult to be installed due to physical space and/or cost limit.

4. A speed normalized autoencoder for rotating machinery fault detection under varying speed conditions

This chapter focuses on the fault detection of rotating machinery that operated under varying speed conditions. It is the research Topic #2 as defined in Section 1.3. A speed normalized autoencoder is proposed for fault detection of rotating machinery under varying speed condition. The proposed model requires the speed as an auxiliary input in addition to the vibration. The speed utilized in this chapter can be measured or extracted from Chapter 3. When a fault is detected with a machine, the type and the severity of the fault is going to be identified in Chapter 5. Materials of this chapter have been published in a journal paper [146].

4.1 Introduction

Rotating machinery in service usually deteriorates over time due to variable internal or external excitations. When the deterioration is accumulated to a certain level, we say a fault is occurred. Faults if not detected and addressed early may lead to the failure of a machine. Failure would result in unexpected shutdown which further results in economical and/or moral loss. Fault detection has, thus, been introduced as an effective way to prevent the occurrence of failure [3].

Fault detection is usually understood as an unsupervised learning problem. Only healthy data is allowed for the development of fault detection algorithms. A typical unsupervised deep learning model, the autoencoder (AE) and its variants were widely used. For example, Zhao et al. [98]

employed a deep AE to detect wind turbine faults. Reddy et al. [82] utilized a deep AE to conduct anomaly detection and fault disambiguation of flight data. Luo et al. [97] trained a sparse AE for the fault detection of a machine tool. Malhotra et al. [85] and Chen et al. [147] used an AE-shaped long short-term memory (AE-LSTM) model for anomaly detection of sensors and wind turbines, respectively. Yu et al. [148] and Jana et al. [149] utilized a convolutional autoencoder (AE-CNN) for fault detection of industrial processes and sensors in structural health monitoring, respectively. Spyridon et al. [150] suggested an AE based generative adversarial network for fault detection of a chemical industrial system.

While utilizing AEs for fault detection, we often follow a reconstruction procedure [151], as shown in Fig. 4.1. Firstly, healthy data is used to train an AE to reconstruct its inputs. The AE can be trained through the widely used backpropagation algorithm [26], or probabilistic learning [152], [153], or the generative-adversarial manner [150], [154]. The well-trained AE is then employed to reconstruct newly collected data with unknown health states. The difference between the reconstructed input and the input is defined as the reconstruction error, which is also known as the residual. The residual is then inputted to a successive detection module which will identify whether the newly collected data indicates a fault or not. The detection module can be a simple health indicator calculated from the residual, such as the harmonic to noise ratio [32] or the widely used root mean square (RMS) [82], [83].

Faut detection algorithms must adapt to dynamic environments [155]. This is critical for rotating machines since they usually work under varying speed conditions. Speed variation introduces amplitude modulation (AM) and frequency modulation (FM) to vibration signals [51]. The AM and FM effects can also be introduced by faults [52], [92]. Their effects are often overlapped [8].

The overlapped effects if not distinguished, may lead to false alarms and/or missed alerts in fault detection [92]. The above-mentioned AEs for fault detection did not consider the effects of speed variation. Their performances cannot be guaranteed for varying speed conditions.

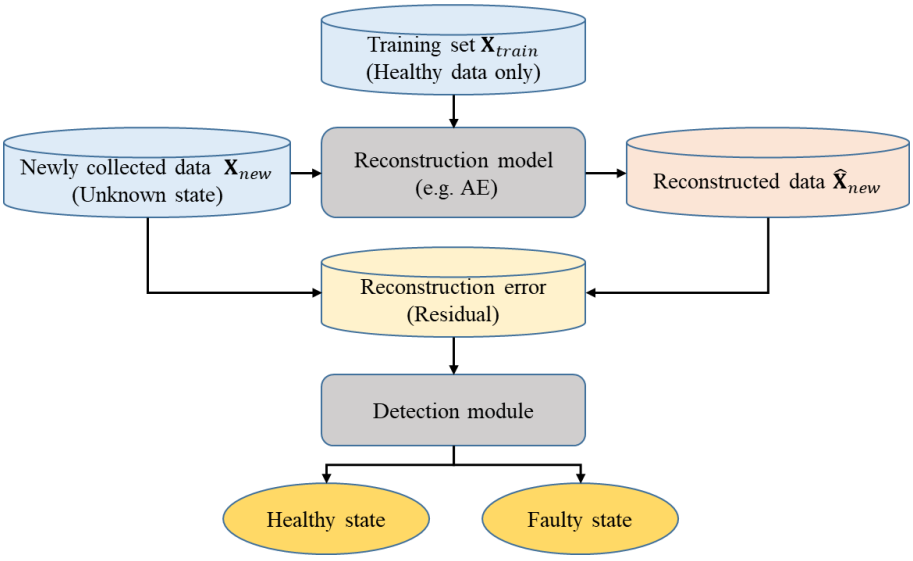


Fig. 4.1: Procedure of fault detection based on data reconstruction [151], [152].

The focus of this chapter is to improve the fault detection performance of AEs for rotating machinery that operated under varying speed conditions. The key is to address the effects induced by speed variation. Once the effects of speed variation are removed, the resultant false alarms and/or missing alerts could be avoided. The fault detection performance would be improved accordingly. As reviewed in Section 1.2.2, exiting models [95], [96] for fault detection under varying speed conditions did not indicate how the effects of speed variation were addressed. Moreover, the employed data were the value-type data in [95], [96], rather than the waveform data as the vibration in our case. The waveform data is much more complicated than the value-type data [3]. Effects of speed variation on value-type data and waveform data is also different.

Therefore, the fault detection performance of the models employed in [95], [96] over the vibration data needs further investigation.

The goal of this chapter is to design an AE model that can remove the effects of speed variation automatically from vibration data, and ultimately improve its fault detection performance. To achieve this goal, we propose a new deep learning model named the speed normalization autoencoder (SN-AE). The idea is to design an auxiliary branch for the AE to remove the effects of speed variation automatically. Specifically, an auxiliary branch named speed normalization (SN) is designed to learns a SN function, which further normalizes the vibration data to removes the effects of speed variation. The normalized vibration is then processed by the AE for fault detection. The idea to normalize the vibration using a certain function with respect to speed is indeed inspired by [52], [92]. Through normalization, the effects of AM induced by the speed variation could be removed. Once the effects of AM are removed, the fault detection performance could be improved as the false alarms and/or missing alerts induced by speed variation could be eased. In this chapter, we will focus on the AM only. The FM will be not considered for now. Indeed, based on our evaluation, the FM has limited impact on the fault detection performance. Th details are given in Section 4.5.1.

Indeed, the primary goal of the proposed model is to remove the speed induced variations in data. For the purpose of removing variation in data, we have spotted a few related works in other domains. For example, Ren et al. [156] proposed a variation-normalized AE for person re-identification in photos shotted by street cameras. In [156], the variation-normalized AE removes all variations in photos while sustains person-related common features, so that a same person in different photos could be re-identified. However, this model may fail in our case as we only want

to remove speed induced variations in data but to sustain fault related variation in the meantime. In the domain of medical health, Rong et al. [157] developed a deep adversarial model to remove batch effects in liquid chromatography mass spectrometry-based metabolomics data. So that the metabolism changes induced by disease would be well estimated. The changes were then used to indicate the occurrence of certain diseases. This model is not appreciated in our case as it will also remove fault related variations intra batches, and thus leads to incorrect detection.

In this chapter, the typical deep AE as used in [82], [98] will taken as the baseline. The AE is selected as it is widely adopted for fault detection, and the variants of the AE such as the sparse AE [95], [97], denoising AE [96], AE-LSTM [85], [147] and AE-CNN [148], [149] are among the state-of-the-art methods for fault detection. Without loss of generality, the proposed SN will be developed based on the typical AE, and the whole package will be applied to these advanced variants to show its generalizability and superiority.

Given a fixed AE, how to select the hyperparameters of the SN branch that returns the best speed effects removal performance and then superior fault detection performances remains a problem. The hyperparameters in this chapter refer to that regarding the structure of the SN branch only. We assume the hyperparameters of the baseline AE are already well selected. Existing strategies such as the grid search [26], random search [26], Bayesian optimization [158] and differentiable architecture search [159] select hyperparameters whilst minimize the mean square error of the reconstruction of the validation set (VMSE). The VMSE measures how well a model reconstructs its input. A minimal VMSE may guarantee a good reconstruction performance but cannot assure a good speed effects removal performance. As the scale of the learned SN function can vary across hyperparameters. A smaller VMSE may be because of relatively smaller SN values, not better

speed effects removal performance. To address this problem, we propose a new measure, i.e., the correlation coefficient between the health indicator and the speed of the validation set (VCOR), for the hyperparameter selection for the SN branch. The rationale is that if the effects of the speed variation are removed, the health indicator should be independent from the rotating speed. The VCOR should approach zeros, otherwise would be large.

The effectiveness of the proposed SN-AE as well as the proposed VCOR is validated with three case studies. The case studies correspond three typical rotating machines, i.e., a planetary gearbox, a fixed-shat gearbox and a rolling element bearing, respectively. Results show that the proposed SN significantly improves the fault detection performances of the baseline AE, as well as its variants. Major contributions of this chapter are as follows:

(1) Proposed a new deep learning model SN-AE for rotating machinery fault detection under varying speed conditions.
(2) Proposed a new hyperparameter selection measure VCOR for the SN branch of the proposed SN-AE model.

The rest of this chapter is organized as follows. Section 4.2 introduces the baseline AE. Details of the proposed SN-AE are given in Section 4.3. Case studies are conducted in Section 4.4. Discussions are made in Section 4.5. Conclusions are drawn in Section 4.6.

4.2 Baseline models

This section provides a brief introduction the AE model that is going to be the baseline model in this chapter. More knowledge of the AE has been provided in Section 2.2.1. The deep AE

employed in [82], [98] is selected as the baseline model in this chapter, as it features a typical AE structure, as shown in Fig. 4.2. The baseline AE consists of an input layer, multiple hidden layers, and an output layer. The hidden layer in the middle is named as the bottleneck layer. The AE is used to reconstruct the inputted vibration data. The RMS of the reconstruction residual is taken as the HI for fault detection. As indicated in the Introduction, the fault detection performance of the baseline AE is deteriorated by speed variation. We will address this problem in this chapter.

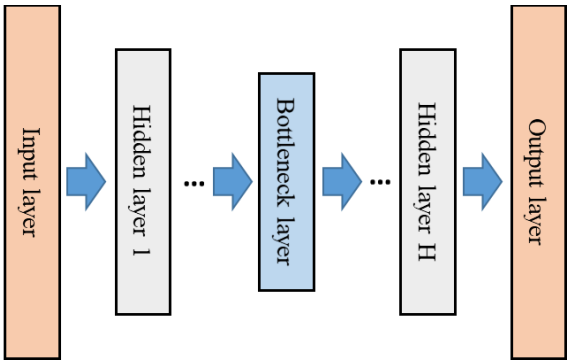


Fig. 4.2: Structure of the baseline deep AE [98].

In this chapter, the area under the receiver operator characteristic curve (AUC) [160] will be used to measure the fault detection performance. The receiver operator characteristic curve is created by plotting the true positive rate and false positive rate at all feasible threshold settings. Suppose a test set contains N faulty (negative) samples and P healthy (positive) samples. At a certain threshold, TP samples out of the P healthy samples are detected as healthy and NP samples out of the N faulty samples are missing alerted. The true positive rate and the false positive rate are found with,

$$\text{True positive rate} = \frac{TP}{P} \tag{4.1}$$

$$\text{False positive rate} = \frac{NP}{N} \tag{4.2}$$

Calculate the true positive rates and the false positive rates at all thresholds. The receiver operator characteristic curve can be plotted with the false positive rate as the x-axis and the true positive rate as the y-axis. The AUC is the area under the receiver operator characteristic curve, which can be found using the numerical integration. The AUC provides a comprehensive measure by integrating false positive rates and true positive rates and is widely used in the context of fault detection [32], [160]. The value of AUC varies from 0 to 1, and the larger the better. An AUC of 1 means that the model performs perfectly. No false alarms or missing alters occur. An AUC of 0.5 corresponds to a coin flip. An AUC of 0 means that all faults are not altered, and all healthy states are falsely alarmed.

4.3 Proposed speed normalized autoencoder (SN-AE)

4.3.1 Structure of SN-AE

As mentioned in the Introduction, existing methods for rotating machinery fault detection under varying speed conditions are either not able to address the effects of speed variation, or address in a manual and separate manner. The goal of this work is to remove effects of speed variation automatically to improve the fault detection performances of AEs. The proposed idea is to learn a speed-related normalization function automatically through the training of AE with a proper design of the model structure. The normalization function is to multiply the vibration to remove or mitigate the AM effects. In this chapter, a deep learning model named speed normalized autoencoder (SN-AE) is proposed to implement the proposed idea.

The structure of the proposed SN-AE is illustrated in Fig. 4.3. It consists of two branches, i.e., an AE branch, and a speed normalization (SN) branch. The AE over here is identical to the baseline AE. It is going to reconstruct the data inputted to it. The SN branch is an auxiliary neural network that is going to learn a speed normalization function $g(\boldsymbol{s}, \boldsymbol{\theta}_s)$. It takes the speed signal $\boldsymbol{s}$ as the input. The $\boldsymbol{\theta}_s$ refers to trainable parameters of the SN branch including the weights and biases. The $g(\boldsymbol{s}, \boldsymbol{\theta}_s)$ is going to normalize the vibration signal $\boldsymbol{x}$ prior to being processed by the AE. The normalized vibration is,

$$\boldsymbol{x}_n = \boldsymbol{x} g(\boldsymbol{s}, \boldsymbol{\theta}_s) \tag{4.3}$$

The normalization operation in Fig. 4.3 is represented by a cross "×". Herein, it refers to the multiplication operation which can be element-wise, sample-wise or others.

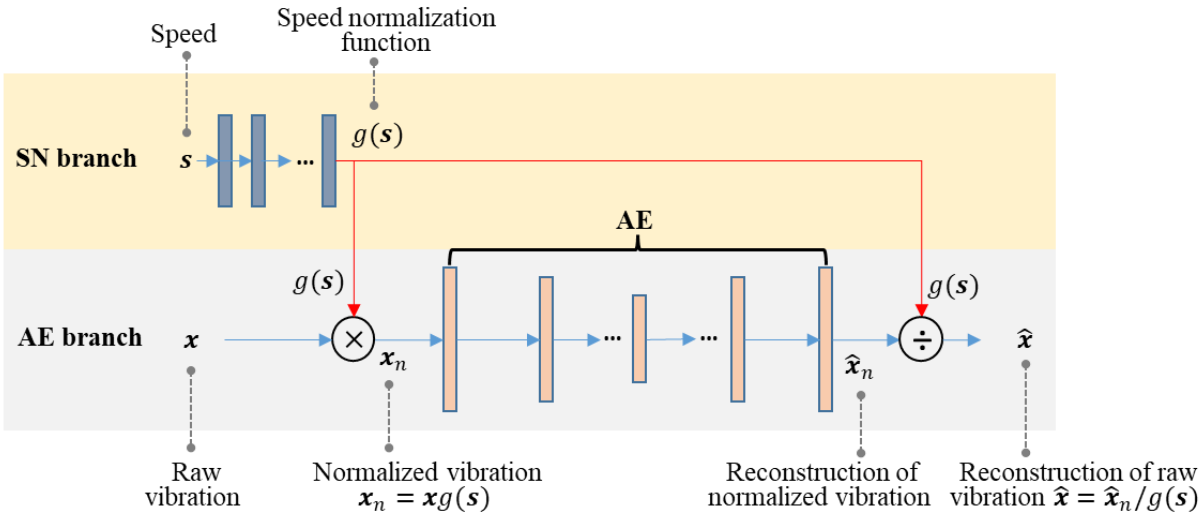


Fig. 4.3: Proposed speed normalized autoencoder (SN-AE).

The normalized vibration $\boldsymbol{x_n}$ is inputted to the AE for reconstruction. Suppose the mapping of the AE is $f(.)$. The reconstructed normalized vibration is,

$$\hat{\boldsymbol{x}}_n = f(\boldsymbol{x}_n, \boldsymbol{\theta}_{\boldsymbol{x}}) = f(\boldsymbol{x}g(\boldsymbol{s}, \boldsymbol{\theta}_{\boldsymbol{s}}), \boldsymbol{\theta}_{\boldsymbol{x}}) \tag{4.4}$$

where, $\boldsymbol{\theta}_{\boldsymbol{x}}$ is the set of trainable parameters of the AE branch. The reconstruction of the input vibration $\boldsymbol{x}$ is the inverse normalization of $\hat{\boldsymbol{x}}_n$,

$$\hat{\boldsymbol{x}} = \frac{\hat{\boldsymbol{x}}_n}{g(\boldsymbol{s},\boldsymbol{\theta}_{\boldsymbol{s}})} = \frac{f(\boldsymbol{x}g(\boldsymbol{s},\boldsymbol{\theta}_{\boldsymbol{s}}),\boldsymbol{\theta}_{\boldsymbol{x}})}{g(\boldsymbol{s},\boldsymbol{\theta}_{\boldsymbol{s}})} \tag{4.5}$$

The inverse normalization operation is represented by a divide symbol "÷" in Fig. 4.3. It refers to the inverse operation of the normalization adopted in Eq. (4.5).

The loss function is the mean squared error of the reconstruction of the vibration,

$$L(\boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{\theta}_{\boldsymbol{s}}) = \frac{1}{2}(\boldsymbol{x} - \hat{\boldsymbol{x}})^2 \tag{4.6}$$

Plug Eq. (4.5) into Eq. (4.6), we get,

$$L(\boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{\theta}_{\boldsymbol{s}}) = \frac{1}{2}\left(\boldsymbol{x} - \frac{f(\boldsymbol{x}g(\boldsymbol{s},\boldsymbol{\theta}_{\boldsymbol{s}}),\boldsymbol{\theta}_{\boldsymbol{x}})}{g(\boldsymbol{s},\boldsymbol{\theta}_{\boldsymbol{s}})}\right)^2 \tag{4.7}$$

Trainable parameters of the AE branch $\boldsymbol{\theta}_{\boldsymbol{x}}$ and the SN branch $\boldsymbol{\theta}_{\boldsymbol{s}}$ are optimized by minimizing the loss function,

$$\boldsymbol{\theta}_{\boldsymbol{x}}^*, \boldsymbol{\theta}_{\boldsymbol{s}}^* = argmin_{\boldsymbol{\theta}_{\boldsymbol{x}},\boldsymbol{\theta}_{\boldsymbol{s}}} L(\boldsymbol{\theta}_x, \boldsymbol{\theta}_s) \tag{4.8}$$

where, $\boldsymbol{\theta}_{\boldsymbol{x}}^*$ and $\boldsymbol{\theta}_{\boldsymbol{s}}^*$ are the optimized parameters. As described in Section 2.1.2, this is an optimization problem and is usually achieved by gradient based methods. We first find the gradient of the loss with respect to parameters, and then update the parameter with a certain learning rate ε,

$$\boldsymbol{\theta}_{\boldsymbol{x}} = \boldsymbol{\theta}_{\boldsymbol{x}} + \varepsilon \frac{\partial L(\boldsymbol{\theta}_{\boldsymbol{x}},\boldsymbol{\theta}_{\boldsymbol{s}})}{\partial \boldsymbol{\theta}_{\boldsymbol{x}}} \tag{4.9}$$

$$\boldsymbol{\theta}_{\boldsymbol{s}} = \boldsymbol{\theta}_{\boldsymbol{s}} + \varepsilon \frac{\partial L(\boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{\theta}_{\boldsymbol{s}})}{\partial \boldsymbol{\theta}_{\boldsymbol{s}}} \tag{4.10}$$

The optima $\boldsymbol{\theta}_{\boldsymbol{x}}^{*}$ and $\boldsymbol{\theta}_{\boldsymbol{s}}^{*}$ are gradually approached when the updating in Eq. (4.9) and Eq. (4.10) meets certain rules. We can also introduce other updating schemes such as the momentum and Adam to speed up the convergence [26]. The process to optimize the parameters is the so-called model training. Note the parameters for the SN branch and the AE branch are trained simultaneously. Eq. (4.9) and Eq. (4.10) only provide the general scheme of the gradients. Exact gradients with respect to certain parameters (e.g., the connection weight between the first neuron of the first layer and the first neuron of the second layer of the SN branch) can be found with the backpropagation algorithm [26].

The reconstruction residual of the normalized vibration $\boldsymbol{x}_n$ instead of that of the raw vibration $\boldsymbol{x}$ is adopted to build the health indicator. That is, the RMS of the $(\boldsymbol{x}_n - \widehat{\boldsymbol{x}}_n)$ not the$(\boldsymbol{x} - \widehat{\boldsymbol{x}})$ will be utilized for fault detection. Because the effects of the speed variation are expected to be removed or mitigated in the residual of normalized vibration, but still preserved in that of the raw vibration. For simplicity, hereinafter and in Fig. 4.3, $g(\boldsymbol{s}, \boldsymbol{\theta}_{\boldsymbol{s}})$ and $f(\boldsymbol{x}_{\boldsymbol{n}}, \boldsymbol{\theta}_{\boldsymbol{x}})$ are written as $g(\boldsymbol{s})$ and $f(\boldsymbol{x}_{\boldsymbol{n}})$, respectively.

The proposed SN-AE is supposed to work because it will remove or at least mitigate the AM effects induced by speed variation through normalization like reported works of [52], [92]. Better performances can be expected as the proposed SN-AE is to be able to automatically remove the effects of speed variation (i.e, AM) and conduct fault detection as a whole. Compared to existing deep learning models such as the AE and its variants, the SN-AE can address the effects of speed variation while the existing cannot, and thus better fault detection performance can be expected. Compared to existing works [52], [92] on removing the effects of speed variation, the proposed

SN features at least two merits. First, the SN is more labor-friendly as it is realized automatically. Second, the proposed SN conduct the normalization and fault detection as a whole and thus an overall optimum can be expected, which generally yields a better performance [26].

In addition to the typical AE as shown in Fig. 4.3, the baseline model can also be the advanced variants of the AE, such as the sparse AE [95], [97], denoising AE [96], AE-LSTM [85], [147] and AE-CNN [148], [149]. Performances over these models will also be evaluated in case studies to show the superiority of the proposed method.

4.3.2 Structure of SN branch

The SN branch is a neural network that intends to learn an SN function. It takes the speed signal as the input and outputs corresponding SN value. The SN function is learned automatically during the training of the SN-AE. The learned SN function normalizes vibration signals to remove speed variation effects.

In this chapter, the CNN is adopted for the SN branch. The intuition is that the CNN preserves local information of speed through moving kernels, which is going to normalize nearby vibration. The adopted structure for the SN branch is shown in Fig. 4.4. It has a typical 1D-CNN structure consisting of several convolutional (Conv) layers and a global average pooling (GAP) layer at the end. The GAP is applied across channels not the time points to preserve the data shape. The activation functions are the commonly used ReLU expect for the last layer, with which a Sigmoid activation is adopted to limit the range of the SN function $g(\boldsymbol{s})$ within (0, 1). Hyperparameters such as the number of convolutional layers (nl), kernel size (ks) and number of channels (ch) will be analyzed in case studies in the next section.

The learned SN function $g(\boldsymbol{s})$ will multiply the input vibration through an element-wise manner. Suppose we have a vibration data sample $\boldsymbol{x} = \{x_i\}_{i=1}^{n}$ and a corresponding speed data sample $\boldsymbol{s} = \{s_i\}_{i=1}^{n}$, where n refers to the number of data points (or length) of the sample. Element-wise multiplication requires that $g(\boldsymbol{s})$ has a same dimension as $\boldsymbol{x}$, i.e., $g(\boldsymbol{s}) = \left\{\left(g(\boldsymbol{s})\right)_{\boldsymbol{i}}\right\}_{i=1}^{n} \in \mathbf{R}^n$. The normalized vibration is,

$$\boldsymbol{x}_n = \boldsymbol{x}\, g(\boldsymbol{s}) = \left\{x_i\left(g(\boldsymbol{s})\right)_i\right\}_{i=1}^{n} \tag{4.11}$$

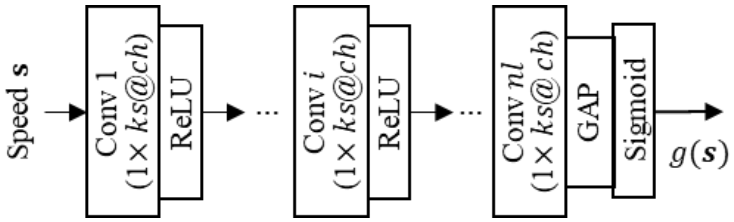


Fig. 4.4: Structure for SN branch. nl – number of convolutional layers, ks – kernel size, ch – number of channels.

The SN function $g(\boldsymbol{s})$ is expected to decrease with respect to speed like the signal processing methods [52], [92]. Such that the effects induced by speed variation would be removed. In the next section, we are going to evaluate the performance of the proposed SN-AE over three experimental datasets.

We acknowledge there may be better structures for the SN branch. Other possible structures can be a feedforward neural network like the AE, or a recurrent neural network, or others. But comparison of different networks is not the focus of this chapter. Once the effectiveness of proposed SN-AE with a CNN-shaped SN branch is validated, we can easily explore other possible implementations for the SN branch in our future studies.

4.3.3 Hyperparameter selection for SN branch

As indicated in the Section 4.1, existing hyperparameter selection strategies use the VMSE as the measure to guide the selection [26], [158], [159]. The best hyperparameters are selected when the VMSE are minimized. The VMSE can be applied to the baseline AE. Since it measures the reconstruction performance of the AE. A smaller VMSE generally yields a better fault detection performance [161]. For the proposed SN-AE, we need to care about the speed effects removal performance as the SN is proposed for this purpose. A minimal VMSE may guarantee a good reconstruction performance but cannot assure a good speed effects removal performance. As the scale of the learned SN function can vary across hyperparameters. A smaller VMSE may be because of relatively smaller SN values, not better speed effects removal performance. To address this problem, we propose a new measure, the Pearson correlation coefficient between the health indicator and the speed with the validation set (VCOR), for the hyperparameter selection for the SN branch. The rationale is as follows. The VCOR measures the linear dependency between two variables. Its absolute value is in between [0, 1]. Hereinafter, the VCOR refers to its absolute unless specifically indicated. A larger VCOR indicates that two variables are more likely correlated, otherwise less correlated. Based on our evaluation, the health indicator returned by the baseline AE and the speed are highly correlated, and the VCOR is large. Such phenomenon can be observed in case studies of this chapter. Remember the goal of the proposed SN is to remove the effects of speed variation. Once such effects are removed, the correlation between the health indicator and speed is supposed to be mitigated, and thus the VCOR would be smaller. A better fault detection performance can be expected accordingly.

In this chapter, the scope of the hyperparameter selection is limited to the SN branch only, that is, the number of layers (nl), kernel size (ks) and the number of channels (ch), for the SN branch. For simplicity, let $\mathbf{\Lambda} = \{nl, ks, ch\}$ herein and after. We assume the baseline model already performs well with the reconstruction task and its hyperparameters are fixed. The widely used grid search method [26] will be adopted together with the proposed measure VCOR to conduct hyperparameter selection for the SN branch. We acknowledge there are many advanced hyperparameter searching methods, such as the random search [26], Bayesian optimization [158] and differentiable architecture search [159]. Since the hyperparameter search method is not the focus of this chapter, we choose the grid search due to its simplicity. Once the proposed searching measure, i.e., the VCOR, is verified effective, it is supposed to be applicable with other searching methods. Detailed steps of hyperparameter selection using the grid search incorporating the proposed VCOR are given below.

Step 1: Configure a validation set which contains healthy data only which is never used in the model training or model testing.

Step 2: Reconstruct the validation set with a well-trained SN-AE with a certain combination of hyperparameters in $\mathbf{\Lambda}$.

Step 3: Calculate the health indicator, i.e., RMS of the reconstruction residual or error (err), of every single sample in the validation set.

Step 4: Calculate the mean speed (sm) of every single sample in the validation set.

Step 5: Calculate the correlation coefficient between err and sm with the following equation,

$$\text{VCOR} = corr(err, sm) = \frac{\sum_i^{NV}(err_i - \overline{err})(sm_i - \overline{sm})}{\sqrt{\sum_i^{NV}(err_i - \overline{err})^2 \sum_i^{NV}(sm_i - \overline{sm})^2}} \tag{4.12}$$

where, $\overline{err}$ and $\overline{sm}$ refer to the average of err and sm, NV means the number of samples in the validate set and $corr$ refers to the function to find correlation coefficient.

Step 6: Repeat Steps 2 – 5 to find VCORs for all candidate hyperparameter combinations in $\boldsymbol{\Lambda}$.

Step 7: Select optimal hyperparameters for the SN branch. The optimal hyperparameters $\boldsymbol{\Lambda}^*$are returned when the absolute value of the VCOR is minimized,

$$\boldsymbol{\Lambda}^* = argmin_{\boldsymbol{\Lambda}}\text{VCOR} \tag{4.13}$$

Given the above description of the proposed SN-AE, an overall flowchart of using the proposed SN-AE to conduct fault detection is given in Fig. 4.5. The overall method consists of three stages including a training stage, a validation stage, and a test stage. In the training stage, we train the proposed SN-AE using the training set to optimize the parameters $\boldsymbol{\theta_x}$ and $\boldsymbol{\theta_s}$, of the SN-AE for each hyperparameter combination in $\boldsymbol{\Lambda}$. The optimal parameters $\boldsymbol{\theta_x^*}$ and $\boldsymbol{\theta_s^*}$ are reached when the loss $L(\boldsymbol{\theta}_x, \boldsymbol{\theta}_s)$ is minimized. In the validation stage, we conduct optimal hyperparameter selection for the SN branch. The optimal hyperparameters are obtained when the correlation coefficient between the RMS of the reconstruction residual of the normalized vibration and the mean of the speed, i.e., the VCOR over the validation set, is minimized. In the test stage, the selected model is used to conduct fault detection over the test set. The RMS of the reconstruction residual is taken as the health indicator. The health indicator is compared with a predefined threshold to determine the health state of a machine. If the health indictor is small than the threshold, the machine is deeded healthy, otherwise faulty. As mentioned in Section 4.2, in this chapter, we do not provide

a specified threshold but try all feasible thresholds, and use an integrated performance measure, i.e., the AUC [32], [160], to show the fault detection performance of the proposed SN-AE.

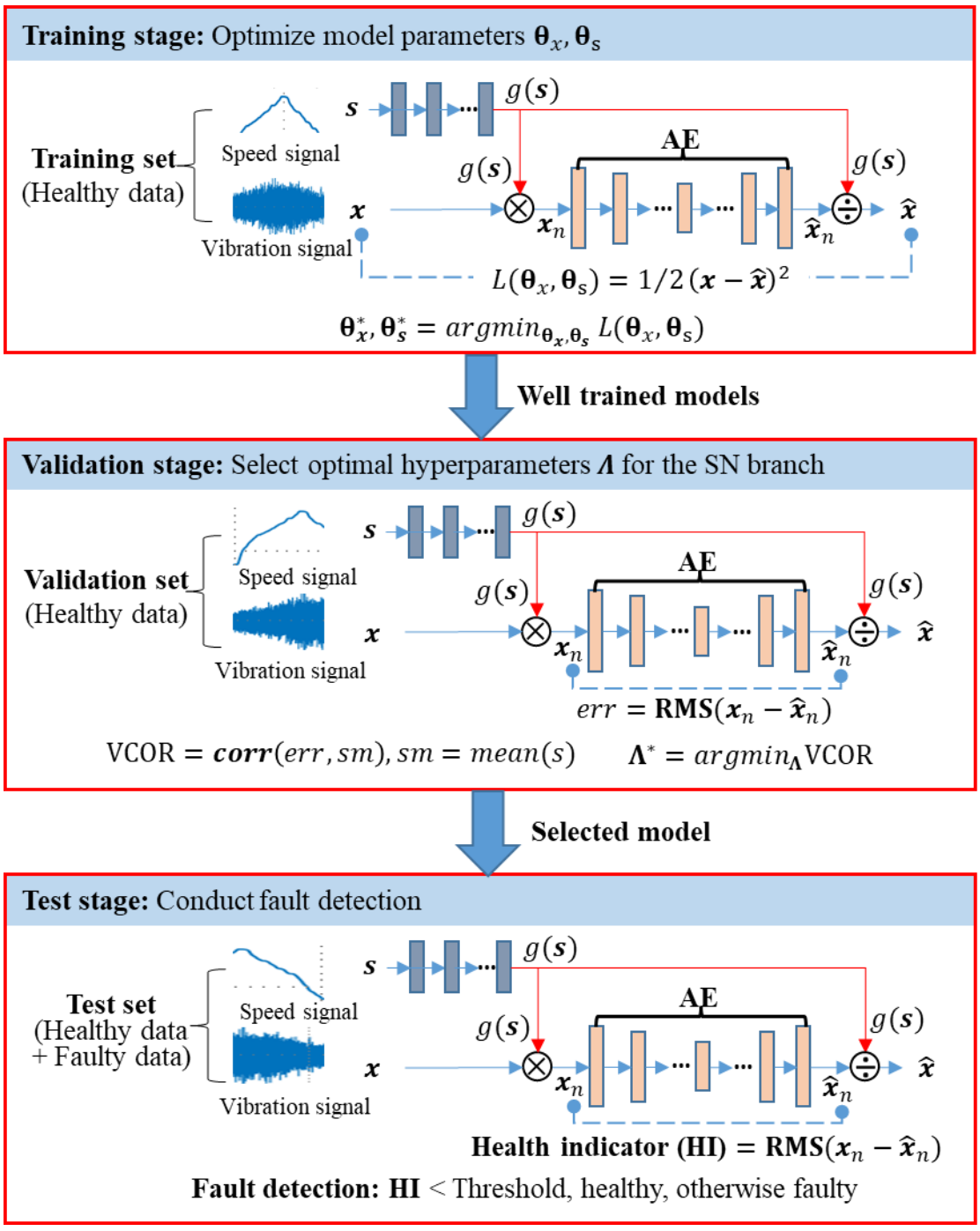


Fig. 4.5: Overall flowchart of the proposed fault detection method.

4.4 Case studies

This section presents three case studies to validate the effectiveness of the proposed SN-AE over three different experimental datasets, respectively. The datasets used include a planetary gearbox dataset, a fixed-shaft gearbox dataset and a bearing dataset.

4.4.1 Case study 1: Planetary gearbox dataset

4.4.1.1 Dataset description

The planetary gearbox dataset was collected in 2021 at Tsinghua University, Beijing, China by the author and colleagues [162]. The teg rig is shown in Fig. 4.6. It consists of motor, a planetary gearbox, and a magnetic power break. Three accelerometers were mounted on the planetary gearbox to collect its acceleration signals from the vertical, lateral and the horizontal direction, respectively. A current sensor was used to collect the current signal of the motor. An encoder was installed in between the motor and the planetary gearbox to measure the rotating speed.

The data was collected under multiple health states and multiple speed conditions. The health states contain 16 states including the healthy state, bearing fault, ring gear fault, sun gear fault and planetary gear fault, and different fault severities, as shown in Fig. 4.7. The speed conditions include constant speed conditions and varying speed conditions. The constant speed condition contains five speed levels of 300 rpm, 600 rpm, 900 rpm and 1500 rpm. The varying speed conditions changes the speed continuously in between 300 rpm and 1500 rpm. Ten repeating tests were conducted for each health state and each health condition. Each test lasted for 60 s. The

sampling rate was 20 kHz. An example of collected speed signal and corresponding acceleration signal is given in Fig. 4.8.

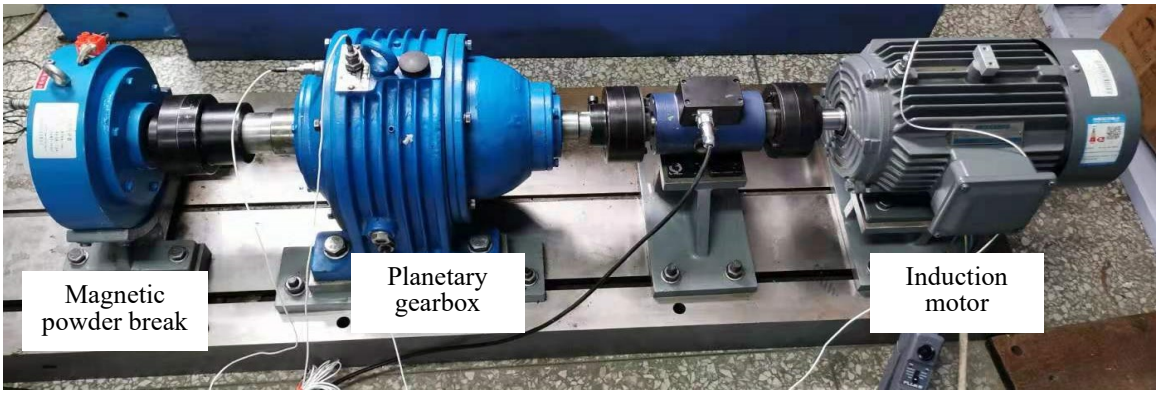


Fig. 4.6: Planetary gearbox test rig.

In this chapter, only the healthy data is going to be used to train the model. The data collected under an incipient fault, i.e., the planet gear tooth root crack, is going to serve as the faulty data to test the performance of models. The vibration in the vertical direction and speed signal are to be used. Other data will not be used. The data is preprocessed in the following steps before being further processed by deep learning models.

(1) Low-pass with a cut-off frequency of 2.5 kHz and then down-sample data from 20 kHz to 5 kHz for both the acceleration and speed signals.

(2) Segment data into short samples with a length of 2000 data points (0.4 s) without overlap.

(3) Delete outliers to balance the distribution across speed. The outliers here are defined as samples with rare speed values (i.e., outside the range of 300 rpm – 1500 rpm) or samples with corrupted speed collection.

(4) Split the data into a training set, a validation set and a test set. The training set and the validation set contain healthy data only. The test set contains half healthy data and half faulty data.

(5) For the training set and the validation set, the data is further segmented into a shorter length of 250 data points (0.05 s) to speed up the training process.

Fig. 4.7: Seeded faults of planetary gearbox: (a) Ring gear tooth missing, (b) Sun gear chipped tooth, (c) Sun gear tooth missing, (d) Planetary gear tooth root crack, (e) Planetary gear chipped tooth, (f) Planetary gear tooth missing, (g) Planetary bearing inner race crack (hereinafter, bearing race crack means the crack width = 0.4 mm), (h) Planetary bearing inner race fault (hereinafter, bearing race fault means induced crack width = 2 mm), (i) Planetary bearing outer race crack, (j) Planetary bearing outer race fault, (k) Planetary bearing rolling element fault, (l) Input shaft bearing inner race crack, (m) Input shaft bearing inner race fault, (n) Input shaft bearing outer race crack and (o) Input shaft bearing outer race fault. Seeded faults are marked in red rectangles.

The preprocessed data is configured as shown in Table 4.1. The training data and the test data have different lengths. The training data is shorter to reduce the model scale and ultimately speed up the training. The test data is longer to assure that at least one cycle is experienced, and the tooth fault is meshed once. Such configuration is realized as follows. The number of neurons of the input layer of the AE branch is identical to the data length of the training set, i.e., 250. In the training stage, the SN-AE directly uses the training set. In the test stage, a test sample is firstly split into consecutive sequences without overlap which have the same length as the training data. Each sequence is reconstructed with the well-trained AE separately. Reconstructed sequences belonging to a same test sample are concatenated. The concatenated sequence is then compared with the test data to find the reconstruction residual, which is further used for fault detection. The validation set is utilized in a same way as the training set and thus will not be detailed.

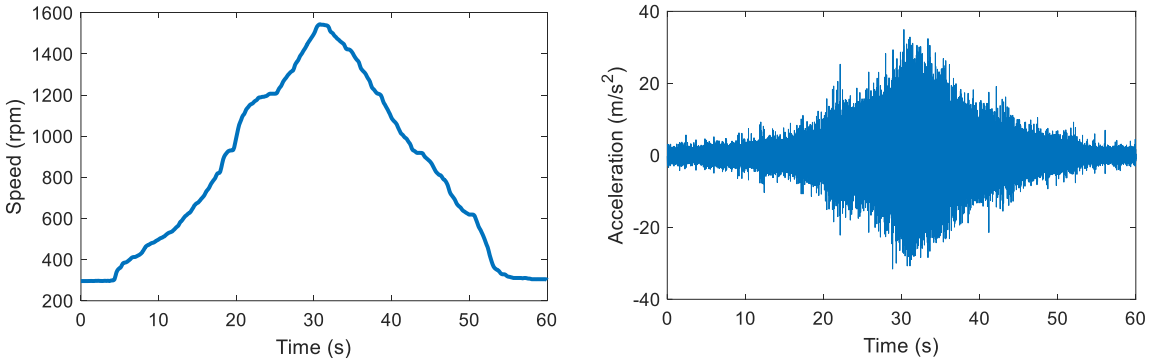


Fig. 4.8: Example of collected data of the planetary gearbox dataset (Healthy): (a) Speed and (b) Acceleration.

Table 4.1: Summary of the planetary gearbox dataset (Continuous varying in between 300~1500 rpm).

Sub-set	# of samples	Sample length
Training set (Healthy only)	2428	250 (0.05 s)
Validation set (Healthy only)	1620	250 (0.05 s)
Test set (Healthy + Faulty)	129 + 129	2000 (0.4 s)

4.4.1.2 Model setting

The baseline AE contains five layers as follows,

- Number of layers: 5.
- Numbers of neurons: 250 (Input) – 128 – 64 – 128 – 250 (Output). The number of neurons of the input layer and the output layer is identical to the training data length. For the planetary gearbox dataset, it is 250.
- Activations: Linear for the output layer and ReLU for others.

The proposed SN-AE consists of two branches, i.e., the AE branch and the SN branch. The structure of the AE is identical to the baseline AE as detailed above. The SN branch is a CNN-shaped network. We will explore the impacts of hyperparameters regarding the SN structure by trying out several feasible values as follows.

- Number of layers nl: 1, 2 and 3
- Kernel size ks: 3, 5, 7 and 9
- Number of channels ch: 4, 8, 16 and 32

Given these values, we have 48 feasible hyperparameter combinations. Other hyperparameters regarding the model training are as follows. The batch size is 64. The maximum epoch is 300. The optimization method is the Adam [26]. The learning rate is 0.0002. The models were coded in Python using the framework of Keras and run at Google Colab. Five repeating trials were conducted for each case. Results are shown below.

4.4.1.3 Results

The fault detection performance of the baseline AE and the proposed SN-AE with different hyperparameters are evaluated. The AUC [160] is used to measure the fault detection performance, which provides a comprehensive measure of false positive rates and true positive rates for fault detection. The results with the test set are given in Fig. 4.9. The curve shows the average AUC of five repeating trials. The error bars are the corresponding standard derivations. A range in between two vertical dash lines contain results of four kernel sizes of 3, 5, 7 and 9 with a certain number of channels. Four consecutive ranges cover results of a certain number of layers. Major observations from Fig. 4.9 are listed below.

(1) The AUC of the bassline AE is 0.6863±0.0206. The highest AUC of the proposed SN-AE is 0.9704±0.0087 achieved at $(nl = 3, ch = 32, ks = 9)$. The lowest AUC of the proposed SN-AE is 0.8101±0.0866 returned when $(nl = 2, ch = 4, ks = 3)$. The proposed SN-AE archives significantly higher AUCs regardless of the optimization of hyperparameters of the SN branch.

(2) The number of layers of $nl = 2$ is preferred out of 3 candidate numbers of layers, as it returns relatively stable and fairly high AUCs. The number of channels of $ch = 32$ is preferred out of 4 candidate numbers of layers, as it features fairly high AUCs. When $nl < 3$ and $ch < 16$, a lager kernel size (ks) is preferred, otherwise no clear clues are observed from Fig. 4.9.

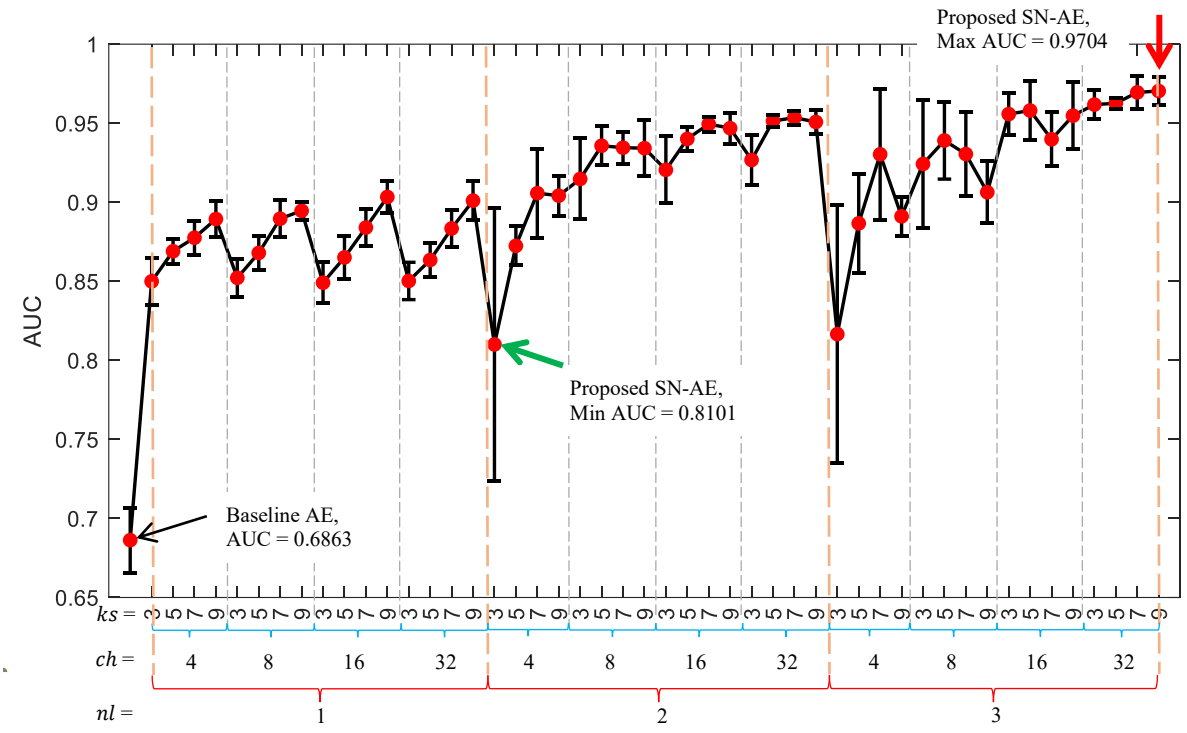


Fig. 4.9: Fault detection performance of the proposed SN-AE over planetary the gearbox dataset. nl – number of layers of the SN branch, ks – kernel size and ch – number of channels.

4.4.1.4 Hyperparameter selection for SN branch

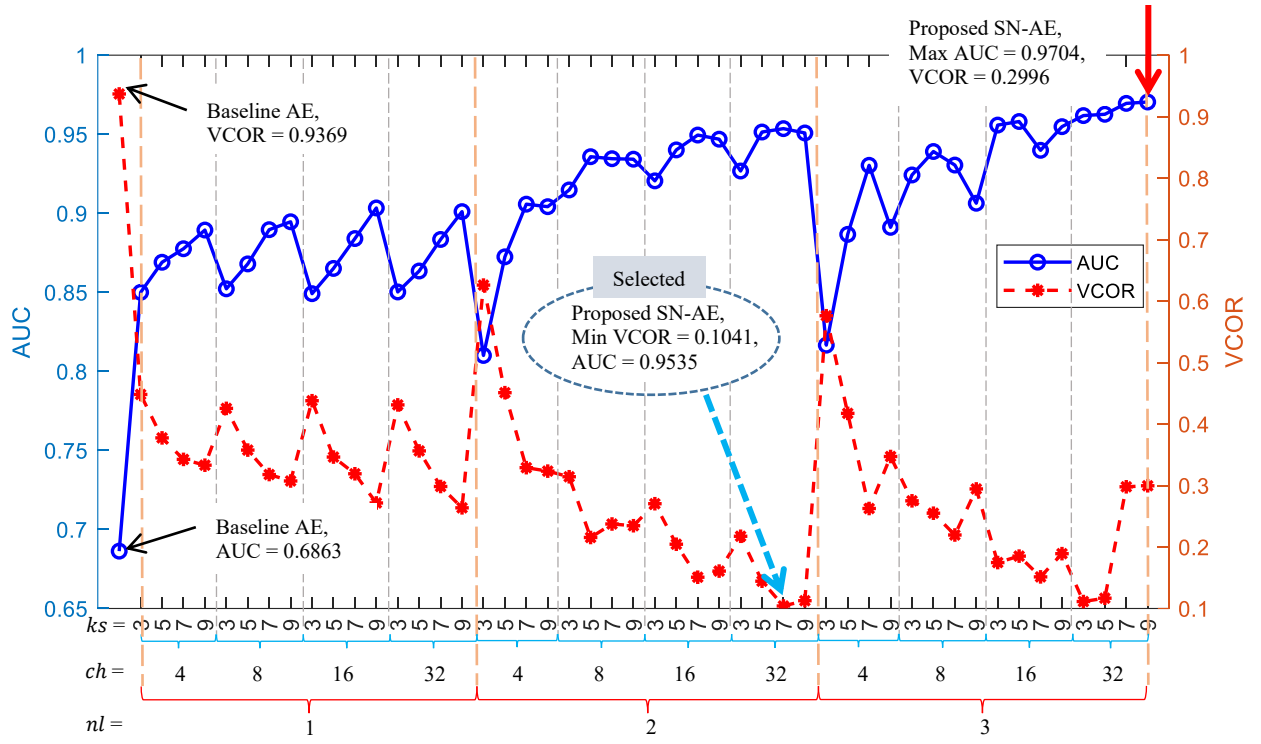
Following the proposed strategy in Section 4.3.3, we calculate the correlation coefficient VCOR between the health indicator and the speed of the validation set for hyperparameter selection for the SN branch. The resultant absolute VCOR of each hyperparameter combination is shown in Fig. 4.10(a). The VMSE of the reconstruction residual is shown in Fig. 4.10(b) for comparison. The AUC is also shown for references. The results including the VCOR and the VMSE are the average of the five repeating trials. The error bars are not shown for a better readability of the figures.

From Fig. 4.10(a), we can see that the VCOR of the validation set generally shows a reverse trend as that of the AUC. A smaller VCOR generally returns a larger AUC, and vice versa. This indicates the feasibility of using VCOR to conduct hyperparameter selection for the SN branch.

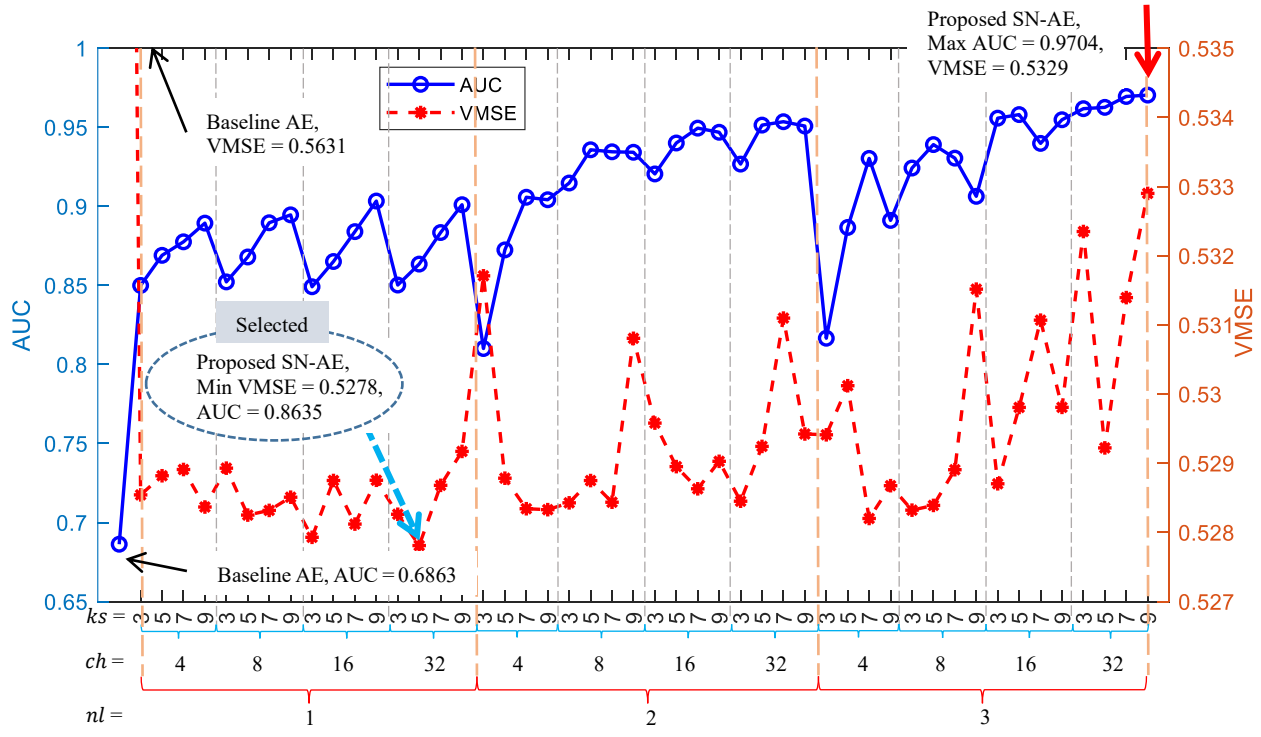
From Fig. 4.10(b), we can see that the VMSE of the validation set even sometimes illustrates a reverse trend as that of the AUC, but there are many scenarios wherein this trend does not hold (e.g, $nl = 3$). This means the VMSE is less favorable for the hyperparameter selection for the SN branch, as mentioned in Section 4.3.3.

The largest VCOR is obtained with the baseline AE, and as expected, the AUC is the lowest. For the proposed SN-AE, the smallest VCOR is pinpointed with a dash arrow in Fig. 4.10(a). For comparison, the highest AUC with the test set is also marked but with a solid arrow. We did the same work for the Fig. 4.10(b). Corresponding values are summarized in Table 4.2. We can see that the hyperparameters selected by the proposed VCOR returns an AUC of 0.9535, which quite approaches the highest AUC of 0.9704 and is significantly higher than the baseline AE of 0.6863. This means that using the proposed measure VCOR of the validation set, we can select a good SN-AE for the purpose of fault detection.

The AUC returned by the minimal VMSE is 0.8635. It is also significantly higher than that of the baseline AE, but much smaller than the one selected by the proposed VCOR. This shows the superiority of the proposed VCOR hyperparameter selection for the SN branch. In the following, the SN-AE with hyperparameters of $(nl = 2, ch = 32, ks = 7)$ selected by the proposed VCOR is used for further analysis.



(a)



(b)

Fig. 4.10: Hyperparameter selection for the SN branch of the proposed SN-AE over the planetary gearbox dataset: (a) Proposed VCOR and (b) Existing VMSE.

Table 4.2: Hyperparameter selection for the SN branch with the planetary gearbox dataset.

Selection method	AUC	VCOR	VMSE	Hyperparameters
Highest AUC with the test set	**0.9704**	0.2996	0.5329	(3, 32, 9)
Existing VMSE	0.8635	0.3563	**0.5278**	(1, 32, 5)
Proposed VCOR	0.9535	**0.1041**	0.5311	(2, 32, 7)

Notes: 1. The hyperparameters are in the form of (nl, ch, ks). 2. AUC of the baseline AE is 0.6863.

4.4.1.5 Learned SN function

The learned SN function $g(s)$ is shown in Fig. 4.11. The hyperparameters are selected by the proposed VCOR, i.e., $(nl = 2, ch = 32, ks = 7)$. The SN function out of other hyperparameters are not shown for simplicity. We can see that the $g(s)$ decreases with speed as expected. Such decreasing trend removes the effects induced by speed variation.

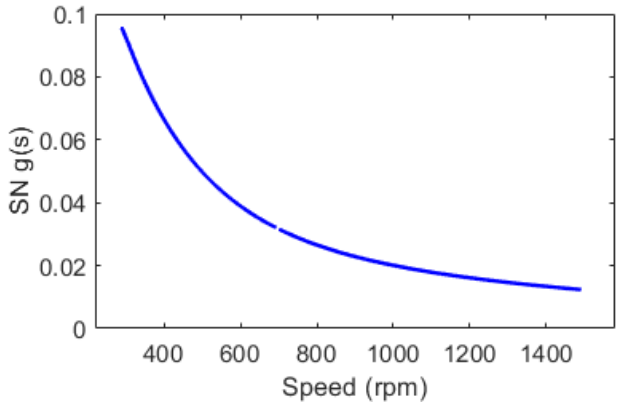


Fig. 4.11: Learned SN function of the proposed SN-AE over the planetary gearbox dataset with selected optimal hyperparameters of $(nl = 2, ch = 32, ks = 7)$.

To illustrate effects of speed variation on the health indicator, we plot the health indicator, i.e., the RMS of the normalized reconstruction residual versus the speed, as shown in Fig. 4.12. The corresponding receiver operator characteristic curve is also shown for reference. We can see that,

(1) In Fig. 4.12(a), the health indicator shows a clear increasing trend with speed. This means the health indicator constructed by the baseline AE is affected by the speed variation.

(2) In Fig. 4.12(b), the increasing trend induced by speed is pretty much mitigated, compared to the baseline AE as shown in Fig. 4.12(a). This means the effects of speed variation are successfully removed with the proposed SN-AE.

(3) The amplitude of health indicator of the SN-AE (Fig. 4.12(b)) is much smaller than that of the baseline AE (Fig. 4.12(a)). This is because the vibration amplitude is reduced when applying the SN function. This does not matter for fault detection as we do not care about the absolute value of health indicators but the care about the relative values.

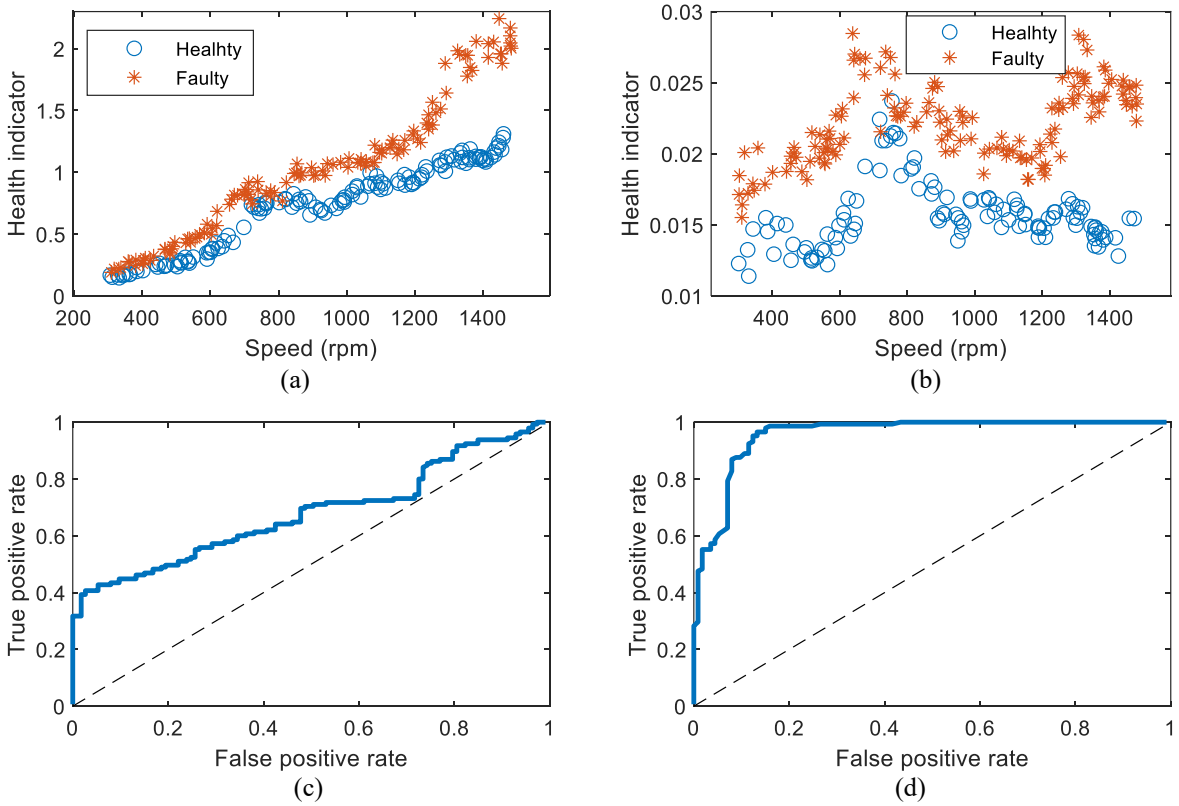


Fig. 4.12: Health indicator versus speed over the planetary gearbox dataset: (a) Baseline AE, (b) Proposed SN-AE with selected hyperparameters of $(nl = 2, ch = 32, ks = 7)$, (c) Receiver operator characteristic curve for (a), AUC = 0.6863 and (d) Receiver operator characteristic curve for (b), AUC = 0.9535.

4.4.2 Case study 2: Fixed-shaft gearbox dataset

4.4.2.1 Dataset description

The fixed-shaft gearbox dataset [144] was collected at the University of Alberta, Edmonton, Alberta, Canada in 2018 by the author and author's colleagues. This dataset contains acceleration signals and speed signals of a fixed-shaft gearbox with different health states under varying speed conditions. The health states include the healthy state and the faulty state with five severity levels of tooth root cracks. See more information of this dataset in Section 3.3.3.

In this chapter, data under the healthy state and the faulty state with the most incipient tooth root crack (shown in the very left panel of Fig. 3.7(a)) will be used. Only a single channel of vibration, i.e., the vertical direction, will be adopted. The data with the most incipient tooth root crack fault is taken as the faulty data. The fixed-shaft gearbox dataset is preprocessed in a same way as that for the planetary gearbox dataset. The resultant dataset is summarized in Table 4.3. The samples are evenly distributed across speed.

Table 4.3: Summary of the fixed-shaft gearbox dataset (Continuously varying in between 50 ~ 180 rpm).

Sub-set	# of samples	Sample length
Training set (Healthy only)	510	256 (0.2 s)
Validation set (Healthy only)	340	256 (0.2 s)
Test set (Healthy + Faulty)	110 + 110	512 (0.4 s)

4.4.2.2 Results

The hyperparameters for the SN-AE model are the same as that for the planetary gearbox dataset, expect that the number of neurons of the input and the output layers is changed to 256 to fit in the fixed-shaft gearbox dataset. The fault detection performances are shown in Fig. 4.13. Similar

trends as that of the planetary gearbox dataset are observed. The largest AUC is 0.9252 ± 0.0043, achieved at $(nl = 3, ch = 8, ks = 9)$. The lowest AUC is 0.9047 ± 0.0098, returned by $(nl = 3, ch = 4, ks = 7)$. They are both significantly improved from 0.8652 ± 0.0071, which is obtained by the baseline AE. The hyperparameters has limited impacts on the fault detection AUC for the fixed-shaft gearbox dataset. Therefore, less layers, less channels and smaller kernel sizes are preferred to reduce the computation load.

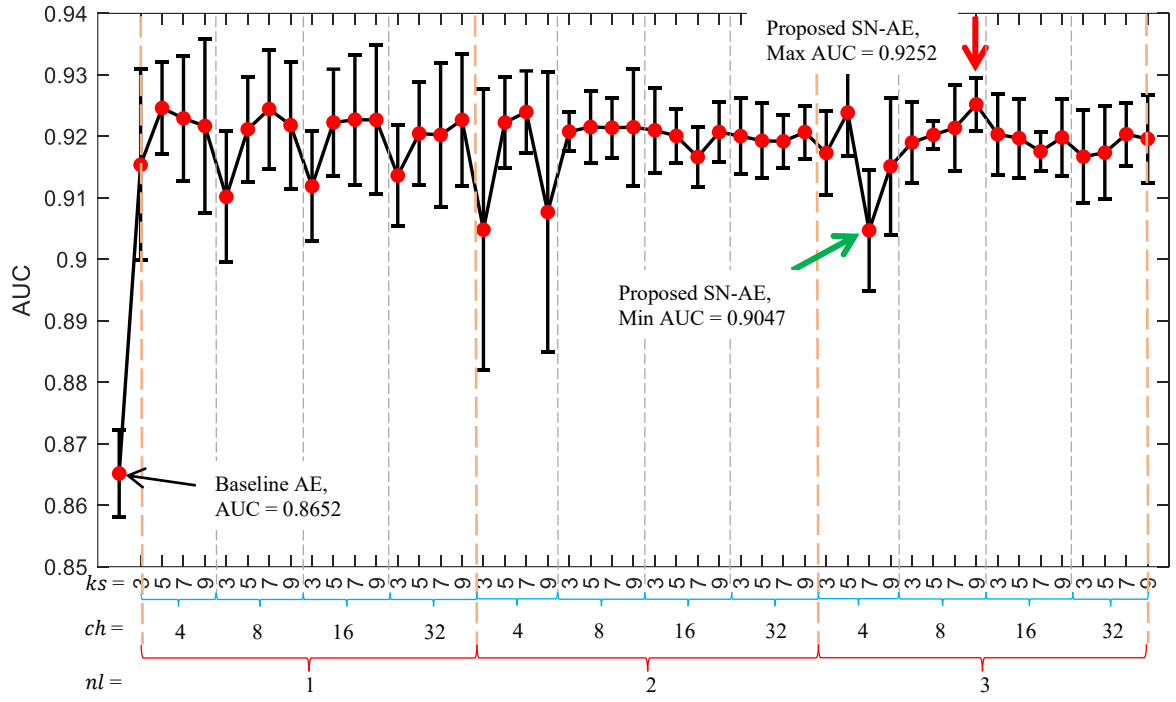
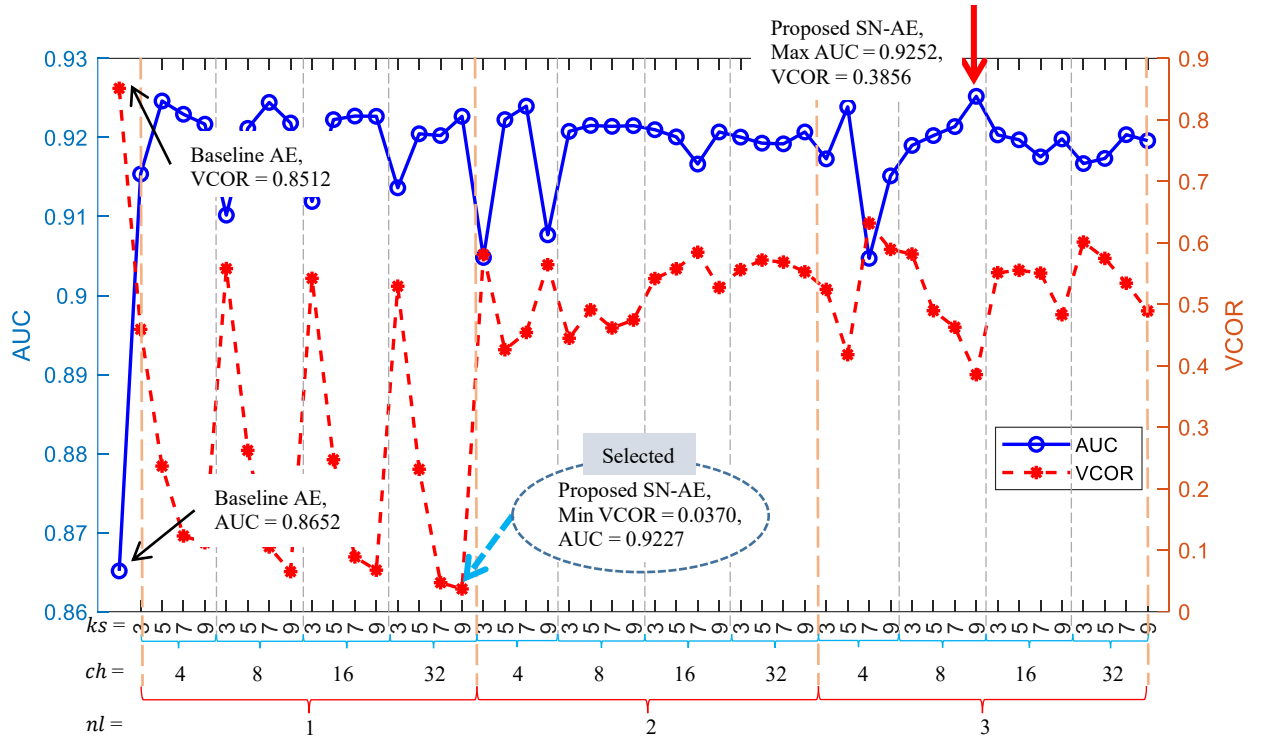
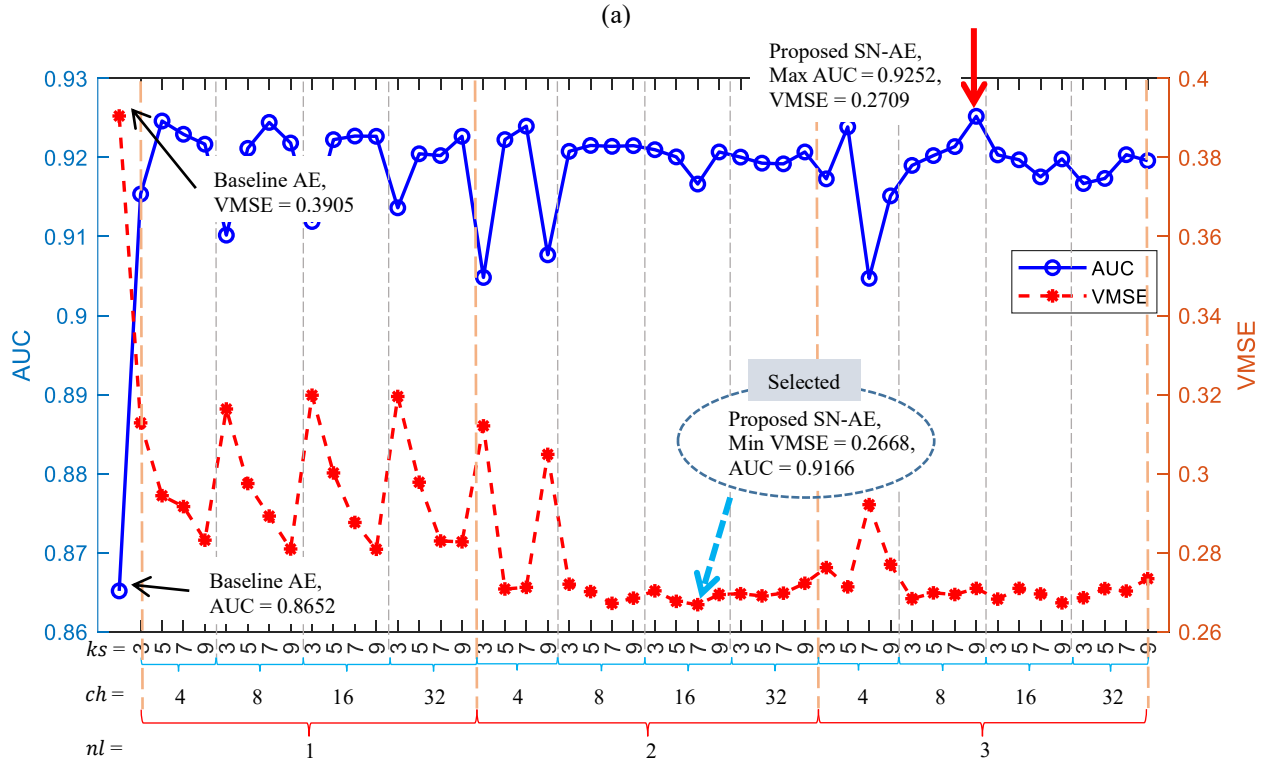


Fig. 4.13: Fault detection performance of the proposed SN-AE over the fixed-shaft gearbox dataset. nl – number of layers of the SN branch, ks – kernel size and ch – number of channels.

Likewise, we also conducted hyperparameter selection for the fixed-shaft gearbox dataset. The results of both the proposed VCOR and the existing VMSE are given in Fig. 4.14. The selected hyperparameters are summarized in Table 4.4.



(a)



(b)

Fig. 4.14: Hyperparameter selection for the SN branch of the proposed SN-AE over the fixed-shaft gearbox dataset: (a) Proposed VCOR and (b) Existing VMSE.

We can observe similar trends as that of the planetary gearbox dataset. The difference is that the VMSE returns quite good fault detection performance. A problem observed with the proposed VCOR is that even the VCOR always shows a reverse trend with the AUC, the amplitude of the VCOR with $nl > 1$ is relatively large. Nevertheless, this does not prevent the proposed VCOR from returning high fault detection AUC for the fixed-shaft gearbox dataset, as shown in Table 4.4. The learned SN function of the proposed SN-AE with selected hyperparameters of $(nl = 1, ch = 32, ks = 9)$ by the proposed VCOR is shown in Fig. 4.15. The corresponding health indicator versus speed is shown in Fig. 4.16. Like the planetary gearbox dataset, the increasing trend in the health indicator which is induced by speed variation, is removed, and the learned SN function decreases with speed as expected.

Table 4.4: Hyperparameter selection for the SN branch with the fixed-shaft gearbox dataset.

Selection method	AUC	VCOR	VMSE	Hyperparameters
Highest AUC with the test set	**0.9252**	0.3856	0.2709	(3, 8, 9)
Existing VMSE	0.9166	0.5846	**0.2668**	(2, 16, 7)
Proposed VCOR	0.9227	**0.0370**	0.2827	(1, 32, 9)

Notes: 1. The hyperparameters are in the form of (nl, ch, ks). 2. AUC of the baseline AE is 0.8652.

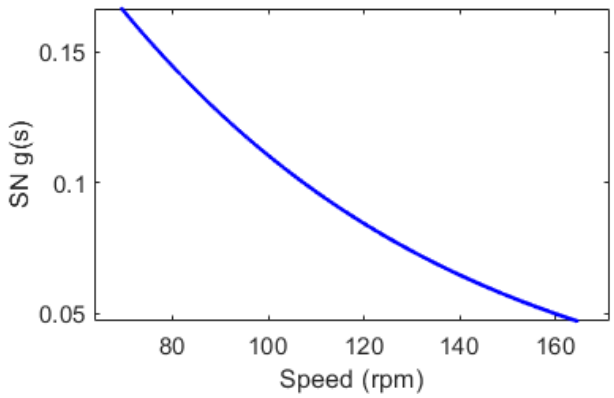


Fig. 4.15: Learned SN function of the proposed SN-AE over the fixed-shaft gearbox dataset with selected hyperparameters of $(nl = 1, ch = 32, ks = 9)$.

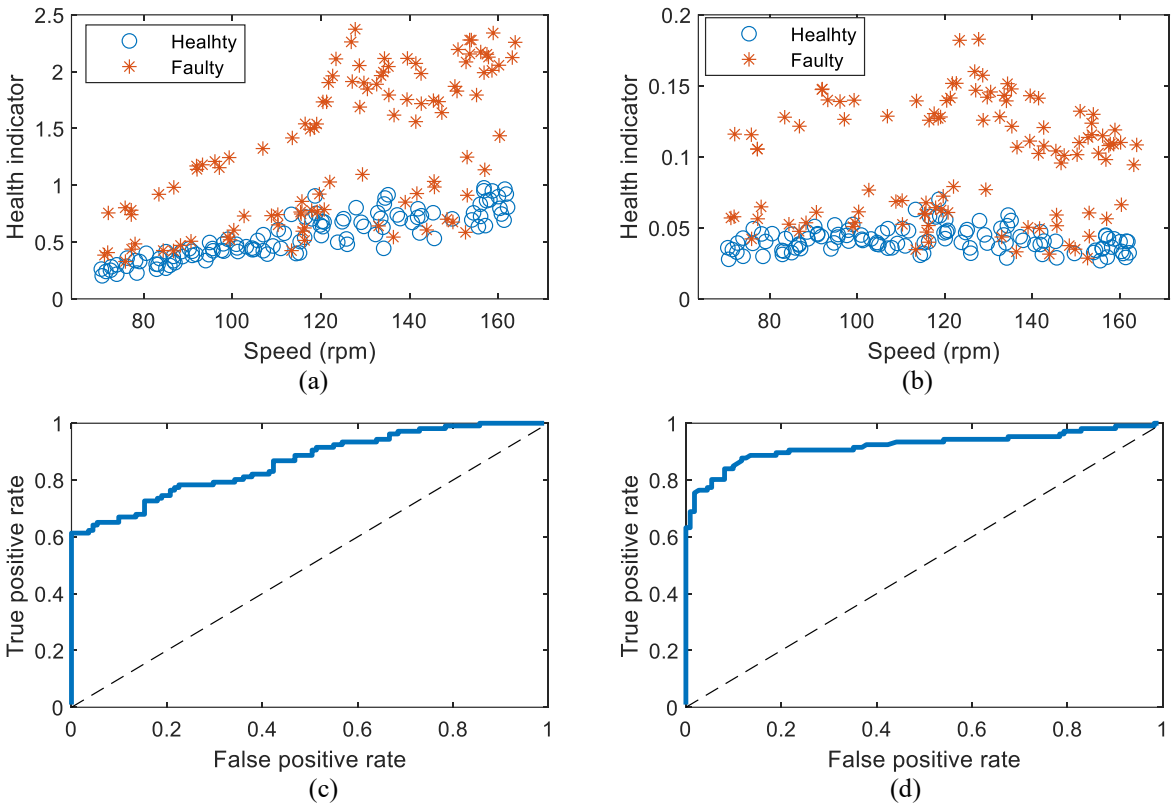


Fig. 4.16: Health indicator versus speed over the fixed-shaft gearbox dataset: (a) Baseline AE, (b) Proposed SN-AE with selected hyperparameters of $(nl = 1, ch = 32, ks = 9)$, (c) Receiver operator characteristic curve for (a), AUC = 0.8652 and (d) Receiver operator characteristic curve for (b), AUC = 0.9227.

4.4.3 Case study 3: Bearing dataset

4.4.3.1 Dataset description

The bearing dataset [163] is an open access dataset which was released by professionals with the Department of Mechanical Engineering, University of Ottawa, Ottawa, Ontario, Canada in 2018. The dataset can be accessed at http://dx.doi.org/10.17632/v43hmbwxpm.1. It contains acceleration signals and speed signals collected from rolling element bearing test rig with different health states under varying speed conditions. The experimental setup is shown in Fig. 4.17. The health states include healthy, faulty with an inner race defect and faulty with an outer race defect. The speed

varied continuously in between 900 – 1500 rpm. An example of collected speed signal and corresponding vibration signals is shown in Fig. 4.18. The collected data is preprocessed in the same way as that for the planetary gearbox dataset. The resultant dataset is summarized in Table 4.5. Over here, the faulty data is the data collected with the outer race fault.

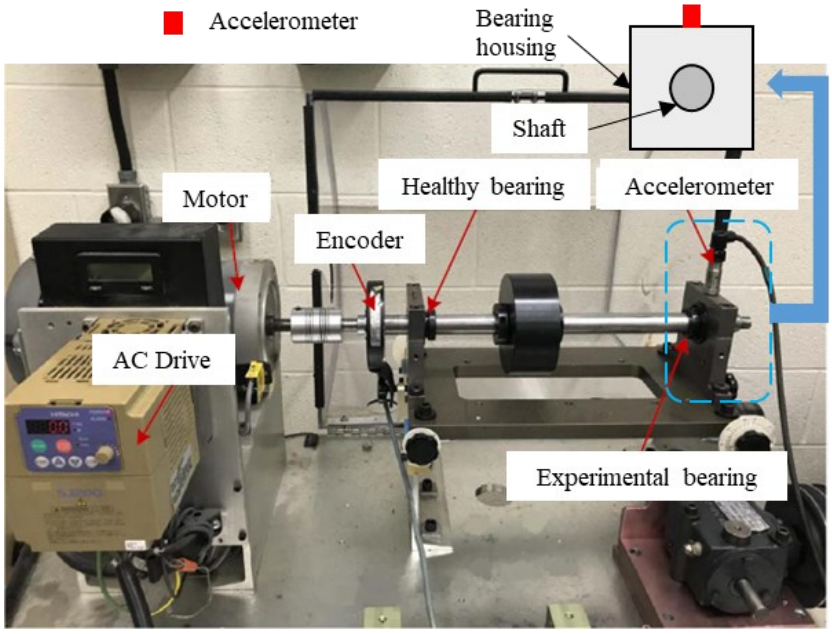


Fig. 4.17: Experimental test rig for the bearing dataset [163].

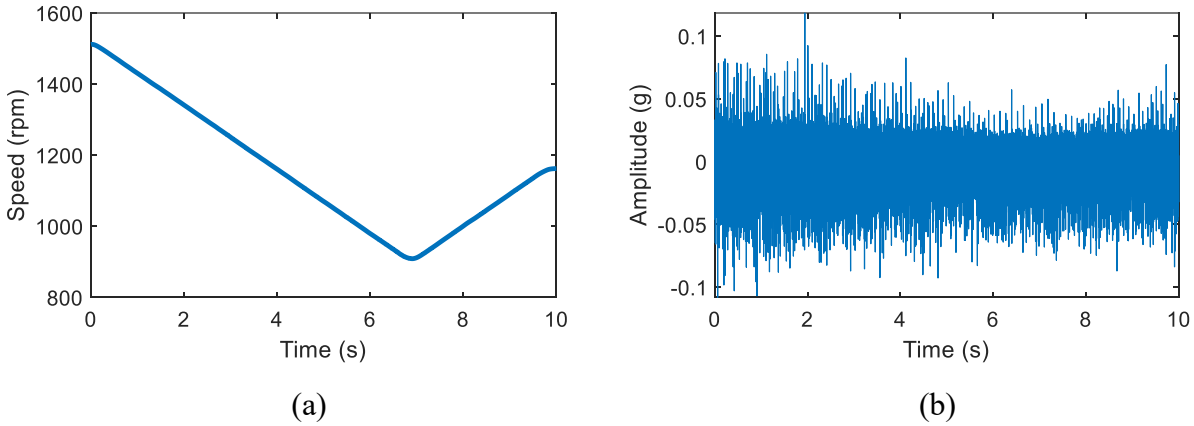


Fig. 4.18: Example of collected data in the bearing dataset: (a) Speed and (b) Acceleration.

Table 4.5: Summary of the bearing dataset (Continuously varying in between 900 ~ 1450 rpm).

Sub-set	# of samples	Sample length
Training set (Healthy only)	558	250 (0.05 s)
Validation set (Healthy only)	372	250 (0.05 s)
Test set (Healthy + Faulty)	190 + 190	500 (0.1 s)

4.4.3.2 Results

The same hyperparameters as that for the planetary gearbox dataset are adopted over here for the bearing dataset. The proposed SN-AEs with all 48 hyperparameter combinations as well as the baseline AE are trained and tested separately. Five repeating trials are conducted for each case. The fault detection AUCs are shown in Fig. 4.19.

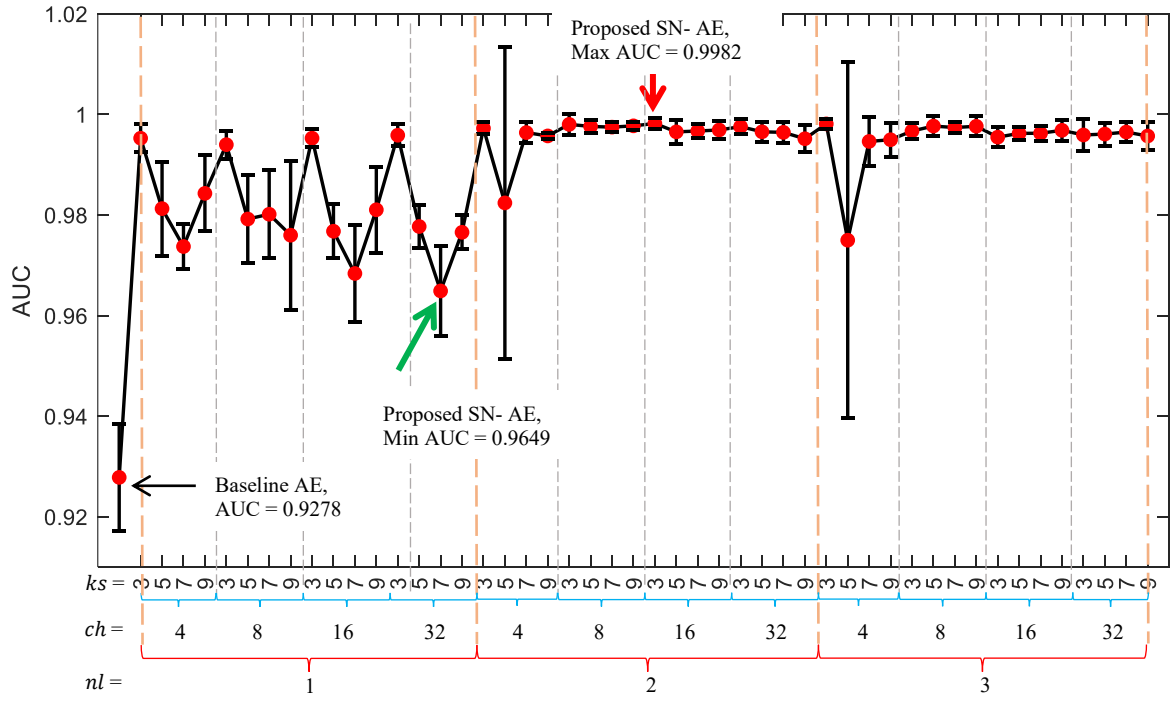


Fig. 4.19: Fault detection performance of the proposed SN-AE over the bearing dataset. nl – number of layers of the SN branch, ks – kernel size and ch – number of channels.

Still, similar trends as that of the planetary gearbox dataset and the fixed-shaft gearbox dataset are observed from Fig. 4.19 for the bearing dataset. The AUCs of the proposed SN-AE are all higher than that of the baseline AE, which is 0.9278$\pm$0.0106. The maximum AUC of the proposed SN-AE is 0.9982 ± 0.0011 obtained when ($nl = 2, ch = 16, ks = 3$). The minimum AUC is 0.9649$\pm$0.0089 returned by ($nl = 1, ch = 32, ks = 7$). Like the planetary gearbox dataset, the AUC becomes stable when $nl > 1$.

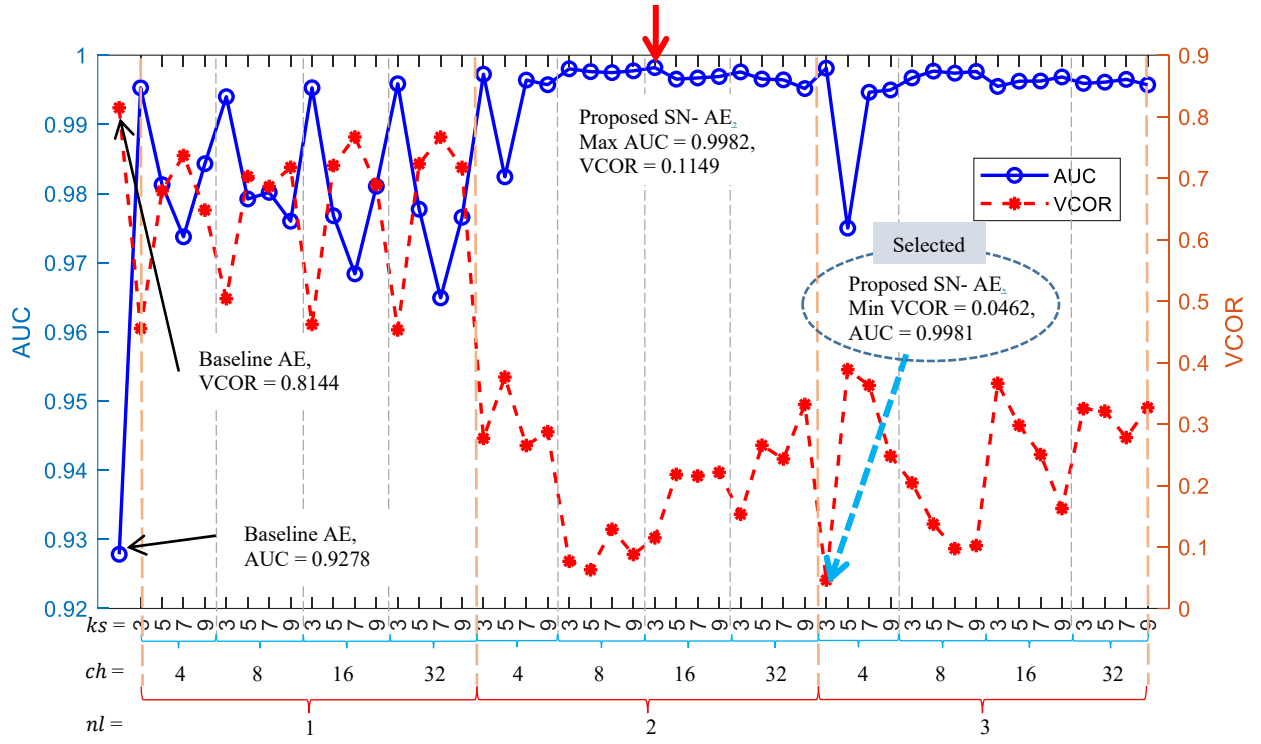
Like the planetary gearbox dataset and the fixed-shaft gearbox dataset, we also conducte hyperparameter selection for the bearing dataset. The results are shown in Fig. 4.20 and Table 4.6. Like the fixed-shaft gearbox dataset, both the proposed VCOR and the existing VMSE return quite high AUCs, which approach the highest AUC with the bearing dataset. The proposed VCOR performs slightly better.

Table 4.6: Hyperparameter selection for the SN branch with the bearing dataset.

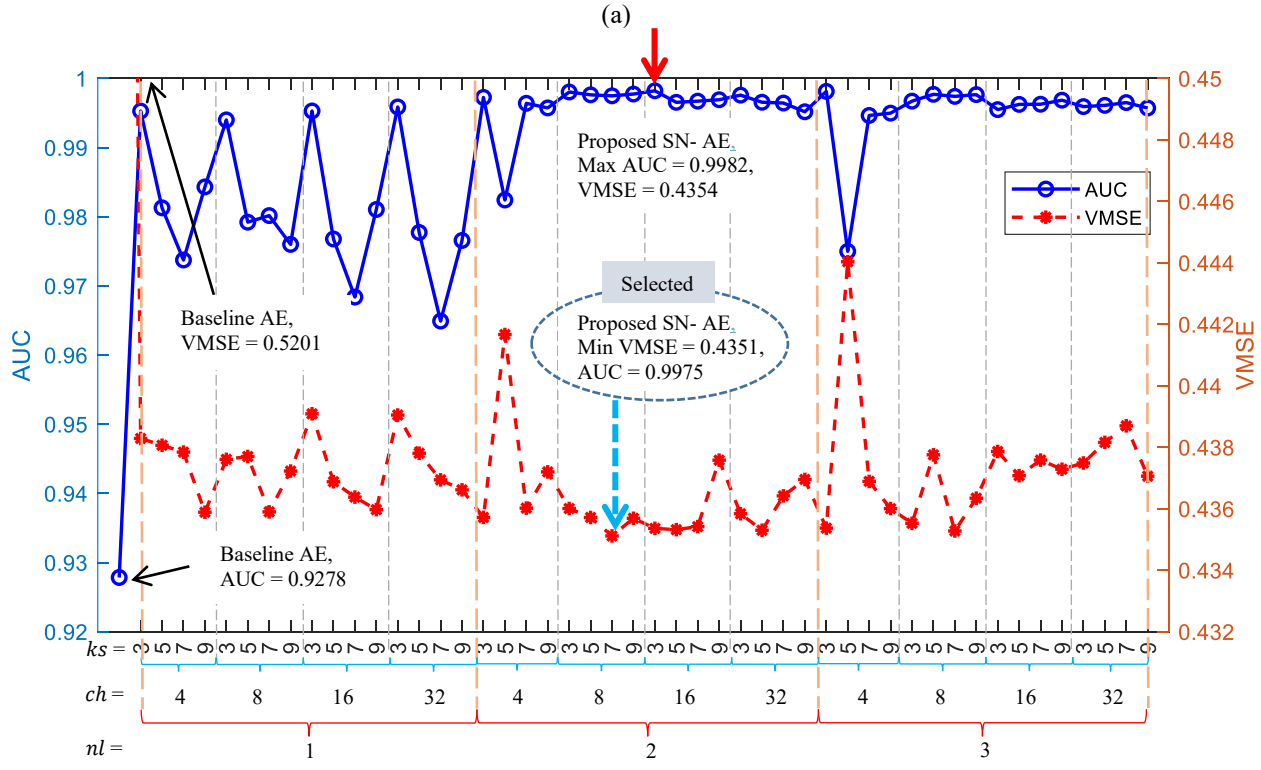
Selection method	AUC	VCOR	VMSE	Hyperparameters
Highest AUC with the test set	**0.9982**	0.1149	0.4354	(2, 16, 3)
Existing VMSE	0.9975	0.1288	**0.4351**	(2, 8, 7)
Proposed VCOR	0.9981	**0.0462**	0.4354	(3, 4, 3)

Notes: 1. The hyperparameters are in the form of (nl, ch, ks). 2. AUC of the baseline AE is 0.9278.

The learned SN function and the health indicator versus speed for the bearing dataset are shown in Fig. 4.21 and Fig. 4.22, respectively. Still, the learned SN values follows a declining pattern with the speed. The increasing trend in the health indicator (Fig. 4.22(a)) is removed with the proposed SN-AE (Fig. 4.22(b)).



(a)



(b)

Fig. 4.20: Hyperparameter selection for the SN branch of the proposed SN-AE over the fixed-shaft gearbox dataset: (a) Proposed VCOR and (b) Existing VMSE.

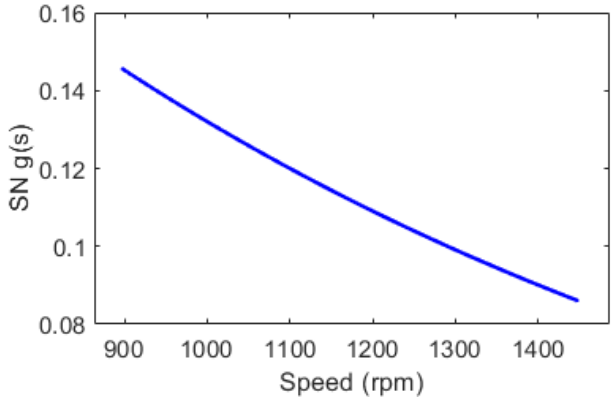


Fig. 4.21: Learned SN function over the bearing dataset with selected hyperparameters of $(nl = 1, ch = 4, ks = 3)$.

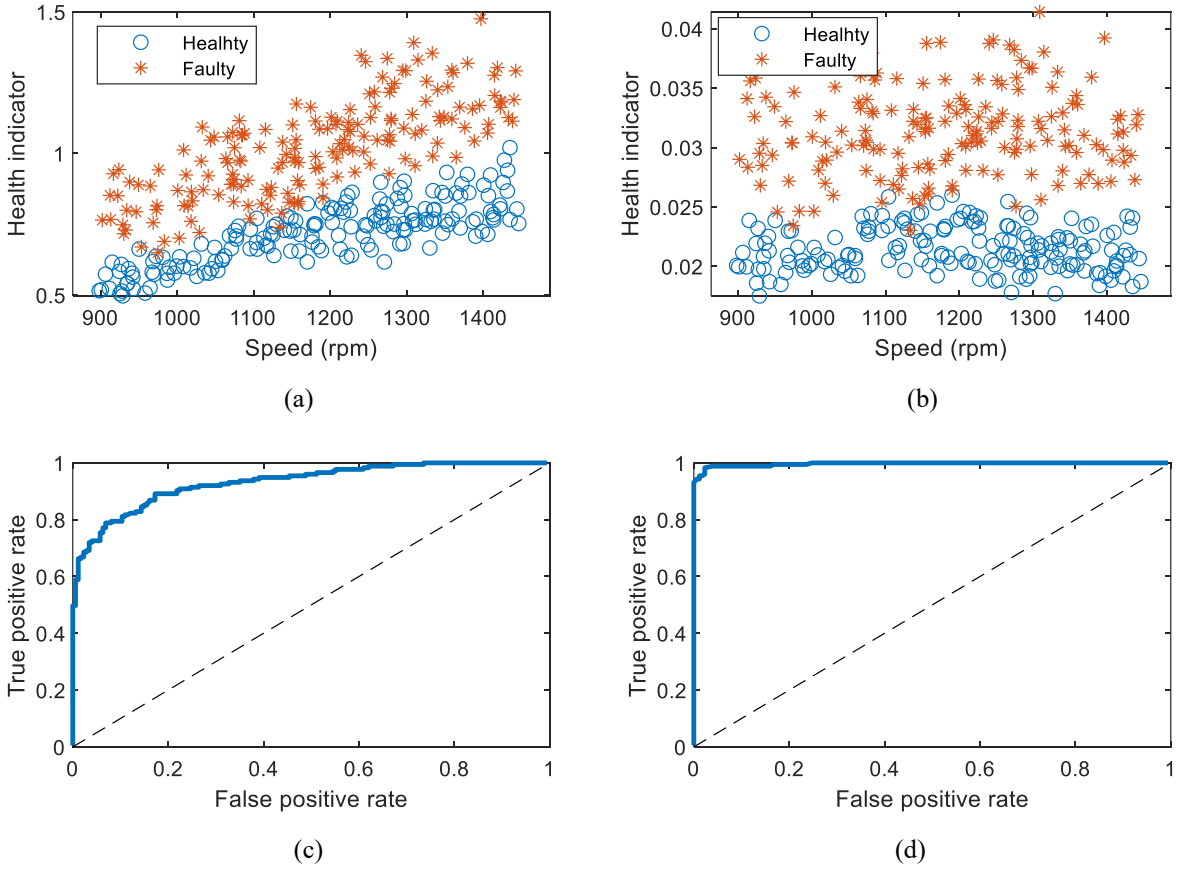


Fig. 4.22: Health indicator versus speed over the bearing dataset: (a) Baseline AE, (b) Proposed SN-AE with selected hyperparameters of $(nl = 3, ch = 4, ks = 3)$, (c) Receiver operator characteristic for (a), AUC = 0.9278 and (d) Receiver operator characteristic for (b), AUC = 0.9981.

4.4.4 Comparisons

The following comparisons are made to show the superiority of the proposed SN-AE over existing methods for rotating machinery fault detection under varying speed conditions.

First, the SN-AE is compared with reported signal processing methods [52], [92] on speed effects removal. Following [52], [92], the vibration signal is divided by its peak envelop and then multiplied by the average envelope to remove the AM effects induced by speed variation (AM Removal). The normalized vibration signals are then processed by the baseline models for fault detection.

Second, the whole package of the proposed SN is applied to four variants of the AE, which are among state-of-the-art methods for fault detection. Details are given below.

- Sparse AE [97], [95]: The model structure is identical to the baseline AE. An L1 regularization with a penalty coefficient of 0.0001 is applied to the activations of the bottleneck layer to inject sparsity to the model.
- Denoising AE [96]: The model structure is identical to the baseline AE. The white noise of 10 dB is added to the input of the model to enhance the robustness of the AE.
- AE-LSTM [85], [147]: The AE-LSTM contains two fully connected (FC) layers and two LSTM layers as follows: Input – FC (16) – LSTM (32) – LSTM (32) – FC (16) – Output. The numbers within the brackets are the number of neurons.
- AE-CNN [148], [149]: The AE-CNN contains 3 convolutional (Conv) layers and 2 pooling layers as follows: Input – Conv ($ch = 32, ks = 3$) – Max pooling (1/2) – Conv ($ch = 16, ks = 3$) – Up-pooling (2/1) – Conv($ch = 32, ks = 3$) – Output. The max pooling

layer down-samples data to halve its length. The up-pooling layer up-samples data to double its length.

Hyperparameters for training these models are the same as the baseline AE. The AUC is still taken as the fault detection performance measure. The average and standard derivation (Std) of AUCs of five repeating trials are given in Table 4.7. We can see that,

(1) With the reported AM removal method [52], [92] adopted, the resultant AUCs are improved across all models, but are smaller than that of the proposed SN for most cases except the case of AE-CNN over the bearing dataset. This means removing AM effects does help in improving fault detection performances of existing models, and our proposed SN outperforms existing AM removing methods.

(2) For the baseline AE and its variants, the proposed SN can significantly improve their fault detection performance, which validates the generalizability of the proposed SN.

(3) The proposed measure VCOR selects better models that return higher AUCs than that of the existing VMSE for most cases expect the case of AE-LSTM over the bearing dataset. This means the proposed hyperparameter selection measure VCOR generally outperforms existing counterparts.

Over here, only one set of hyperparameters for the AE and its variants are tried. It is obvious that better results may be obtained if hyperparameters of these baseline models are properly selected. This work was not conducted as comparisons among these baseline models are not the focus of this chapter. Instead, the focus is to show the effectiveness of the proposed SN branch for these models. It is reasonable to believe that if the performances of baseline models are improved, their

counterparts with the proposed SN branch applied will be improved accordingly. More discussions about this problem are given in Section 4.5.2.

Table 4.7: Comparison of the proposed SN-AE with exiting methods for fault detection: AUC.

Model	Hyperparameter selection	Planetary gearbox		Fixed-shaft gearbox		Bearing	
		Average	Std	Average	Std	Average	Std
Baseline AE	-	0.6863	0.0206	0.8652	0.0071	0.9270	0.0084
AM Removal + Baseline AE	-	0.7631	0.0251	0.9175	0.0217	0.9386	0.0418
Proposed SN-AE	VMSE	0.8635	0.0109	0.9166	0.0048	0.9975	0.0008
	Proposed VCOR	**0.9535**	**0.0044**	**0.9227**	**0.0107**	**0.9981**	**0.0008**
Spares AE	-	0.6984	0.0148	0.8474	0.0143	0.9274	0.0063
AM Removal + Spare AE	-	0.7772	0.0243	0.9005	0.0254	0.9576	0.0056
Sparse AE + proposed SN	VMSE	0.9137	0.0026	0.9100	0.0167	0.9971	0.0010
	Proposed VCOR	**0.9561**	**0.0012**	**0.9225**	**0.0289**	**0.9985**	**0.0009**
Denoising AE	-	0.6936	0.0154	0.8582	0.0189	0.9286	0.0063
AM Removal + Denoising AE	-	0.7771	0.0238	0.8469	0.0160	0.9314	0.0064
Denoising AE + proposed SN	VMSE	0.7809	0.0980	0.9154	0.0197	0.9795	0.0322
	Proposed VCOR	**0.8455**	**0.0955**	0.9212	0.0228	**0.9795**	**0.0322**
AE-LSTM	-	0.7232	0.0013	0.8359	0.0110	0.9466	0.0268
AM Removal + AE-LSTM	-	0.7809	0.0233	0.8627	0.0319	0.9568	0.0093
AE-LSTM + proposed SN	VMSE	0.7232	0.0013	0.8359	0.0110	**0.9946**	**0.0020**
	Proposed VCOR	**0.9147**	**0.0098**	**0.9087**	**0.0194**	0.9475	0.0052
AE-CNN	-	0.6064	0.0113	0.7859	0.0054	0.9532	0.0052
AM Removal + AE-CNN	-	0.7162	0.0247	0.8034	0.0023	**1.0000**	**0.0000**
AE-CNN + proposed SN	VMSE	0.6064	0.0113	0.7721	0.0140	0.9715	0.0038
	Proposed VCOR	**0.8170**	**0.0035**	**0.8149**	**0.0249**	0.9860	0.0025

Note: VMSE (VCOR) corresponds to results selected with such hyperparameter selection measure.

To this end, we have evaluated the fault detection performance of the proposed SN-AE and its variants over three experimental datasets. Results have shown that the proposed SN-AE is effective for rotating machinery fault detection under varying speed conditions, and the SN branch can be generalized to the variants of the AE.

We acknowledge the hyperparameters selected using the proposed VCOR are not perfect. As seen in all case studies, a smaller VCOR generally returns a higher fault detection AUC, but not strictly. Possible reasons are as follows.

First, the proposed SN-AE only removes the AM effects induced by speed variation in vibration signals. The VCOR measures how much such effects are removed. However, the vibration signals are not only affected by speed variation, but also load variation and environment noise and more [8]. Effects of these factors sometimes are dependent. Removing the effects of speed variation may exaggerate effects of others. As such, when the speed effects are moved the most, i.e., the VCOR is the smallest, the fault detection AUC is not guaranteed to be the highest.

Second, the hyperparameter selection is conducted over the validation set. Specifically, the optimal hyperparameters (or also the optimal model) are selected when the VCOR over the validation set is the smallest. The selected model is applied to the test set to conduct fault detection. An underlying assumption herein is that the selected SN-AE model which has the best speed variation effects removing ability over the validation set will perform well over the test set too. This is generally true but not strictly. Because the validation set only contains healthy data. This means the VCOR selects a model removing most speed variation effects with healthy data. However, the test set contains both healthy and faulty data. When the selected model is applied to the test set, the speed variation effects can be well removed with the healthy data but not guaranteed with the faulty data. As the faulty data features different patterns from healthy data. As such, a smallest VCOR over the validation set is not guaranteed to achieve the best speed variation removing performance over the whole test set, and thus is not guaranteed to yield the highest fault detection AUC over the test set.

However, the proposed VCOR generally returns models that foster just next to best fault detection AUCs and performs more stable than the existing VMSE. Therefore, we believe that the proposed VCOR for hyperparameter selection for the SN branch is effective for fault detection.

4.5 Discussion

This section provides discussions on five remaining concerns regarding the proposed SN-AE, including why the proposed SN-AE works, the impacts of hyperparameters of baseline models, training time requirements, structure for the SN branch, and limitations of the proposed SN-AE.

4.5.1 Why proposed SN-AE works

The proposed SN-AE works for fault detection because it normalizes the amplitude of raw vibration data to remove the speed induced AM effects. This is like the reported works of [52], [92] wherein the vibration amplitude is also normalized with a certain normalization function to remove the effects of speed variation to facilitate the fault detection and fault severity assessment. Our work differs from the reported works of [52], [92] on the designing of the normalization function. In [52], [92], the normalization function is manually designed using signal processing techniques, while in our proposed SN-AE, it is automatically learned. We believe that automatic learning returns better normalization functions than manual design, thanks to its performance guided nature. Accordingly, better fault detection performance has achieved as shown in Table 4.7.

The proposed SN-AE only removes the AM effects but does not alter the FM effects yet. Fig. 4.23 shows the speed, waveform, and spectrogram of a vibration signal of the planetary gearbox, and its normalized counterparts. Detailed observations are given below.

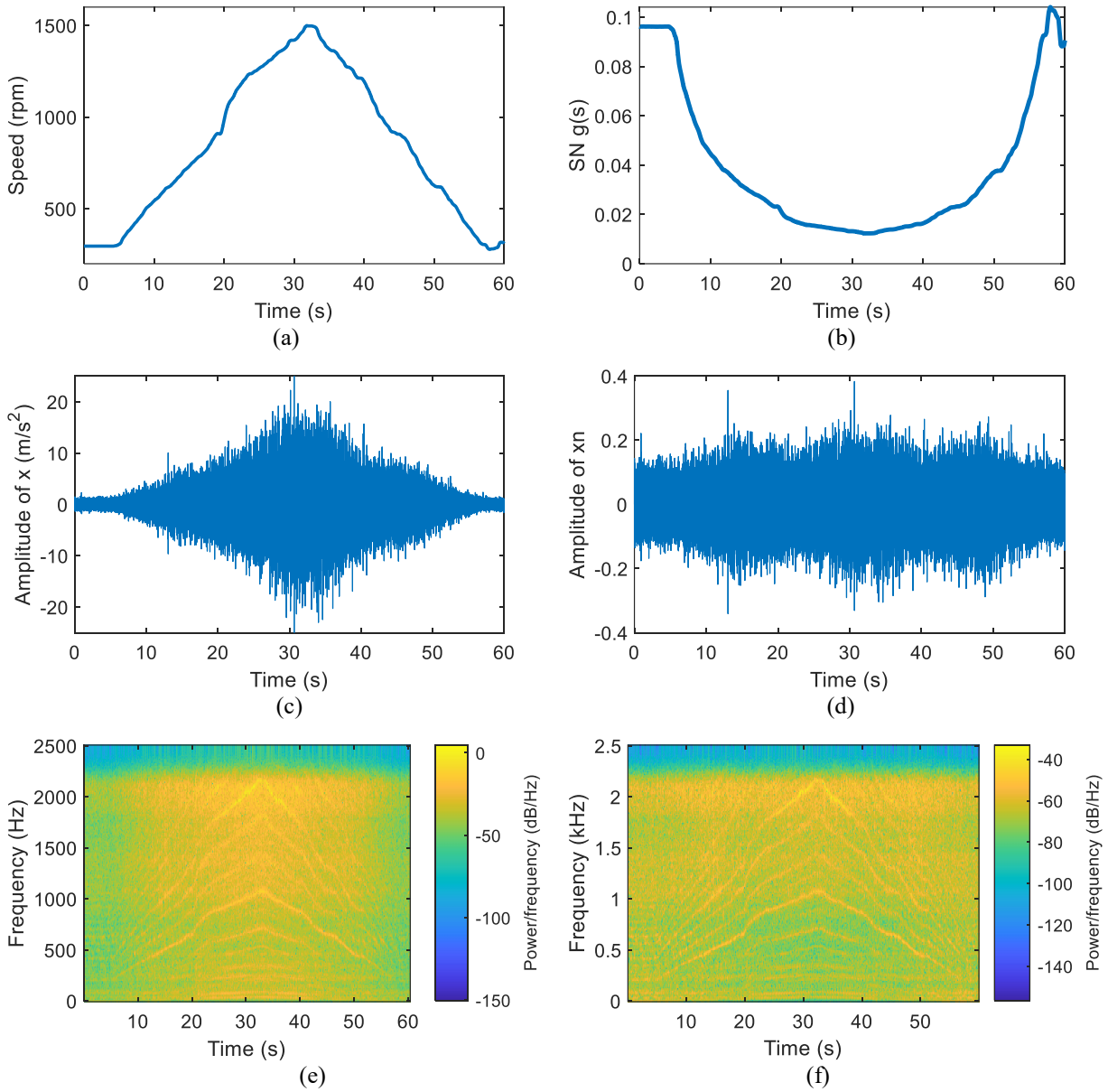


Fig. 4.23: Normalized vibration and its spectrogram of a piece of data in the planetary gearbox dataset: (a) Speed $\boldsymbol{s}$, (b) Normalization function $g(\boldsymbol{s})$, (c) Raw vibration $\boldsymbol{x}$, (d) Normalized vibration $\boldsymbol{x}_n = \boldsymbol{x}g(\boldsymbol{s})$, (e) Spectrogram for (c) and (f) Spectrogram for (d).

The normalization function $g(s)$ in Fig. 4.23 (b) is obtained using linear interpolation with the learned SN function shown in Fig. 4.11. Specifically, Fig. 4.11 shows the learned SN function $g(s)$, which is with respect to speed s. Fig. 4.23 (a) shows a speed signal $s(t)$ which is with respect

to time t. The SN function showing in Fig. 4.23 (b) is obtained by plugging $s(t)$ into $g(s)$, yielding the $g(s(t))$. Since the $s(t)$ and $g(s)$ do not have explicit formulas, it is not feasible to conduct the plugging analytically. Instead, the plugging is implemented through linear interpolation in Fig. 4.11 to finally get the $g(s(t))$ which is shown in Fig. 4.23 (b).

The normalized vibration in Fig. 4.23(d) is the element-wise product of raw vibration in Fig. 4.23(c) and the SN function in Fig. 4.23(b). We can see the amplitude variations in the normalized vibration are pretty much removed as compared to the raw vibration. Still remaining amplitude fluctuations may be caused by load variation and other possible factors that affect the amplitude of vibration signals [8]. However, the spectrograms of the raw vibration and the normalized vibration are similar as shown in Fig. 4.23(e) and (f). Therefore, we say the SN-AE only removes the AM effects but does not remove the FM effects.

To foster a complete solution to rotating machinery fault detection under varying speed conditions, we may use the matured order tracking [93] to resample the vibration to remove the FM effects first, and then apply the proposed SN-AE to the resampled signals to remove the AM effects. Results are given in Table 4.8. The resampling rate of order tracking is 720 pulses per revaluation. The AUCs are those selected with the proposed VCOR when applicable. We can see that, removing FM effects using order tracking does help in improving the fault detection performance with the baseline AE (e.g., from 0.6863 to 0.8713 for the planetary gearbox dataset), but the contribution is less than that of removing AM effects using the proposed SN-AE (e.g., from 0.6863 to 0.9535 for the planetary gearbox dataset). Removing both AM and FM effects does not bring clear benefits compared to removing the AM effects only. This may be because that in the present fault detection task, the employed heath indicator (i.e., RMS of the reconstruction residual) is more

sensitive to the amplitude of vibrations while less sensitive to the frequency changes. We thus empirically conclude that the AM effects outweigh the FM effects in the context of fault detection. While implementing fault detection tasks, we may only need to consider the AM effects.

Table 4.8: Fault detection results over the planetary gearbox dataset with order tracking applied (AUC).

Model	Planetary gearbox	Fixed-shaft gearbox	Bearing
Baseline AE*	0.6863 ± 0.0206	0.8652 ± 0.0071	0.9270 ± 0.0084
Proposed SN-AE*	0.9535 ± 0.0044	0.9227 ± 0.0107	0.9981 ± 0.0008
Order tracking + baseline AE	0.8713 ± 0.0214	0.8785 ± 0.0177	0.9328 ± 0.0106
Order tracking + proposed SN-AE	0.9476 ± 0.0180	0.9302 ± 0.0310	0.9438 ± 0.0076

* Raw signals are used. Order tracking is not applied. Results are identical to case studies.

4.5.2 Impacts of hyperparameters of baseline models

In case studies, the baseline AE uses a fixed group of hyperparameters. This may raise a doubt that the results are ad-hoc. To show that the proposed SN-AE is robust to hyperparameters of the baseline AE, we evaluate its performances with different numbers of neurons for the AE as follows.

- Setting 1: Input – 64 – 32 – 64 – Output.
- Setting 2: Input – 256 – 128 – 258 –Output.

Recall the default setting used in the case studies is of Input – 128 – 64 – 128 – Output. The number of neurons for the input and output is either 256 or 250 depending on the input data length. Results with these settings are shown in Fig. 4.24. For demonstration and simplicity purposes, only results of the planetary gearbox dataset are shown. We can see that the number of neurons affects the AUCs of the baseline AE and the proposed SN-AE simultaneously. The AUCs of the proposed SN-AE are improved from the baseline AE regardless of the number of neurons of the AE branch.

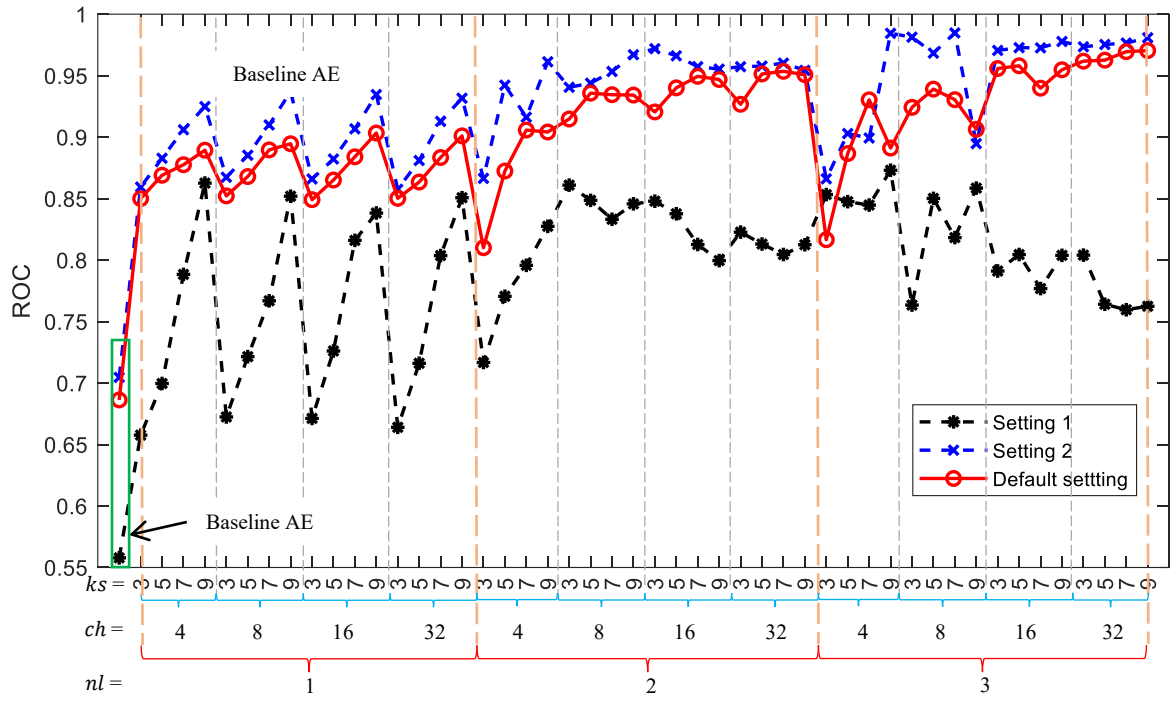


Fig. 4.24: Fault detection performance of the proposed SN-AE with different hyperparameters for AE branch over the planetary gearbox dataset.

4.5.3 Training time requirements

More training time is required for the proposed SN-AE because it has a larger model scale. Table 4.9 lists the training time of the SN-AE over different datasets. We can see that the proposed SN-AE requires at least 30% more training time, and up to three times of that for the baseline AE. The selected model also consumes significantly more training time. However, considering that the training process can be completed offline, the training time is not a critical problem if one has sufficient computation power. In real applications, we care more about the test time. For case studies in this chapter, the test time follows the same trend as the training time. The proposed SN-

AE consumes as much as 0.2 s, compared to about 0.05 s out of the baseline AE. Such a short time would not be a big problem in real applications.

Table 4.9: Training time requirements for the proposed SN-AE.

Dataset	Baseline AE (s)	Proposed SN-AE (s)	
		Range	Selected
Planetary gearbox	35.58	48.71~100.35	76.70
Fixed-shaft gearbox	8.78	11.73~30.11	15.79
Bearing	15.92	20.68~47.25	33.40

Note: 1. Hyperparameters for the AE branch are the default of Input – 128 – 64 – 128 – Output. 2. "Selected" means the model selected using the proposed VCOR.

4.5.4 Structure of SN branch

In this chapter, the SN branch uses a convolutional neural network, which is generally complex, and thus has greatly increased the computation load as shown in Table 4.9. Besides, the learned SN function (Fig. 4.11, Fig. 4.15 and Fig. 4.21) are relatively simple and smooth. As such, there is a need to explore simpler structures for the SN branch.

The results in case studies (Fig. 4.9, Fig. 4.13 and Fig. 4.19) have shown that even an SN branch consisting of a single convolutional layer with the least number of channels and smallest kernel size can return significantly better fault detection performance than the baseline model. This means that it is not a must to use complex structures for the SN branch to obtain superior fault detection performances. Therefore, it is feasible and promising to explore simpler structures for the SN branch.

4.5.5 Limitations

One obvious limitation is that the speed signals are mandatory for the proposed SN-AE. For scenarios wherein the speed signals are not available, the SN-AE is not applicable.

In this chapter, the proposed SN has been applied to the AE and its variants. The proposed SN is supposed to be applicable for any deep learning models that follow a data reconstruction-based fault detection procedure as shown in Fig. 4.1. However, the SN only works when we follow a reconstruction-based procedure. As shown in Fig. 4.3, a same SN function learned by the SN branch is multiplied by the input and divided by the output of the baseline model. This requires that the time stamps of the input and the output of the baseline model should be identical. This requirement is satisfied when the baseline model is implementing a reconstruction task in the context of fault detection.

A remaining problem is that how the SN branch learns the expected SN function lacks mathematical support. Like most of the current deep learning models, this is a black box that needs further exploration to look inside to unfold the working mechanism. With the advances of the theory of deep learning, this problem might be solved accordingly.

The proposed VCOR is for the hyperparameter selection of the SN branch only. It essentially measures the degree of how much the effects of speed variation is removed. It is not appropriate for the selection of other hyperparameters like the number of neurons of the AE and the learning rate. As these hyperparameters are dependent from removing effects of speed variation but regarding of the reconstruction performance of the model. Besides, we only selected hyperparameters for the SN branch. We did not conduct hyperparameter selection for the AE branch, not mention for the whole SN-AE model. If we could figure out a way to select hyperparameters for the SN branch and the AE branch together, even better fault detection performances can be expected as an overall optimum of hyperparameters of both branches could be achieved.

4.6 Summary and conclusion

To improve the fault detection performance of the baseline AE for rotating machinery under varying speed conditions, we proposed an improved deep learning model named SN-AE. It uses an SN branch to normalize the vibration to remove the AM effects induced by speed variation before the vibration being processed by the AE. Effectiveness of proposed SN-AE is validated over three datasets. Major conclusions are drawn below:

(1) The proposed SN-AE significantly improves the fault detection performance of the baseline AE.

(2) The proposed SN-AE works because it removes AM effects induced by speed variation.

(3) It is promising to use correlation coefficient between the reconstruction error and the speed as the performance measure to select hyperparameters for the SN branch of the proposed SN-AE.

(4) Removing AM effects contributes more to better fault detection performances than removing FM effects under varying speed conditions.

In this chapter, we only used the convolutional neural network for the SN branch. In the future, we will explore simpler structures for the SN branch to ease the computation load.

5. Speed adaptive gate for improving fault classification accuracy of deep learning models for rotating machinery under varying speed conditions

This chapter focuses on fault classification of rotating machinery that operated under varying speed conditions. It is the Topic #3 as defined in Section 1.3. An auxiliary branch named the speed adaptive gate is proposed for existing deep learning models to improve their fault classification accuracies under varying speed conditions. The speed extracted in Chapter 3 can be used here as the input to the proposed auxiliary branch. The purpose of fault classification in this chapter is to find the type and severity of an occurred fault that has been detected in Chapter 4. Materials of this chapter have been documented in a submitted journal paper [164] and a published conference paper [165].

5.1 Introduction

Rotating machines like motors, generators, gearboxes, and wind turbines are often subjected to faults in industrial applications. When a fault has occurred, a natural concern is to identify the root cause and the severity of the fault. This is often a fault classification problem in the context of fault diagnosis. Fault classification is usually understood as a supervised learning problem. Labeled data including vibration data and corresponding health states are needed. Deep learning models like the deep feedforward neural network (FNN), convolutional neural network (CNN), residual network

(ResNet) and recurrent neural network (RNN), and their variants are widely employed for the fault classification of various rotating machines, like bearings, gears, rotors, motors, CNC machines, wind turbines, compressors and more [38], [99], [104].

As reviewed in Section 1.2.3, even reported works [114] have shown demonstrated performances with rotating machinery fault classification under varying speed conditions, they still suffer from the following drawbacks: (1) Effects of speed variation on fault classification performances of deep learning models are not unfolded and (2) how to address the effects of speed variation is not specifically investigated.

In this chapter, we will address the above-mentioned drawbacks and finally improve the fault classification accuracy of exiting deep learning models for rotating machinery under varying speed conditions. To achieve the goal, we firstly investigate the effects of speed variation on the fault classification performances of deep learning models. Existing understanding of effects of speed variation is from the perspective of signal processing. Such understanding is hard to be perceived by deep learning models. We propose to analyze the effects of speed variation from the perspective of deep learning. We find the fault information in vibration signals is imbalanced with speed, i.e., higher speeds excite more fault information. We further find that, due to the imbalance, the fault classification accuracy of deep learning models is not constant but increases with speed. To address the speed variation induced fault information imbalance, we then propose an auxiliary branch named speed adaptive gate (SAG) for existing deep learning models. The idea behind is to use an SAG to balance the fault information usage. The SAG takes the speed signal as the input. It controls the information flow of deep learning models in terms of speed, such that the effects of speed variation are mitigated. Effectiveness of the proposed SAG is validated with two baseline

models, i.e., a CNN and a ResNet [114], [123], over two experimental datasets, i.e., a planetary gearbox dataset and a fixed-shaft gearbox data. The CNN and the ResNet are selected because they are strong baselines for time series classification [123], and are among state-of-the-art models for machinery fault classification [114]. Major contributions of this chapter are summarized below.

(1) Investigated the effects of speed variation on vibration signals from the perspective of deep learning and found that speed variation would lead to fault information imbalance in vibration signals.
(2) Proposed a speed adaptive gate (SAG) for existing deep learning models to address the speed induced fault information imbalance.

The rest of this chapter is organized as follows. The baseline models are introduced in Section 5.2. Section 5.3 analyzes effects of speed variation and presents the proposed SAG. Case studies of the proposed SAG for CNNs and ResNets are conducted in Section 5.4 and Section 5.5, respectively. Discussions are made in Section 5.6. Conclusions are drawn in Section 5.7.

5.2 Baseline models

This section briefly introduces the baseline CNN and the baseline ResNet. Fundamentals of CNNs and ResNets have been provided in Section 2.2.2.

(1) Convolutional neural network (CNN)

In this chapter, for the rotating machinery fault classification task, the model shown in Fig. 5.1 will be a baseline CNN model. The architecture and hyperparameters of the baseline CNN are

adopted from state-of-the-art models as seen in [114], [123]. The structure of the baseline CNN is as follows:

- 5 convolutional layers with numbers of channels of (32, 32, 32, 64 and 64) and corresponding kernel sizes of (5, 3, 3, 3 and 3).

A max pooling layer is adopted before the second convolutional layer to halve the data length. A global average pooling (GAP) layer is adopted before the fully connected layer to prevent possible overfitting.

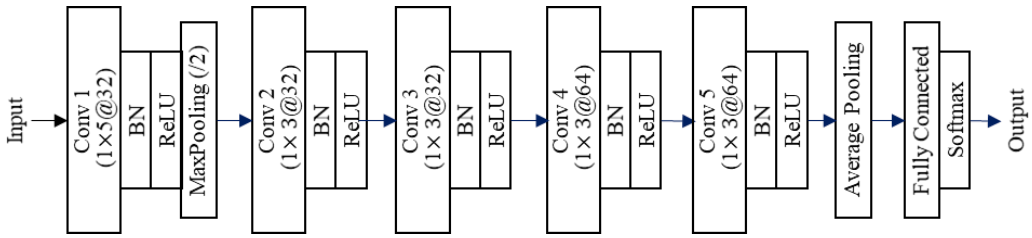


Fig. 5.1: Baseline CNN model [114], [123].

(2) Residual Network

ResNets are a special kind of CNNs, which feature an additional shortcut connection [124]. Fundamentals of ResNets have been given in Section 2.2.2. ResNets were initially introduced to address the performance degradation problem in the training of very deep networks for image classification [124], but later show competitive performances with time series classification [123], including the fault classification in PHM [166]. In this chapter, a ResNet adopted from [114], [123] will serve as another baseline model for rotating machinery fault classification. The architecture of the baseline ResNet is exhibited in Fig. 5.2. It has a same primary architecture as the baseline CNN but has two additional shortcut connections.

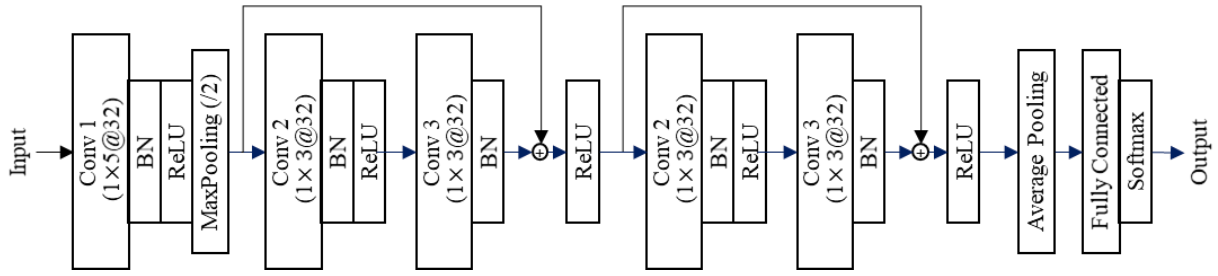


Fig. 5.2: Baseline ResNet model [114], [123].

As described in Section 5.1, the baseline CNN and ResNet do not consider the effects of speed variation when applied for fault classification. Next, we will attempt to address this problem and ultimately improve their fault classification performances under varying speed conditions.

5.3 Proposed speed adaptive gate (SAG)

This section firstly analyzes the effects of speed variation, and then presents details of the proposed SAG, which is designed to address such effects.

5.3.1 Effects of speed variation

From the perspective of signal processing, effects of speed variation are usually interpreted as additional amplitude modulation (AM) and frequency modulation (FM) in collected vibration signals. However, when deep learning models process vibration signals, the models do not "understand" what the AM and the FM are. They only perceive signals as discrete values. To make the effects of speed variation "understandable", in this chapter, we will attempt to analyze the effects of speed variation from the perspective of deep learning.

Taking a rotating machine with a localized fault as an example, fault related information contained in vibration is often revealed as impulses [8]. The impulses are excited once or more in every single revolution [18]. In an identical sampling period, the amount of fault related information or impulses, is proportional to revolutions, thus proportional to rotating speed. An example of collected acceleration signals from a planetary gearbox which suffers from a tooth missing fault in the planet gear is given in Fig. 5.3. We can observe that within the sampling period of 4 s, the acceleration signal under 600 rpm contains more impulses than that of the 300 rpm, say, about 10 impulses versus 5 impulses. Besides, the amplitude of impulses under 600 rpm is also larger.

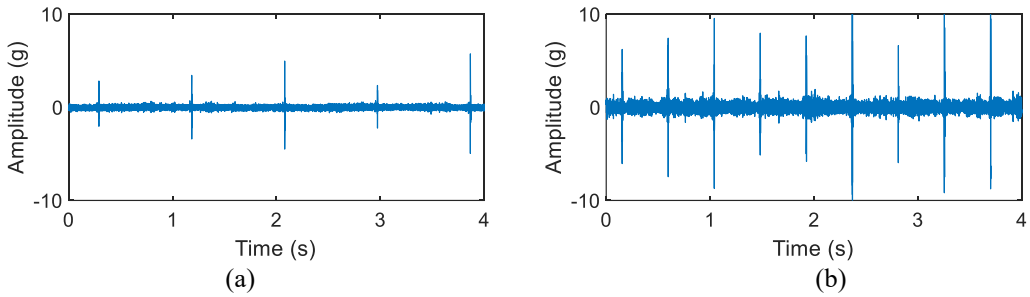


Fig. 5.3: Experimental acceleration signals of a planetary gearbox with a tooth missing fault in a planet gear: (a) 300 rpm and (b) 600 rpm.

Based on above reasoning and observations, we suggest a term named “speed induced fault information imbalance” to describe the effects of speed variation from the perspective of deep learning. The term is defined as that the amount of fault information in vibration signals varies with the speed. Specifically, higher speeds excite more fault information in vibration signals in an identical sampling period, which future includes two sides:

- Amplitude side: Higher speeds increase machine vibration amplitude (as well as amplitude of fault signatures), making defects more evident.

- Frequency side: Higher speeds increase the occurrence frequency of fault signatures, which increases the number of fault signatures.

Both sides can be illustrated by a global level measure like the RMS. However, the RMS is not used to measure the degree of the information imbalance in this chapter. As the RMS is not only affected by the fault information imbalance, i.e., speed, but also fault types. Different fault types may result in different RMS values with an identical speed. Therefore, in this chapter, we suggest simply using the speed to measure the degree of fault information imbalance.

The definition of fault information imbalance is inspired by a similar term in deep learning, i.e., the class imbalance [167], which indicates the inequality of data amount of classes. Given the speed induced fault information imbalance, our hypothesis here is that deep learning models will bias to conditions with more fault information like the class imbalance problem. Correspondingly, the fault classification accuracy of higher speeds would be larger than that of lower speeds, that is, accuracy would increase with speed. In this chapter, we do not have a clear boundary for low or high speed conditions. We use lower or higher to illustrate relative speed levels within a dataset.

5.3.2 Proposed SAG

Here introduces the proposed SAG, and how to apply SAG to baseline models, i.e., SAGed models.

5.3.2.1 Building block

Given that the fault information is imbalanced due to speed variation, the motivation is to address such imbalance problem. The idea is to balance the information usage in term of speed. Specifically, if deep learning models process samples unequally, and preferably, speed adaptively and with more emphasis on lower-speed conditions, the effects of speed induced fault information

imbalance might be mitigated, and the fault classification accuracy of deep learning models would be improved. To implement this idea, in this chapter, we propose an auxiliary branch SAG to multiply existing deep learning models. The SAG is supposed to adaptively regulate the information flow in deep learning models in terms of speed to balance the information usage.

The building block of the proposed SAG for CNNs (SAG-CNNs) is shown in Fig. 5.4. The SAG, $G(\boldsymbol{s})$, and routes to SAG, are in red color. Contents in black color are a copy of the basic building block for CNNs as shown in Fig. 2.4. The output of the ReLU layer $\boldsymbol{xr}$ is multiplied (shown as "×") by the SAG $G(\boldsymbol{s})$. The output of the building block for SAG-CNNs then becomes,

$$\boldsymbol{xr} = ReLU(\boldsymbol{G}(\boldsymbol{s}) \times \boldsymbol{xr}). \tag{5.1}$$

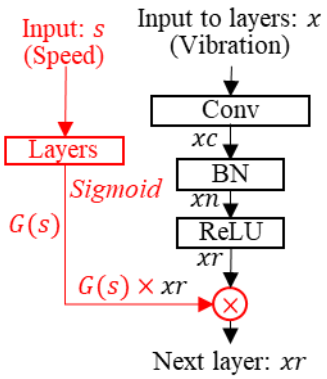


Fig. 5.4: Proposed speed adaptive gate for CNNs: Building block.

The building block of the proposed SAG for ResNets (SAG-ResNets) is illustrated in Fig. 5.5, wherein SAGs are in red color, and the basic building block for ResNets (same as Fig. 2.5) is in black color. Different from SAG-CNNs which has only one SAG, we have two SAGs for ResNets, i.e., $GR(\boldsymbol{s})$ and $GI(\boldsymbol{s})$. Each multiplies a branch of the ResNet. The overall mapping of the building block for SAG-ResNets is,

$$H(\boldsymbol{x}) = GR(\boldsymbol{s}) \times F(\boldsymbol{x}) + GI(\boldsymbol{s}) \times \boldsymbol{x}. \tag{5.2}$$

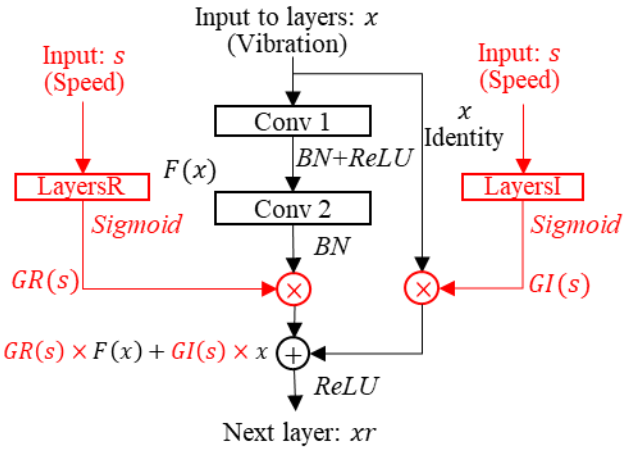


Fig. 5.5: Proposed speed adaptive gates for ResNets: Building block.

For either SAG-CNNs or SAG-ResNets, the SAGs are originated from the auxiliary inputted speed signal $\boldsymbol{s}$, and then learned through neural network layers. For SAG-ResNets, layers to learn the residual gate and identity gate are named LayerR and LayerI, respectively, to distinguish from each other. The type of layers could be convolutional layers, fully connected layers, or others. The multiplication operation between SAGs and baseline models could be elementwise, scalar, or others, depending on the type of layers of the SAG branch.

The SAGs are to regulate the information flow of baseline neural networks. The values of SAGs control how many portions of fault information to be used. They are expected to adaptively change with speed. As such, different portions of fault information would be used under different speed conditions. It is worth to mention that using SAGs to regulate information flow is inspirited by the gating mechanism employed in the Long Short-term Memory (LSTM) [126] and the Highway

Network (HN) [168]. In these works, the gates are learned from the input $\boldsymbol{x}$ itself, while our proposed SAGs are explicitly learnt from speed $\boldsymbol{s}$, thus being speed adaptively.

The weights and biases of the SAG branches are trainable and will be automatically learned while training the whole deep learning models i.e., SAG-CNNs or SAG-ResNets. Values of SAGs are between 0 and 1 thanks to the sigmoid activation function. Hopefully, the learned values of SAGs will be larger when the speed is lower and smaller when the speed is higher. In this way, more emphasis is given to lower speeds to balance the speed induced fault information imbalance. The fault classification accuracy is supposed to be improved accordingly. These guesses will be empirically validated in case studies.

5.3.2.3 SAGed models

With the SAG building blocks as shown in Fig. 5.4 and Fig. 5.5, we can build SAG-CNNs and SAG-ResNets by stacking them as needed. Deep learning models with SAG added will be shorted as SAGed models. To configure SAGed models, we shall consider at least the following aspects:

- Structure of the SAG branch.
- Location to apply the SAG.

The structure of the SAG will be determined in reference to [126], [168] for the purpose of straightforward to implement in this chapter. That is, we are going to use same convolutional layers as the baseline models, but with less layers. We understand there may be better, at least simpler structures for the SAG. However, this is not the focus of this chapter. Our thought is that if the current structure which is straightforward, not specifically designed and easy-to-implement, does work, the effectiveness of our proposed SAGs would be solid. Readers can explore better

structures without doubt of possible ineffectiveness. The location of SAGs will be analyzed as it is a critical parameter to apply SAGs. Next, we will give two examples of applying SAG to baseline models, including the locations of the SAG.

(1) SAG-CNN

Given above-mentioned considerations, we suggest the structure of the baseline CNN with SAGs, i.e., SAG-CNN, as shown in Fig. 5.6. The baseline CNN (Same as Fig. 5.1) is in black color. The SAGs are in red color. Three tentative SAGs, G1, G2 and G3, are suggested for the baseline CNN to regulate its information flow at different locations. The G1 multiplies the baseline CNN at shallower depth, G2 is for middle depth and G3 is for deeper depth. The multiplication operation here is elementwise. We will evaluate the effects of SAG locations and the number of SAGs by applying either one or more of these three SAGs.

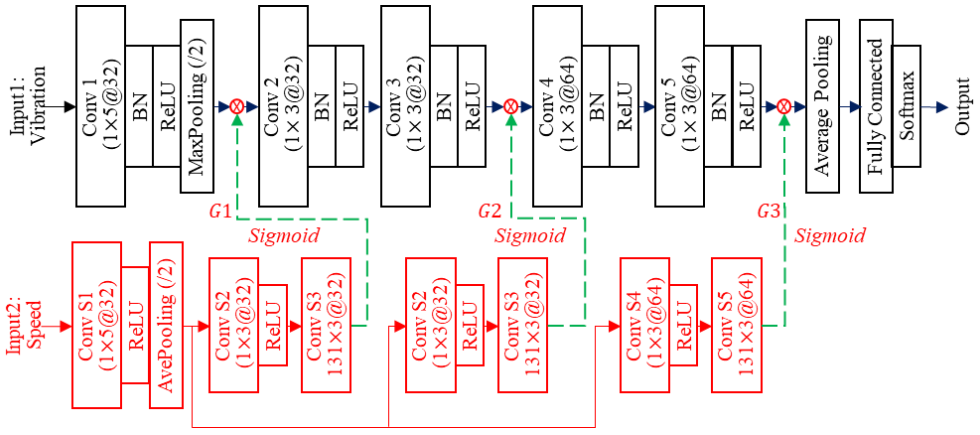


Fig. 5.6: SAGs for the baseline CNN: SAG-CNN.

In this chapter, networks to obtain SAGs have similar structures as that of the baseline CNN, but with following differences. First, the batch normalization (BN) layer is not adopted, as we want to reserve the variations of speed across batches. Second, the information flow of the SAGs is not

consecutive, e.g., the G2 is not directed to G3. As such, the diversity of SAGs is expected. Third, the average pooling instead of max pooling is used to mitigate small fluctuations in speed signals. The number of layers to obtain G1, G2 and G3 are the same to assure a same learning ability of each gate. The dimensionalities of G1, G2 and G3 can be different and are identical to their counterparts in the baseline CNN.

(2) SAG-ResNet

Following the same thoughts for building SAG-CNNs, we have the structure of SAG-ResNets as shown in Fig. 5.7. The baseline ResNet (Same as Fig. 5.2) is in black color. The SAGs are in red color. The SAG-ResNet possesses four tentative SAGs, i.e., GR1, GI1, GR2 and GI2. Among them, GR1 and GR2 multiply residual branches, and GI1 and GI2 multiply identity branches. GR1 and GI1 are applied at lower depth and GR2 and GI2 are for deeper depth. Performances of applying either one or more of these four SAGs will be checked in case studies.

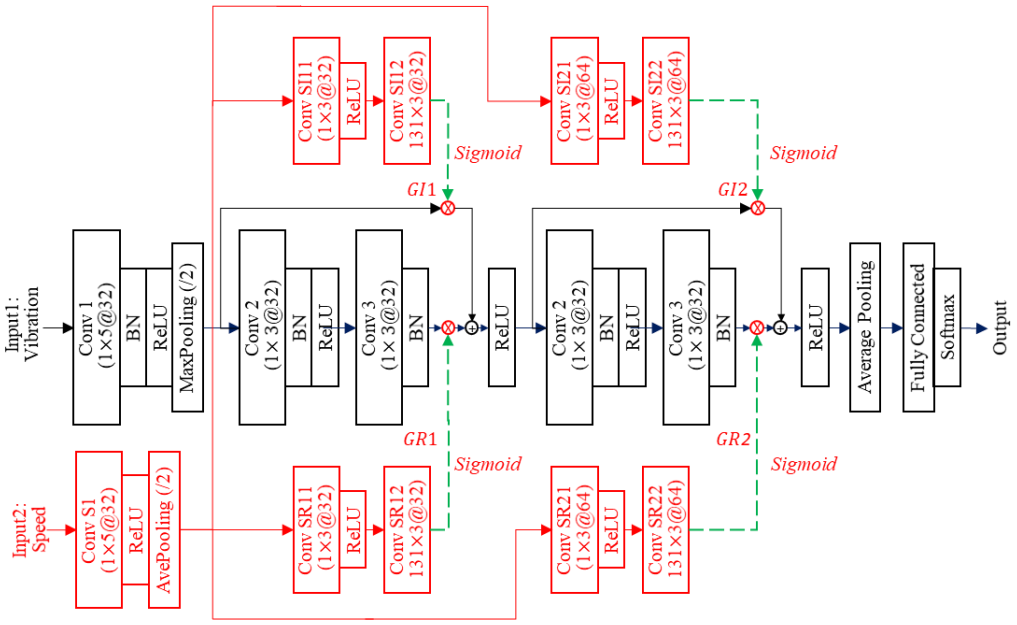


Fig. 5.7: SAGs for the baseline ResNet: SAG-ResNet.

The overall flowchart of using the proposed SAG-CNN or SAG-ResNet for rotating machinery fault classification is illustrated in Fig. 5.8. The vibration data and the speed data is firstly preprocessed and split into a training set and a test set. Each set contains both vibration and speed. The training set is used to training the model. The test set is to test the performance of the trained model. The fault classification results of the test set are to be analyzed to gain insights of the proposed model. Performances of the proposed SAG-CNN and the SAG-ResNet will be evaluated in Section 5.4 and Section 5.5, respectively.

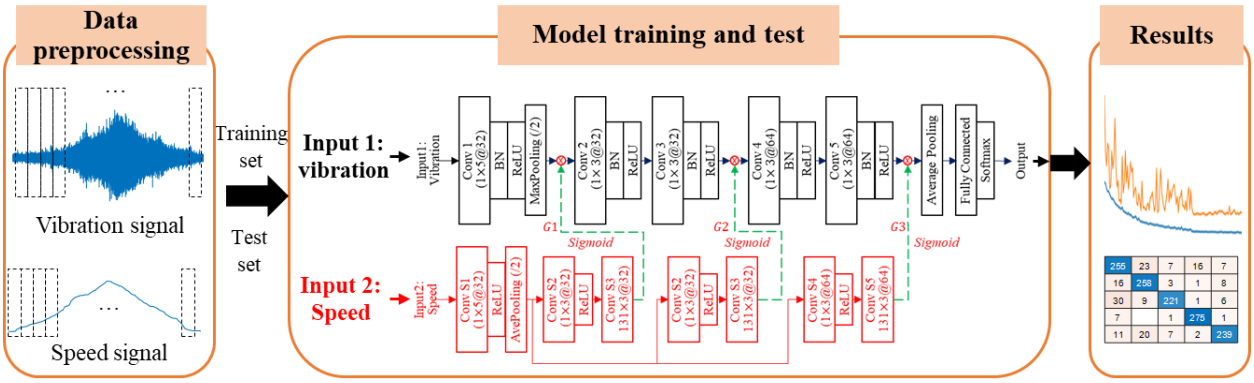


Fig. 5.8: Flowchart of using the proposed SAGed model for rotating machinery fault classification. The SAG-CNN is illustrated as an example.

5.4 Case studies with SAG-CNN

This section presents two case studies including a planetary gearbox dataset and a fixed-shaft gearbox dataset to validate the effectiveness of the proposed SAG for CNNs. Comparisons are made with a method for the class imbalance problem named cost-sensitive loss [169], and a method of utilizing speed for fault classification [170], to highlight the superiority of our proposed SAGs from multiple perspectives.

5.4.1 Case study 1: Planetary gearbox dataset

5.4.1.1 Dataset description

The planetary gearbox dataset was collected by the author and colleagues in 2021 at Tsinghua University, Beijing, China [162]. Detailed introduction has been provided in Section 4.4.1. Healthy sates of this dataset are summarized in Table 5.1. In this chapter, data with all health states will be used for fault classification. Still, only the vertical vibration will be employed.

Table 5.1: Health states of the planetary gearbox dataset.

Health sate ID	Health state	Numbering in Fig. 4.7
1	Healthy	-
2	Ring gear tooth missing	(a)
3	Sun gear chipped tooth	(b)
4	Sun gear tooth missing	(c)
5	Planetary gear tooth root crack	(d)
6	Planetary gear chipped tooth	(e)
7	Planetary gear tooth missing	(f)
8	Planetary bearing inner race crack	(g)
9	Planetary bearing inner race fault	(h)
10	Planetary bearing outer race crack	(i)
11	Planetary bearing outer race fault	(j)
12	Planetary bearing rolling element fault	(k)
13	Input shaft bearing inner race crack	(l)
14	Input shaft bearing inner race fault	(m)
15	Input shaft bearing outer race crack	(n)
16	Input shaft bearing outer race fault	(o)

Note: Bearing race fault (crack) means seeded crack width = 2 mm (0.4 mm).

For the planetary gearbox dataset, we have 80 measurements (5 repeating tests × 16 health states) in total. Speed profiles of each measurement are close but slightly different as the speed was manually controlled with the knob of the converter. All the measurements are preprocessed following similar works [110], [114] to fit in the CNN as follows.

(1) Low-pass filter and down-sample data including both acceleration and speed. For the planetary gearbox dataset, the data is down-sampled from 20 kHz to 5 kHz.

(2) Segment data into short samples with a length of 2000 data points (0.4 s).

(3) Delete outliers to balance the distribution across speed values. The outliers here refer to samples with rare speed values which are pretty much outside the range of 300 rpm – 1500 rpm.

(4) Randomly split the samples into a training set and a test set with a ratio of 8:2.

(5) Normalize the dataset. The vibration is normalized with the average and the standard derivation of the training set. The speed is divided by the maximum of the training set.

After preprocessing, the resultant dataset is summarized in Table 5.2. Out of the 10447 samples, 80% are in the training set and 20% are in the test set. The number of samples is evenly distributed across health states and speed.

Table 5.2: Summary of the planetary gearbox dataset.

Speed conditions	# of health states	# of samples	Sample length
Continuously varying (300 ~ 1500 rpm)	16	10,447	2,000 data points (0.4 s)

5.4.1.2 Model setting

For the baseline CNN and the proposed SAG-CNN, the hyperparameters are in reference to [114], [124] as follows. The learning rate follows a decaying scheme [26], i.e., initial learning rate = 0.001, divided by 10 every 50 epochs. The maximum epoch is 150. The batch size is 256. The optimization method is the Adam [26]. The models are programmed in Python with Keras and run in Google Colaboratory [140] with one GPU utilized.

5.4.1.3 Results

(a) Accuracy

The fault classification accuracy of the proposed SAG-CNN over the planetary gearbox dataset is shown in Fig. 5.9. The curves are the average accuracy of five repeating trials. The error bars show the standard derivations (std). The red solid curve represents the results of the proposed SAG-CNN wherein the gates are learned from speed (speed gated). The dash curve shows results of models with a same structure as SAG-CNN, but the Input2 is replaced with vibration (vibration gated). Reasons to have this curve are as follows. First, the SAG-CNN is more complex than the baseline CNN. A larger complexity often contributes to a better performance [26]. To tell whether the performance gain of the proposed SAG-CNN is contributed by the proposed SAG or the model complexity, we eliminate speed from the model but sustain the model complexity, that is, replace Input2 from speed with vibration. Second, using vibration to obtain gates somewhat reimplements how gates are learned in LSTM [126] and HN [168]. The dash curve thus provides a reference of how well these reported gates perform with fault classification of rotating machinery that operated under varying speed conditions.

In Fig. 5.9, NA means gates are Not Available, corresponding to results of the baseline CNN. The G1 means only G1 is applied, G123 ≡ G1+G2+G3 means three gates G1, G2 and G3 are applied, and so on. Fig. 5.9 shows that,

(1) With the SAG applied, the fault classification accuracy is improved compared to the baseline CNN. This means our proposed SAGs are helpful for improving the fault classification accuracy of the baseline CNN.

(2) The speed gated always has higher accuracies than that of the vibration gated. This means learning gates from speed works better than that learning from vibration as in [126], [168]. In the following we will focus on the speed gated only.

(3) For SAG-CNNs, the highest accuracy is 98.65% when G123 applied, followed by 98.63% (G1 applied) and 98.62% (G2 applied). These values are significantly higher than the classification accuracy of 95.16% out of the baseline CNN.

(4) With a single gate applied, G1 and G2 foster higher fault classification accuracies than that of G3. This means applying gates close to the output layer of the CNN is not favorable. Applying multiple gates sometimes results in relatively worse performances, especially when G3 is included. As such, applying a single SAG is preferred.

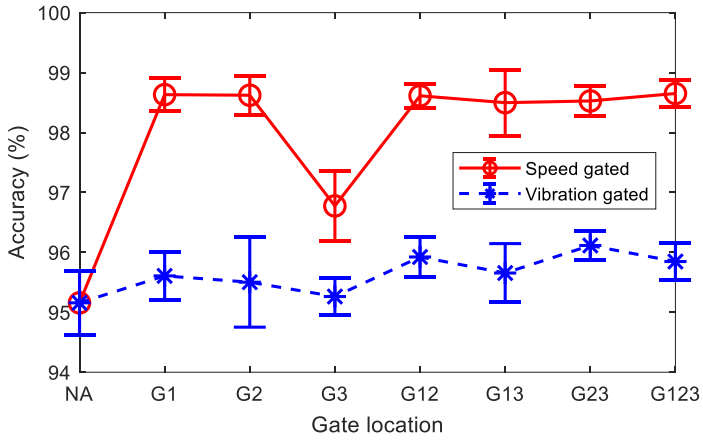


Fig. 5.9: Fault classification results of the proposed SAG-CNN over the planetary gearbox dataset. NA-Baseline model.

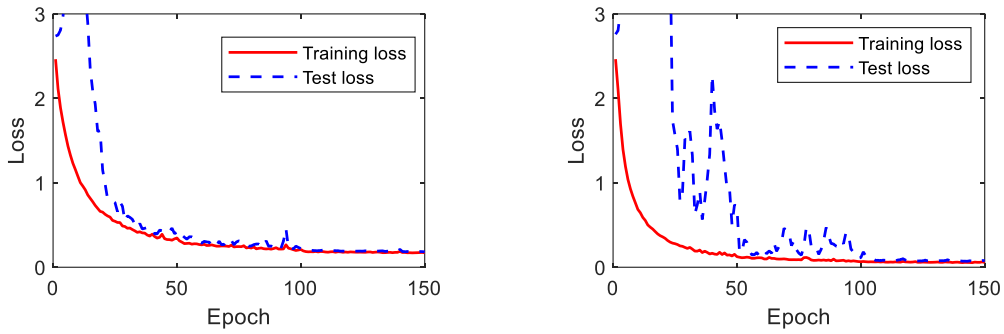
Considering that more gates would consume more training time as shown in Table 5.3 and the G1 consumes least training time while achieves just next to the highest accuracy (See Fig. 5.9), the

results of G1 will be analyzed in the following. Results of other gates are similar and thus not shown to reduce the scale of this chapter.

Table 5.3: Training time consumption of the proposed SAG-CNN over the planetary gearbox dataset.

Model	NA (Baseline)	G1	G2	G3	G12	G13	G23	G123
CPU time (s)	143	179	182	199	206	221	224	242

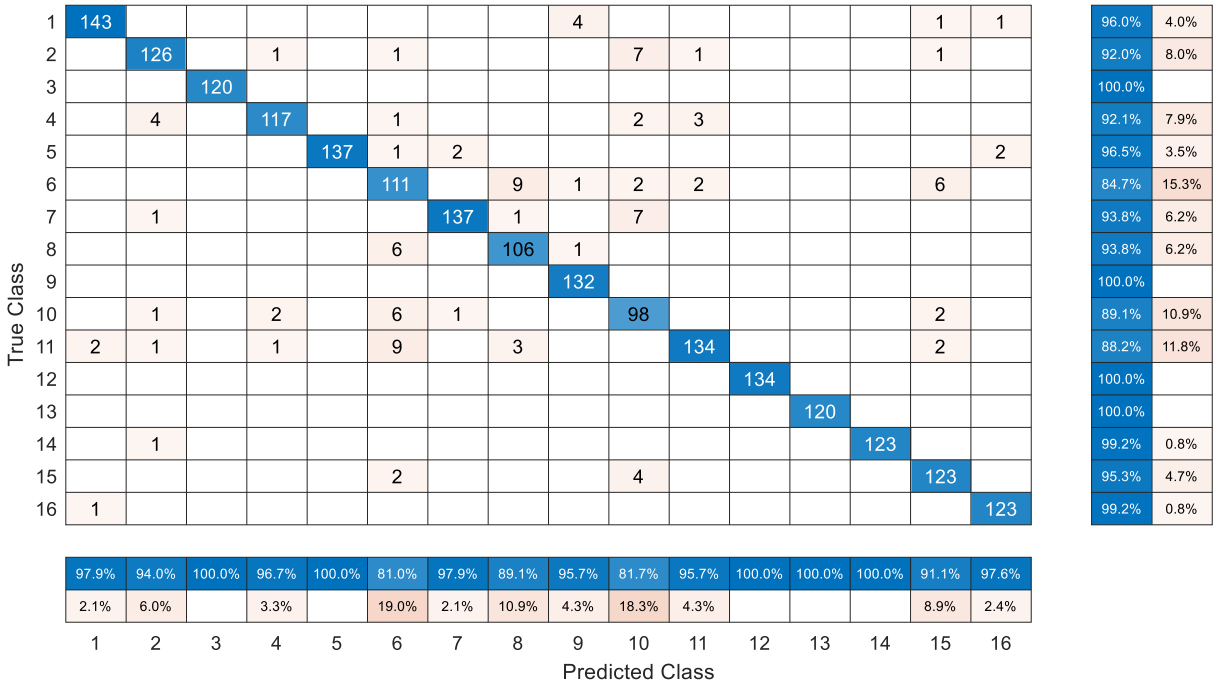
To show the stability of the proposed SAG, the loss of one trial during the training process is given in Fig. 5.10. We can see that before 100 epochs, the proposed SAG-CNN is more fluctuated with the test set. After 100 epochs, both the baseline and the proposed become stable. This means both models can be well converged during the training over the planetary gearbox dataset.



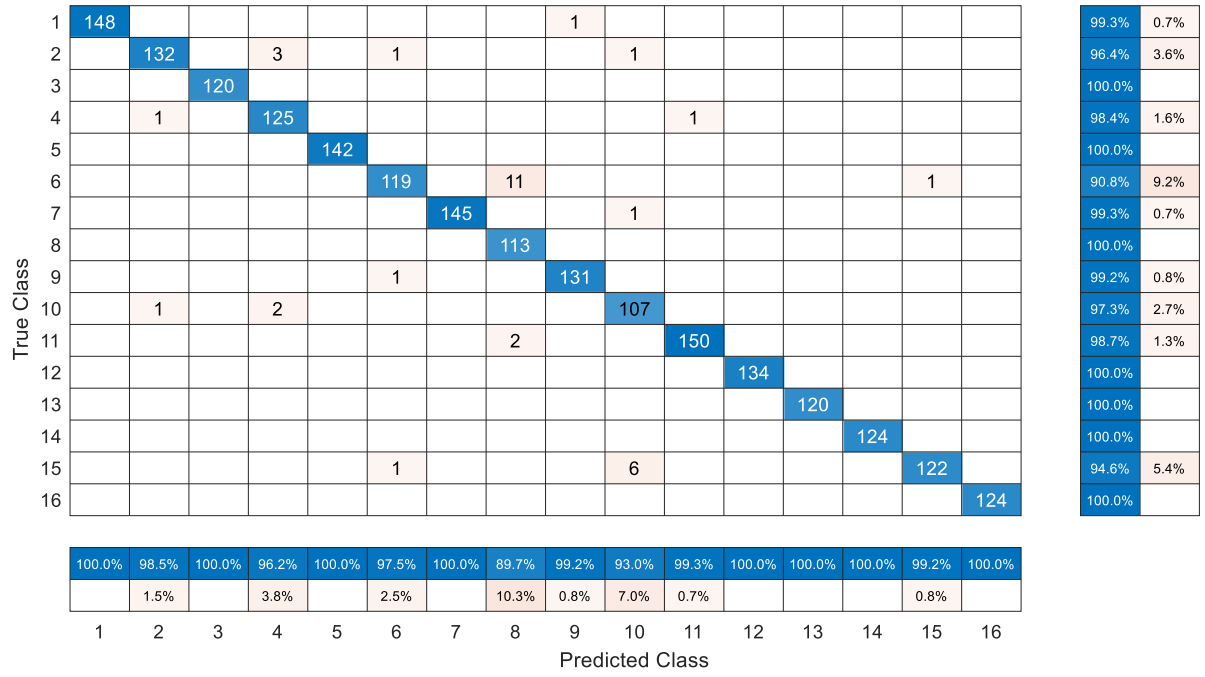
(a) (b)

Fig. 5.10: Training and test loss over the planetary gearbox dataset: (a) Baseline CNN and (b) Proposed SAG-CNN (G1).

To understand the main hits and misses of the classification, the confusion matrix of one trial of the baseline CNN and the proposed SAG-CNN (G1) is shown in Fig. 5.11. We can see the proposed SAG-CNN improves the classification accuracy for all classes. The main miss of both models is the health state 6, i.e., planetary gear chipped tooth.



(a)



(b)

Fig. 5.11: Confusion matrix of the proposed SAG-CNN over the planetary gearbox dataset: (a) Baseline CNN and (b) Proposed SAG-CNN (G1). See Table 5.1 for fault types corresponding to the class numbers.

(b) Effects of speed variation on fault classification accuracy

As discussed in the Section 5.3, the fault classification accuracy of the baseline CNN is supposed to increase with speed due to fault information imbalance. The proposed SAG is expected to address this problem. We plot the accuracy versus speed of the baseline CNN and the SAG-CNN (G1), as shown in Fig. 5.12. Only the result of G1 is shown for simplicity. Since the speed varies continuously, it is infeasible to calculate the fault classification accuracy at a single speed value. We instead calculate the classification accuracy of samples within a speed range as follows.

Step 1: Calculate the average speed of each sample in the test set.

Step 2: Find the range of the average speed calculated in Step 1. For the planetary gearbox dataset, the speed range is 300 rpm – 1500 rpm.

Step 3: Equally split the speed range found in Step 2 into certain subranges. Over here, the speed is split into the following 5 subranges: [300, 600), [600, 900), [900, 1200) and [1200, 1500].

Step 4: Categorize the samples in the test set into corresponding subranges in terms of the average speed of each sample.

Step 5: Calculate the classification accuracy of each subrange.

Step 6: Plot the accuracy of each subrange versus the mid-speed of this range.

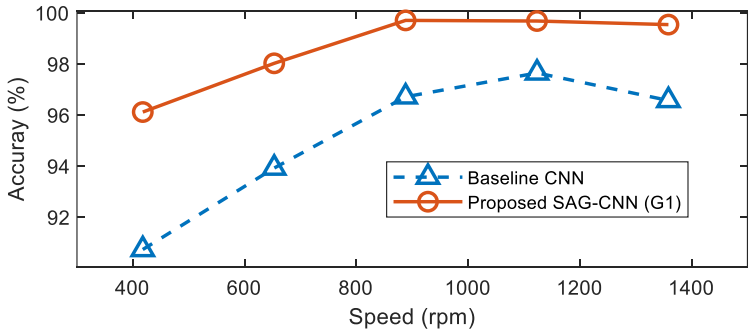


Fig. 5.12: Accuracy versus speed of CNNs over the planetary gearbox dataset.

We can observe that the fault classification accuracy of the baseline CNN generally increases with speed. This validates our hypothesis that the accuracy would increase with speed due to fault information imbalance. An unexpected decline is observed at the end. Possible reasons are analyzed in the Section 5.6.2. We can also observe that with the SAG applied, the fault classification accuracy is improved across the whole speed range, but the accuracy under lower-speed conditions is improved more than that of higher-speed conditions. Visually, the curve of the proposed SAG-CNN is more flattened than that of the baseline CNN. This means effects of speed induced fault information imbalance is mitigated. However, a slight increasing trend between the accuracy and speed is remained. Reasons are discussed in Section 5.6.3.

(c) Leaned SAG values

As indicated in Section 5.3, the learned SAG values are supposed to be larger for lower-speed conditions and vice versa. Fig. 5.13 shows SAG values versus speed of the proposed SAG-CNN over the planetary gearbox dataset. Each point in the figure represents a sample in the test set. The x-axis represents the average speed of this sample. The y-axis presents the average gate values learned for this sample. Still only the result of the G1 is shown for simplicity. We can see that the SAG values do decrease with speed as expected.

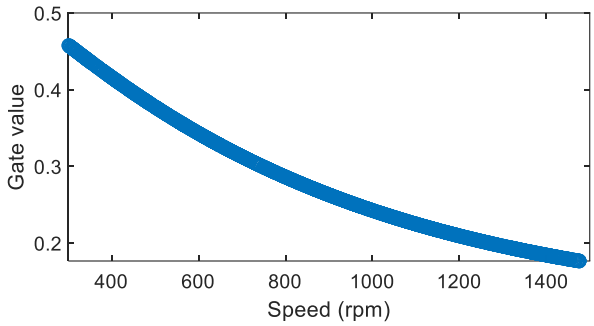


Fig. 5.13: Average SAG values of SAG-CNN (G1) over the planetary gearbox dataset.

5.4.1.4 Comparisons

Here gives the comparisons of the proposed SAG-CNN with two related methods. One is for the class imbalance problem and the other is about speed utilization.

(a) Comparison with reported method for class imbalance problem

As mentioned in Section 5.3, the definition of the speed induced fault information imbalance is inspired by the class imbalance problem. A natural thought is to compare the proposed SAG with existing methods for the class imbalance problem. Reported methods to such problem can be categorized into two types, i.e., data-level methods and algorithm-level methods [171]. Data-level methods modify the data while algorithm-level methods modify existing algorithms. Considering that our proposed SAGs modify existing deep learning models and thus pertain to algorithm-level methods, we are going to compare SAGs with reported algorithm-level methods.

Cost-sensitive methods are among popular algorithm-level methods for the class imbalance problem [172]. Cost-sensitive methods modify the loss function of deep learning models. We manually assign higher costs to minority classes by incorporating class-wise weights. The cost-sensitive loss (CSL) used in [169] will be adopted in this chapter thanks to its simplicity and interpretability,

$$L_{cs} = w(x) \cdot L(x) \tag{5.3}$$

where, $L(x)$ is the loss function of a deep learning model given input x, L_{cs} is the modified loss function, i.e., the cost-sensitive loss, and $w(x)$ is the cost-sensitive weight. In [169], $w(x)= w(c|x)=\frac{\max\{n_c\}_{c=1}^{C}}{n_c}$, wherein, c is the class, C is number of all classes and n_c is the number of

samples of class c. To fit in the speed induced information imbalance problem, the cost-sensitive weight is modified as,

$$w(x) = w(s|x) = \left(\frac{\max\{s \in \mathbf{S}\}}{s}\right)^{\alpha} \tag{5.4}$$

where, s is the average speed of sample x, and $\mathbf{S}$ is the set of average speed of all samples. Considering that the effectiveness of cost-sensitive weights is case dependent, we introduce an hyperparameter α to control the power of $\frac{\max\{s \in \mathbf{S}\}}{s}$ to seek possibly better results.

The cost-sensitive loss will be applied to the baseline CNN. The hyperparameters are the same as that for the baseline CNN as shown in Section 5.4.1. The fault classification results with different α values are shown in Table 5.4. We can see that, the best result (bolded fonts) of the CNN+CSL is only slightly better than that of the baseline CNN, but significantly worse than that of the proposed SAG-CNN. The CNN+CSL consumes similar training time as baseline CNN. This is fair as applying CSL does not change the model scale. The above observations indicate that our proposed SAG outperforms the reported CSL method over the planetary gearbox dataset.

(b) Comparison with reported speed utilization method

In the proposed SAG, the speed signal is directly utilized. This brings a potential benefit that fault related information embedded in speed signals [53] would contribute to a better fault classification performance. Speed signals were also utilized in a reported model named stacked auto-encoder deep convolutional neural network (AE-CNN) [170] but in a different manner. In our proposed SAGed models, the gates (information) learned from speed signals multiply the information learned from vibrations. In AE-CNN [170], fault related information contained in vibration and speed is firstly mined via a CNN and an AE separately and then concatenated for fault

classification. We are going to compare our proposed SAG with the this reported AE-CNN model to show the speed utilization efficiency of our proposed model.

The fault classification results of the reported AE-CNN [170] over the planetary gearbox dataset are given in Table 5.4 too. Different AE structures as shown in Table 5.4 are tried to avoid possibly ad-hoc results. We can observe that, reported AE-CNNs achieve significantly higher fault classification accuracies than that of corresponding baseline models, but still slightly lower than the proposed SAG-CNN. However, the reported AE-CNNs consume at least twice training time of that of the baseline CNN, while the proposed SAG-CNN spends only about 20% more time than the baseline CNN. This means that the proposed SAG-CNN is superior to the reported AE-CNN with higher fault classification accuracies and less strict training time requirements. Please be advised the training time of the reported AE-CNNs is the summation of three parts, i.e., pre-training of CNN, pre-training of AE and fine-tuning of AE-CNN.

Table 5.4: Comparison of the proposed SAG-CNN with CNN+CSL and AE-CNN over the planetary gearbox dataset.

Model	Model Structure	Accuracy		CPU time (s)
		Average	Std	
Baseline CNN [114], [123]	-	95.16%	0.54%	143
CNN+CSL [169]	α=0.1	**95.33%**	0.31%	144
	α=0.3	95.27%	0.31%	147
	α=0.5	95.05%	0.50%	147
	α=1.0	93.92%	0.69%	144
	α=2.0	88.56%	1.00%	145
AE-CNN [170]	1024-512-32-512-1024	97.55%	0.55%	394
	1024-256-32-256-1024	**97.87%**	0.54%	355
	1024-128-32-128-1024	97.75%	0.32%	337
	1024-256-64-256-1024	97.38%	0.48%	350
	1024-256-16-256-1024	97.32%	0.32%	347
Proposed SAG-CNN	G1	**98.63%**	0.28%	179

5.4.2 Case study 2: Fixed-shaft gearbox dataset

5.4.2.1 Dataset description

The fixed-shaft gearbox dataset [144] was collected by the author and colleagues at the University of Alberta, Edmonton, Alberta, Canada in 2018. Detailed information of this dataset has been provided in Section 3.3.3. In this chapter, data under five faulty states will be used. Only a single channel of vibration, i.e., the one in the vertical direction, is going to be used in this chapter. The fixed-shaft gearbox dataset is preprocessed in a same way as that for the planetary gearbox dataset. After preprocessing, we get 7116 samples as shown in Table 5.5**.**

Table 5.5: Summary of the fixed-shaft gearbox dataset.

Speed conditions	# of health states	# of samples	Sample length
Continuously varying (Up and down within180 rpm)	5	7116	1024 data points (0.4 s)

5.4.2.2 Results

Over here, the hyperparameters are the same as that for the planetary gearbox dataset, except that the batch size is halved as we have less samples in the fixed-shaft gearbox dataset. The fault classification results of the fixed-shaft gearbox dataset are briefly given below.

(a) Accuracy

The fault classification accuracy of the proposed SAG-CNN over the fixed-shaft gearbox dataset is given in Fig. 5.14. We can observe similar trends as that of the planetary gearbox dataset. The proposed SAG-CNN (speed gated) outperforms the baseline CNN. This means our proposed SAG-CNN are effective with the fixed-shaft gearbox dataset. The highest accuracy is 90.01% achieved

when G1 applied, compared to 87.37% out of the baseline CNN. Likewise, the results with G1 applied will be analyzed in the following.

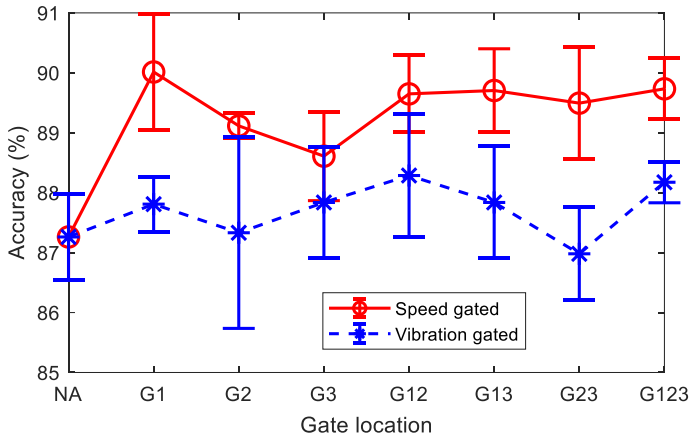


Fig. 5.14: Fault classification results of the proposed SAG-CNN over the fixed-shaft gearbox dataset. NA-Baseline model.

The training and test losses over the fixed-shaft gearbox dataset are shown in Fig. 5.15. Both are relatively stable. The confusion matrix is shown in Fig. 5.16. We can see that the major misses with the baseline CNN, i.e., the Classes 1 and 3, are better classified with the proposed SAG-CNN.

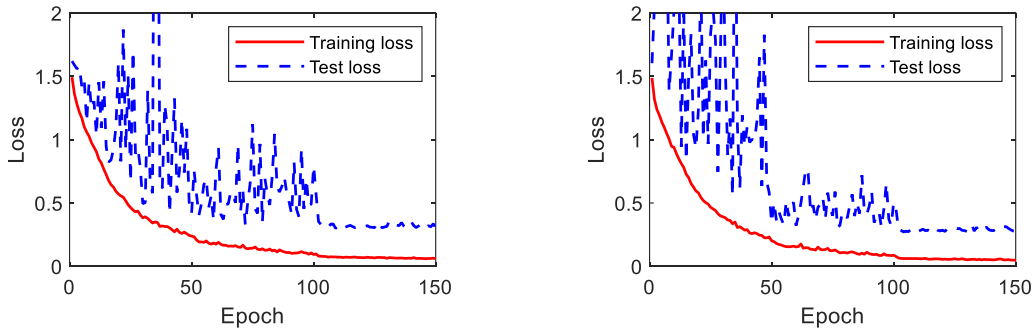


Fig. 5.15: Training and test loss over the fixed-shaft gearbox dataset: (a) Baseline CNN and (b) Proposed SAG-CNN (G1)

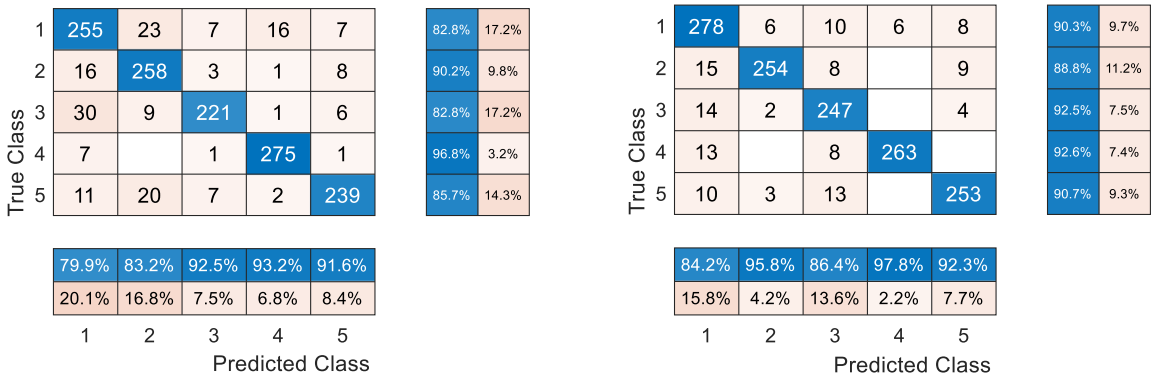


Fig. 5.16: Confusion matrix over the fixed-shaft gearbox dataset: (a) Baseline CNN and (b) Proposed SAG-CNN (G1).

(b) Effects of speed variation on fault classification accuracy

Likewise, the relationship between the accuracy and the speed over the fixed-shaft gearbox dataset is shown in Fig. 5.17 to show the effects of speed variation on the fault classification accuracy of deep learning models. For the fixed-shaft gearbox dataset, similar trends are observed as that of the planetary gearbox dataset. Compared to the baseline CNN, the increasing trend is flattened. This means that the proposed SAG does address the speed induced fault information imbalance with the fixed-shaft gearbox dataset.

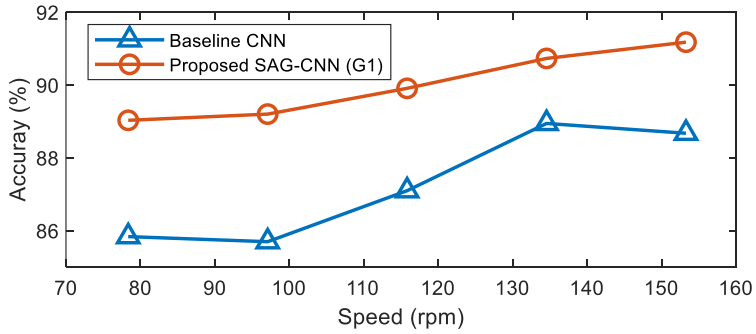


Fig. 5.17: Accuracy versus speed of CNNs over the fixed-shaft gearbox dataset.

(c) Leaned SAG values

The learned SAG values over the fixed-shaft gearbox dataset are shown in Fig. 5.18. Only results of G1 are shown for the purpose of simplicity. The SAG value decreases with speed as expected.

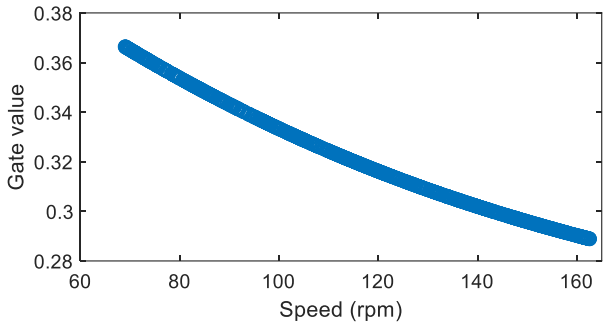


Fig. 5.18: Average SAG values of SAG-CNN (G1) over the fixed-shaft gearbox dataset.

5.4.2.3 Comparisons

The performance of the proposed SAG-CNN is also compared with the reported CNN+CSL and AE-CNN. The results are shown in Table 5.6. Compared to the CNN+CSL, the proposed SAG-CNN achieves significantly higher accuracy. Compared to the AE-CNN, the proposed SAG-CNN achieve slightly better accuracy but consumes much less training time.

5.5 Case studies with SAG-ResNet

Section 5.4 has validated the effectiveness of the proposed SAG when applied to the CNN, i.e., SAG-CNN, with two case studies. This section will conduct the same work to show the effectiveness of the SAG when applied to the ResNet, i.e., SAG-ResNet.

Table 5.6: Comparison of the proposed SAG-CNN with CNN+CSL and AE-CNN over the fixed-shaft gearbox dataset.

Model	Model Structure	Accuracy		CPU time (s)
		Average	Std	
Baseline CNN [114], [123]	-	87.26%	0.72%	118
CNN+CSL [169]	α=0.1	87.37%	0.54%	118
	α=0.3	**87.57%**	0.36%	116
	α=0.5	87.04%	1.32%	116
	α=1.0	86.99%	1.41%	115
	α=2.0	86.19%	1.31%	119
AE-CNN [170]	1024-512-32-512-1024	89.00%	0.66%	283
	1024-256-32-256-1024	89.02%	0.99%	281
	1024-128-32-128-1024	**89.21%**	0.62%	278
	1024-256-64-256-1024	88.60%	0.57%	278
	1024-256-16-256-1024	88.85%	0.50%	278
Proposed SAG-CNN	G1	**90.01%**	0.96%	132

5.5.1 Case study 1: Planetary gearbox dataset

The same planetary gearbox dataset and the same hyperparameters as that for the SAG-CNN are adopted for the SAG-ResNet. The classification accuracy is shown in Fig. 5.19. Comparison results with reported methods are shown in Table 5.7. For the reported methods [169], [170], we simply replace the baseline model from the CNN to the ResNet. Similar trends as that for the SAG-CNN are observed. The proposed SAG-ResNet archives higher accuracy than that of the baseline ResNet, the reported ResNet+CSL and the reported AE-ResNet. This means that the proposed SAG-ResNet works with the planetary gearbox dataset. For tentative SAGs for the ResNet, the GR1 is preferred as it achieves significantly high accuracy (Fig. 5.19) while has a simpler structure as shown in Fig. 5.7).

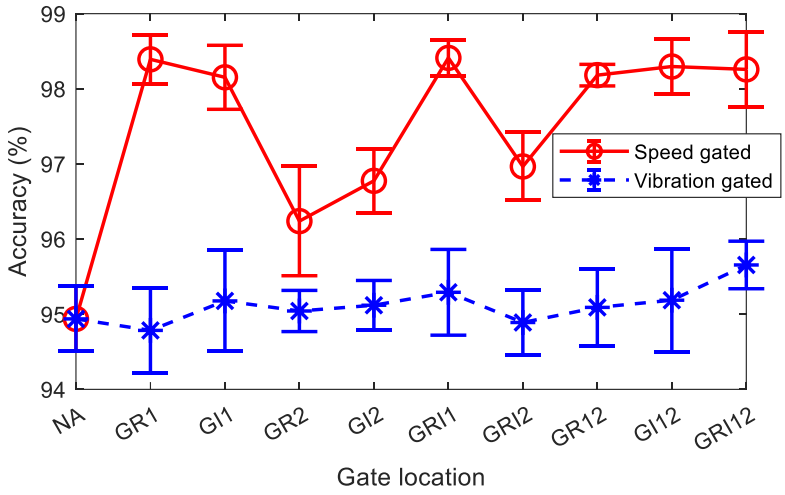


Fig. 5.19: Fault classification results of the proposed SAG-ResNet over the planetary gearbox dataset. NA-Baseline model.

Table 5.7: Comparison of the proposed SAG-ResNet with ResNet+CSL and AE-ResNet over the planetary gearbox dataset.

Model	Model Structure	Accuracy		CPU time (s)
		Average	Std	
Baseline ResNet [114], [123]	-	94.94%	0.43%	181
ResNet+CSL [169]	α=0.1	**94.96%**	0.60%	190
	α=0.3	94.87%	0.48%	187
	α=0.5	94.76%	0.57%	185
	α=1.0	93.84%	0.51%	173
	α=2.0	88.88%	0.72%	187
AE-ResNet [170]	1024-512-32-512-1024	97.13%	0.28%	467
	1024-256-32-256-1024	**97.55%**	0.40%	425
	1024-128-32-128-1024	97.22%	0.51%	429
	1024-256-64-256-1024	97.53%	0.29%	430
	1024-256-16-256-1024	97.18%	0.35%	429
Proposed SAG-ResNet	GR1	**98.39%**	0.33%	208

5.5.2 Case study 2: Fixed-shaft gearbox dataset

Likewise, the same fixed-shaft gearbox dataset and the same hyperparameters as that for the SAG-CNN are adopted. Results are provided in Fig. 5.20 and Table 5.8. Still, similar trends as that for the SAG-CNN are observed. The proposed SAG-ResNet outperforms the baseline ResNet, the reported ResNet+CSL and the reported AE-ResNet with the fixed-shaft gearbox dataet. Still, the GR1 achieves the best performance.

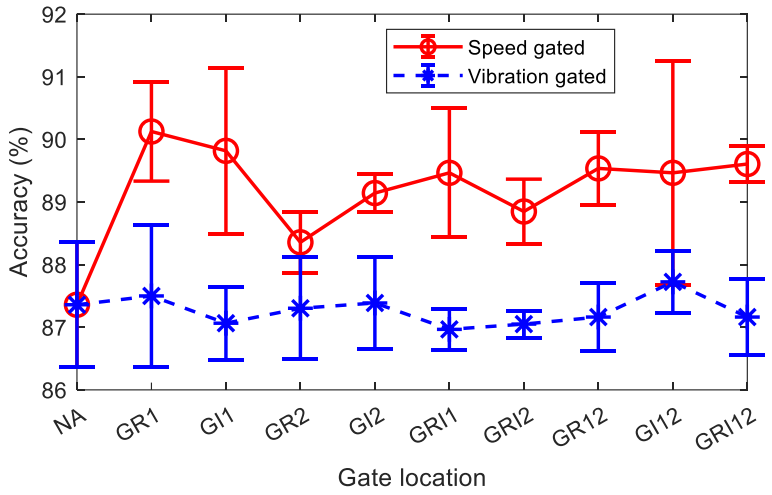


Fig. 5.20: Fault classification results of the proposed SAG-ResNet over the fixed-shaft gearbox dataset. NA-Baseline model.

To this end, we have validated the effectiveness of the proposed SAG with CNNs and ResNets over a fixed-shaft gearbox dataset and a planetary gearbox dataset, respectively. The results empirically show the generalization of the proposed SAG across different deep learning models and different rotating machines. Hopefully the proposed SAG would work with other deep learning models like the feedforward neural network and the recurrent neural network, and other rotating machines like pumps, generators, bearings, and rotors.

Table 5.8: Comparison of the proposed SAG-ResNet with ResNet+CSL and AE-ResNet over the fixed-shaft gearbox dataset.

Model	Model Structure	Accuracy		CPU time (s)
		Average	Std	
Baseline ResNet [114], [123]	-	87.36%	1.00%	127
ResNet+CSL [169]	α=0.1	87.60%	0.95%	128
	α=0.3	87.64%	0.77%	127
	α=0.5	**88.03%**	0.99%	129
	α=1.0	87.75%	0.72%	126
	α=2.0	85.88%	1.09%	126
AE-ResNet [170]	1024-512-32-512-1024	**89.42%**	1.09%	311
	1024-256-32-256-1024	89.33%	0.64%	320
	1024-128-32-128-1024	89.30%	0.76%	309
	1024-256-64-256-1024	89.26%	0.72%	305
	1024-256-16-256-1024	89.42%	0.88%	313
Proposed SAG-ResNet	GR1	**90.13%**	0.79%	146

5.6 Discussion

5.6.1 Why proposed SAG works

Our proposed SAG works because it adjusts the learning emphasis regarding of speed. Such that the speed induced fault information imbalance is mitigated. This is like the reported work (e.g., CSL [169]) for the class imbalance problem wherein the emphasis is adjusted in terms of classes. In our proposed SAGed models, the emphasis is adjusted through SAG values, which are automatically learned in the training of SAGed models. The automation process outperforms reported methods for class imbalance like CSL wherein the emphasis is manually assigned.

To interpret how the information is balanced, we analyze the learned features within the deep learning model before and after the SAG applied. Only the SAG-CNN over the planetary gearbox is shown for illustration purpose. A 60 s long measurement is preprocessed (Ring gear tooth missing; including 131 samples with a length of 0.4 s after preprocessing) and then inputted to a well-trained SAG-CNN (G1). This measured vibration and speed, learned SAG, learned features before and after the SAG applied, and the spectrograms of the features are shown in Fig. 5.21.

At the location of G1, both the SAG and the features contain 32 channels. The SAG of all channels is shown. Each curve corresponds to a channel. The features of channel #24 is randomly selected for illustration purpose. We can see that the amplitude of the features is relatively flattened or balanced after SAG applied. This means that the SAG mitigates the effects of speed in the amplitude side. After SAG applied, the spectrogram (STFT) of the features is like that before SAG applied. This means the SAG does not address the effects of speed in the frequency side, otherwise the harmonics in the STFT of the features after SAG applied should be flattened, not the current speed ones that change with speed. In a word, the proposed SAG works because it balances the information in features, but only in the amplitude side while the frequency side is not addressed.

More explanations over here are as follows. First, the reason to show the STFT of features is for the interpretation of readers not the deep learning models. Second, frequency aliasing is observed in the STFT of features. This is because that the pooling layer in the model (See Fig. 5.6) down-samples the data without preceding low-pass filtering.

Another reason to facilitate the performance of the proposed SAG is the use of speed. This is evidently supported by the results that speed gated outperforms vibration gated as shonw in Fig. 5.9, Fig. 5.14, Fig. 5.19 and Fig. 5.20. Considering that the reported speed utilization method (AE-

CNN and AE-ResNet) [170] also achieves competitive performances, we can infer that properly using speed signals does help improve the fault classification performances of existing deep learning models.

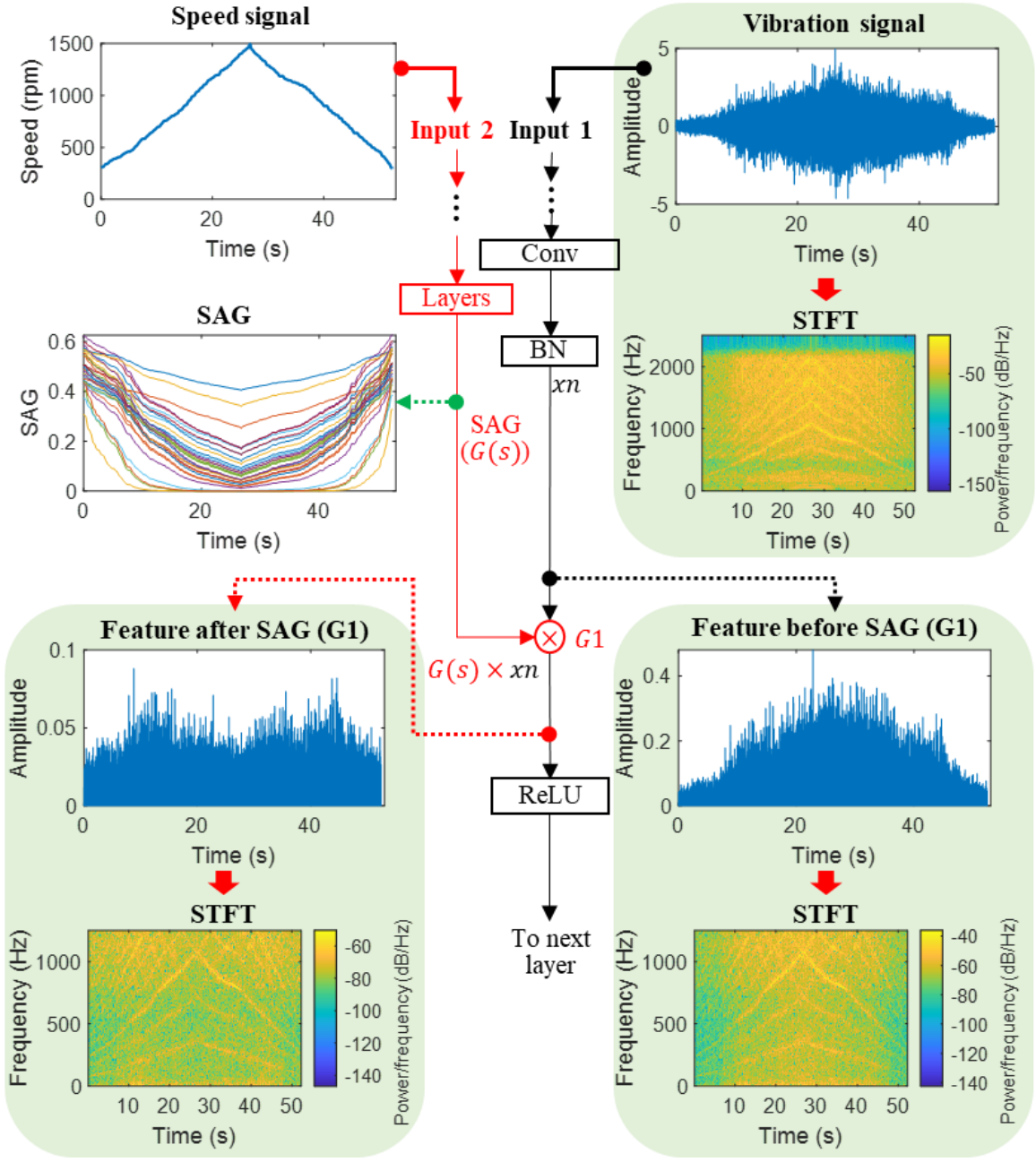


Fig. 5.21: Interpretation of how SAG works. The features of channel #24 are illustrated as an example.

Given that the proposed SAG only addresses the effects in the amplitude side, and the order tracking, a widely used signal processing technique which can balance the number of fault signatures in samples by resampling the time signal into order domain [94], [106], is able to address the effects in the frequency side, a possibly complete solution to the problem of rotating machinery fault classification under varying speed conditions is to integrate them. That is, resampling the signal using order tracking first, and then the resampled signal is processed by deep learning models for fault classification. The results of the baseline CNN and the proposed SAG-CNN over the planetary gearbox dataset are shown in Table 5.9 for illustration purpose. Only the SAG applied at the location of G1 is discussed over here as it achieves the best performances among all locations as seen in the case studies. The resampled dataset is as follows. The number of samples is 7534 with a sample length of 2880 data points. The resampling frequency is 360 points per revolution.

Table 5.9: Fault classification accuracy of the proposed SAG-CNN (G1) over resampled data using order tracking with the planetary gearbox dataset.

Data	Model	Accuracy	Standard derivation
Raw signal (Time domain)	Baseline CNN	95.16%	0.54%
	Proposed SAG-CNN	98.63%	0.28%
Resampled signal (Order domain)	Baseline CNN	96.84%	0.29%
	Proposed SAG-CNN	99.12%	0.13%

We can see from Table 5.9 that, with the order tracking adopted, both the baseline CNN and the proposed SAG-CNN outperform their counterparts with the raw vibration data. This means that using order tracking to balance the number of fault signatures does help in improving the fault classification accuracy. With the resampled data in the order domain, the proposed SAG-CNN

achieves higher accuracy than that of the baseline CNN. This is because the proposed SAG further addresses the effects in the amplitude side.

5.6.2 Unexpected accuracy decrease with planetary gearbox dataset

In Fig. 5.12, the accuracy of the baseline CNN over the planetary gearbox dataset shows an unexpected decreasing trend when speed > 1200 rpm. Possible reasons may be in the data side or the model side. We will examine both in the following.

(a) Data side

Here checks whether the reason is due to the data or not. To understand why the data with speed > 1200 rpm results in smaller fault classification accuracy, we double check the spectrogram of collected vibration as shown in the top right panel of Fig. 5.21. The outstanding curves are harmonics of the speed and are supposed to reveal the fault signatures [8]. We can see that even the harmonics are strong with speed > 1200 rpm, the background noise is even stronger than neighbors, making the fault signatures less obvious. Such phenomenon can be quantitatively measured using the signal-to-noise ratio (SNR). Fig. 5.22 shows the SNR of this measurement over time. The SNR over here is the logarithm of the ratio between the spectra power of the harmonics to the residual spectra power. The harmonics are extracted using the maximum tracking algorithm from the spectrogram [55]. We can see that the SNR with speed > 1000 rpm is smaller than its neighbors, making the data harder to be learned, and thus resulting in the unexpected decreasing trend in such a speed range (> 1200 rpm). The relatively lower SNR may be because that the structure resonance of the gearbox is excited within this speed range.

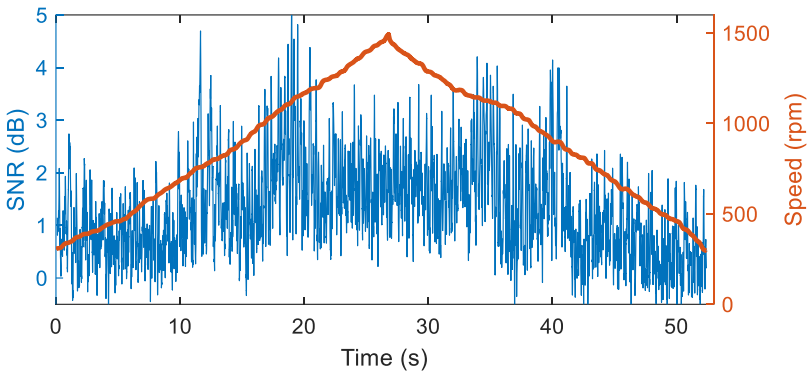


Fig. 5.22: SNR of a measured vibration of the planetary gearbox.

(b) Model side

Here checks whether the unexpected trend is caused by the model or not. Different hyperparameters for baseline models are tried out for fault classification. For illustration purpose, only the results of the baseline CNN are shown in Fig. 5.23. The tried hyperparameters are as follows.

- Setting 1: 5 convolutional layers with numbers of channels = (16, 16, 16, 32, 32) and kernel sizes = (5, 3, 3, 3, 3)
- Setting 2: 5 convolutional layers with numbers of channels = (64, 64, 64, 128, 128) and kernel sizes = (5, 3, 3, 3, 3)

Recall that the default setting in case studies is an CNN consisted of 5 convolutional layers with number of channels = (32, 32, 32,64, 64) and kernel sizes = (5, 3, 3, 3, 3). We can see that the decreasing trend at the end is observed with all settings but is eased when the model scale is larger (Setting 2). This means that a larger model can alleviate the effects of SNR.

Given above discussions, we can conduct that the unexpected accuracy decrease at the end is because of the relatively smaller SNR of the vibration data. Such decreasing trend can be alleviated through larger models.

An interesting observation from Fig. 5.22 is that the SNR of the vibration signal under lower speeds is generally smaller than that of higher speeds. This may be because that, given ambient noise, larger speed would induce more fault related energy to the vibration. The SNR of the vibration will increase accordingly. A common sense is that it is often more difficult to learn fault features from more noisy (lower SNR) data [26]. Such that the fault classification accuracy with lower SNR data, i.e., under lower speed condition, would be smaller. Therefore, the speed induced SNR difference is another reason for the overall accuracy increase in terms of speed.

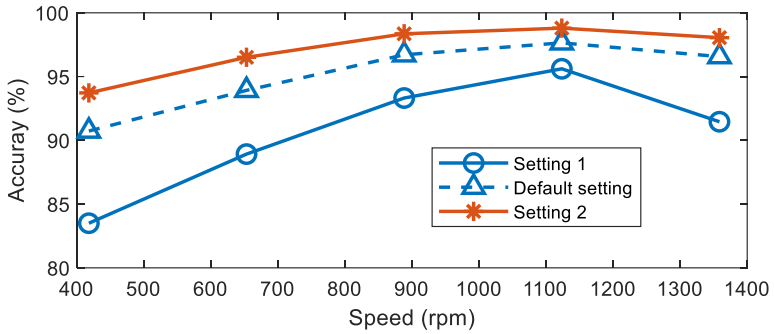


Fig. 5.23: Accuracy versus speed of the baseline CNN over the planetary gearbox dataset with different hyperparameter settings.

5.6.3 Remaining increasing trend in accuracy

As noted in the results of all case studies, even the speed induced fault information imbalance is somewhat addressed, the fault classification accuracy is not as expected to be evenly distributed across speed. A slight increasing trend is still observed between the speed and the accuracy. One

possible reason is that the SNR of vibration signals increases with speed as shown in Fig. 5.22. The SNR of data with lower speeds is smaller and thus hard to be classified. Another possible reason, as mentioned in Section 5.6.1, is that the proposed SAG only addresses the effects of speed variation in the amplitude side, while the frequency side is not addressed. The fault information imbalance due to the frequency side may lead to lower accuracy for the lower speeds.

5.6.4 Limitations

An obvious limitation of the proposed SAG is that the speed is mandatory. The SAG is, thus, not applicable when the speed signals are not available. We only considered the constant load condition. For the varying load conditions, we may simply switch the auxiliary input from the speed to the load. How and why a load adaptive deep learning model may work need further exploration. In this chapter, the SAG has been applied to CNNs and ResNets. It may also be applied to other deep learning models like the LSTM [113].

5.7 Summary and conclusion

In this chapter, we firstly investigate the effects of speed variation from the perspective of deep learning, and then propose an SAG for existing deep learning models to address the effects of speed variation for the task of fault classification of rotating machinery under varying speed conditions. The effectiveness of the proposed SAG is empirically validated with two baseline models including an CNN and a ResNet over two experiment datasets including a planetary gearbox dataset and a fixed-shaft gearbox dataset. Major conclusions are drawn as follows,

(1) Speed variation induces fault information imbalance in vibration signals, which further deteriorates the fault classification accuracy of deep learning models, especially under lower-speed conditions.

(2) The proposed SAG addresses the speed induced fault information imbalance problem and thus improves the fault classification accuracy of baseline deep learning models.

(3) To achieve better fault classification performances, a single SAG is preferred, and the SAG is better to be added to early layers of deep learning models.

(4) Speed signals matter. Properly utilizing speed signals as an auxiliary input to deep learning models can improve their fault classification accuracy.

In the future, we are going to explore simpler structures for the SAG and address the effects of SNR difference due to speed variation. How to automatically address the effects of the frequency side, i.e., more frequent fault signatures with higher speeds, will be another future work. One possible solution is to integrate the order tracking into the baseline CNN or ResNet to remove such effects automatically.

6. Summary and future work

This chapter summarizes the work of this thesis and suggests works that deserve future investigation.

6.1 Summary

Rotating machinery is often subjected to faults. Fault diagnosis is to detect the occurrence of a fault and identify the root cause of an occurred fault. The outcome of fault diagnosis facilitates predictive maintenance. Such that, unexpected machine shutdown can be avoided, the uptime is increased, and ultimately the maintenance cost is reduced. Deep learning is an emerging and promising tool that could enable automated fault diagnosis and is capable of processing massive data, and thus attracts increasing attentions nowadays. Considering that rotating machinery usually operates under varying speed conditions, this research focuses on deep learning-based fault diagnosis for such conditions. The machinery of interest is the typical rotating machinery such as gearboxes, bearings, and rotor systems. Major works of this thesis are summarized as follows.

Rotating speed extraction from vibration signals: Rotating speed is not only an important condition monitoring indicator, but also contributes to effective fault diagnosis. For scenarios wherein the speed cannot be measured, we often need to extract speed from vibration signals. Existing speed extraction method, the many to one long short-term memory (LSTM) model, did not adequately exploit the information in vibration, thus leaving room for improving its speed

extraction accuracy. To address this problem, we proposed a deep learning model named many-to-many-to-one bi-directional long short-term memory (MMO-BiLSTM) to automatically extract rotating speed from vibration signals. The proposed model consisted of two parts, i.e., a many-to-many BiLSTM part and a many-to-one LSTM part. The BiLSTM part learned information from vibration signals in both forward-time and backward-time directions. The LSTM part future enhanced the speed extraction accuracy of the BiLSTM part. The two parts were trained separately with a two-stage training manner. Case studies over an internal combustion engine dataset, a rotor system dataset, and a fixed-shaft gearbox dataset showed that the proposed MMO-BiLSTM could effectively extracted speed from vibration signals. The percentage speed extraction errors for the three datasets were 1.03%, 1.50% and 1.51%, yielding relative error reductions of 67.30%, 54.00% and 25.61% over the reported MO-LSTM model, respectively.

Fault detection of rotating machinery under varying speed conditions: Rotating machinery often works under varying speed conditions. Fault detection is effective to prevent machine failures. Existing autoencoder based (AE-based) fault detection methods did not address the effects of speed variation, and thus performed poorly under varying speed conditions. To address this problem, we proposed a new deep learning model named speed normalized autoencoder (SN-AE). The SN-AE consisted of two branches, i.e., a speed normalization (SN) branch and an AE branch. The SN branch took the speed signal as the input. It automatically learned an SN function, which further normalized the vibration signal to remove the effects of speed variation. The normalized vibration signal was inputted to the AE branch for fault detection. Case studies were conducted to detect faults of three typical rotating machines including a planetary gearbox, a fixed-shaft gearbox and a rolling element bearing. Results showed that the proposed SN-AE successfully removed the effects of speed variation, and achieved detection performances (i.e., area under the receiver

operator characteristic curve) of 0.9535, 0.9227 and 0.9981 for the three datasets, respectively. Corresponding relative performance improvements over the baseline AE model were 38.93%, 6.64% and 7.67%, respectively.

Fault classification of rotating machinery under varying speed conditions: Once a fault is detected, the very next step is to classify the type and the severity of this fault. Knowing the information of the type and the severity of an occurred is critical for successive maintenance decision making. Existing deep learning models such as convolutional neural networks (CNNs) and residual networks (ResNets), for fault classification did not unfold the effects of speed variation on the fault detection performances and did not address such effects specifically. To address these problems, we firstly investigated the effects of speed variation from the perspective of deep learning. We found that the fault information in vibration signals was imbalanced due to speed variation. The imbalance made the fault classification accuracy of existing deep learning models changing with speed, specifically, increasing with speed. We then proposed an auxiliary branch named speed adaptive gate (SAG) for existing deep learning models to address the speed induced fault information imbalance. The SAG took speed signals as the input. It controlled the information flow of deep learning models in terms of speed, such that the effects of speed variation were mitigated. Case studies with two baseline models, i.e., a CNN and a ResNet, over two experimental datasets, i.e., a planetary gearbox dataset and a fixed-shaft gearbox dataset, validated the effectiveness of the proposed SAG for fault classification under varying speed conditions. Results showed that the proposed SAG achieved fault classification accuracies of about 95.51% and 90.07% for the two datasets, yielding relative improvements over existing deep learning models of about 3.64% and 3.16%, respectively, and the unexpected increasing trend between the accuracy and speed was eased.

Major contributions of this thesis are summarized as follows.

- A deep learning model to automatically extract rotating speed from vibration signals was developed. The developed model used the BiLSTM and followed a many-to-many-to-one learning mode. The developed model exploited more speed related information in the vibration signals and thus achieved higher speed exaction accuracy.
- A deep learning model for effective fault detection of rotating machinery under varying speed conditions was developed. The developed model removed the amplitude modulation effects induced by speed variation, and thus improved the fault detection performance of existing deep learning models.
- A deep learning model for effective fault classification of rotating machinery under varying speed conditions was developed. The developed model mitigated the information imbalance induced by speed variation, and thus promoted the fault classification accuracy of existing deep learning models.

6.2 Future work

Based on the research of this thesis, we suggest three research topics for future study as follows.

Fault diagnosis under varying load conditions

The load is assumed constant in this thesis. This limits the application scenarios of the methods proposed in the thesis. Indeed, there are scenarios wherein the load changes over time like the speed in industry. For example, the motor of an elevator may experience varying load when the passengers are getting on board and off board. Load variation often leads to amplitude modulation

of vibration signals, and slight speed fluctuation [173], [174], [175]. The speed fluctuation furthers induce frequency modulation in vibration signals. This is somewhat alike the effects induced by speed variation.

Fault diagnosis under varying load conditions may be achieved by simply adopting methods proposed in the presented thesis. We only need to change the auxiliary input from the speed signals to the load signals. A challenge is how to obtain the load signals, which is even more difficult than to obtain speed signals. To this end, the further work for fault diagnosis under varying load conditions may need to be carried out without access to load signals. An even more aggressive scenario is that the speed and the load vary simultaneously. The variations of them can be dependent or independent. How to carry out fault diagnosis with such a scenario is also worth future exploration.

Multi-sensor fusion for fault diagnosis

In Chapters 4 and 5, only a single channel of vibration signals is used for fault diagnosis. This is an obvious limitation as we did not use all available data and thus leave room for the improving the fault diagnosis performances. In condition monitoring systems and/or experimental tests, multiple sensors are often used to collect diverse condition monitoring data. The multiple-sensor data can be of a same type but from different measurement locations of a machine, and/or different types, such as acceleration, acoustic, and current. It is straightforward to use data collected by all sensors for fault diagnosis. More sensors often mean more information, which is believed to contribute to a better performance than a single sensor [176], [177]. However, it is not guaranteed that the more sensors the better. Different sensors or sensors at different installation locations are different in sensitivities to faults. Besides, information abundancy may exist among sensitive

sensors, which would deteriorate the fault diagnosis performance of a deep learning model [178], [179]. A possible future work is to design proper fusion algorithms that can select sensitive and independent sensors while use the information collected by all sensors as much as possible.

Interpretation of deep learning-based fault diagnosis

In Chapters 3 – 5, we proposed three deep learning models for rotating speed extraction, fault detection and fault classification, respectively. The performances of these models were validated with certain case studies. However, we do not know, mathematically, why such good performances are achieved. For example, in our proposed SN-AE, we know the SN-AE works because it learns a speed normalization function, which normalizes the vibration to remove the effects of speed variation. The fault detection performance is improved accordingly. However, we do not know, theoretically, why such a speed normalization function can be learned. This may raise doubts that the good performances were ad-hoc. Indeed, even deep learning has achieved admirable success in not only the field of PHM, a common concern which is also a fundamental problem is that why deep learning works lacks solid theoretical support, or simply, remains a black box. Indeed, the situation is that if a deep learning model can return a sufficiently good performance such as high accuracy with a limited number of case studies, we then claim it effective.

We acknowledge that the interpretation of why deep learning-based fault diagnosis works is quite challenging and may pretty much rely on the breakthrough in the field of deep learning research. We still suggest it as a future work because it is of vital importance, otherwise we may face with one more pitfall as seen in the history of deep learning [26].

References

[1] A. Brkovic, D. Gajic, J. Gligorijevic, I. Savic-Gajic, O. Georgieva and S. Gennaro, "Early fault detection and diagnosis in bearings based on logarithmic energy entropy and statistical pattern recognition," in *2nd International Electronic Conference on Entropy and Its Applications*, Online, 2015.

[2] M. Bengtsson and G. Lundström, "On the importance of combining “the new” with “the old” – One important prerequisite for maintenance in Industry 4.0," in *8th Swedish Production Symposium*, Stockholm, Sweden, 2018.

[3] A. K. Jardine, D. Lin and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing,* vol. 20, no. 7, pp. 1483-1510, 2016.

[4] P. O'Connor and A. Kleyner, Practical reliability engineering, Hoboken, New Jersey, USA: John Wiley & Sons, 2012.

[5] Y. Lei, Intelligent fault diagnosis and remaining useful life prediction of rotating machinery, Waltham, Massachusetts, USA: Butterworth-Heinemann, 2016.

[6] A. I. B. Norway, "Investigation of helicopter accident at turoy near bergen in hordaland county, norway," AIBN, Lillestrom, 2017.

[7] G. Wallace and L. Ly, "NTSB says videos of Ohio train derailment include one showing wheel bearing in ‘final stage of overheat failure’," CNN, 21 2 2023. [Online]. Available: https://www.cnn.com/2023/02/14/us/ohio-train-derailment-investigation/index.html. [Accessed 23 2 2023].

[8] R. B. Randall, Vibration-based condition monitoring: Industrial, aerospace and automotive applications (2nd edition), Hoboken, New Jersey, USA: Johm Wiley & Sons, 2021.

[9] B. Wire, "Global Machine Condition Monitoring Market Size is Estimated to be USD 2.6 Billion in 2019 & is Projected to Reach USD 3.9 Billion by 2025 - ResearchAndMarkets.com," 30 1 2020. [Online]. Available: https://www.businesswire.com/news/home/20200130005363/en/Global-Machine-Condition-Monitoring-Market-Size-Estimated. [Accessed 20 6 2022].

[10] K. L. Tsui, N. Chen, Q. Zhou, Y. Hai and W. Wang, "Prognostics and health management: A review on data driven approaches," *Mathematical Problems in Engineering,* vol. 2015, pp. 1-17, 2015.

[11] L. Biggio and I. Kastanis, "Prognostics and health management of industrial assets: Current progress and road ahead," *Frontiers in Artificial Intelligence,* vol. 88, pp. 1-24, 2020.

[12] J. Qu, Support-vector-machine-based diagnostics and prognostics for rotating systems, Edmonton, Alberta, Canada: PhD Thesis, University of Alberta, 2011.

[13] K. Javed, R. Gouriveau and N. Zerhouni, "State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels," *Mechanical Systems and Signal Processing,* vol. 94, pp. 214-236, 2017.

[14] H. Sarih, A. Tchangani, K. Medjaher and E. Pere, "Data preparation and preprocessing for broadcast systems monitoring in PHM framework," in *IEEE 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, Paris, France, 2019.

[15] L. Liu, Vibration signal analysis for planetary gearbox fault diagnosis, Edmonton, Alberta, Canada: PhD Thesis, University of Alberta, 2018.

[16] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing,* vol. 115, pp. 213-237, 2019.

[17] J. Liu and Y. Shao, "An improved analytical model for a lubricated roller bearing including a localized defect with different edge shapes," *Journal of Vibration and Control,* vol. 24, no. 17, pp. 3894-3907, 2018.

[18] X. Liang, M. J. Zuo and Z. Feng, "Dynamic modeling of gearbox faults: A review," *Mechanical Systems and Signal Processing,* vol. 98, pp. 852-876, 2018.

[19] Z. Feng and M. J. Zuo, "Vibration signal models for fault diagnosis of planetary gearboxes," *Journal of Sound and Vibration,* vol. 331, no. 22, pp. 4919-4939, 2012.

[20] Y. Murakami, T. Takagi, K. Wada and H. Matsunaga, "Essential structure of S-N curve: Prediction of fatigue life and fatigue limit of defective materials and nature of scatter," *International Journal of Fatigue,* vol. 146, p. 106138, 2021.

[21] M. Rao, X. Yang, D. Wei, Y. Chen, L. Meng and M. J. Zuo, "Structure fatigue crack length estimation and prediction using ultrasonic wave data based on ensemble linear regression and Paris' law," *International Journal of Prognostics and Health Management,* vol. 11, no. 2, pp. 1-14, 2020.

[22] S. R. Saufi, Z. A. B. Ahmad, M. S. Leong and M. H. Lim, "Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review," *IEEE Access,* vol. 7, pp. 122644-122662, 2019.

[23] S. Zhang, S. Zhang, B. Wang and T. G. Habetler, "Machine learning and deep learning algorithms for bearing fault diagnostics-a comprehensive review," *arXiv preprint arXiv,* no. 1901.08247, pp. 1-21, 2019.

[24] P. D. A. J. D.-R. Larrañaga, A. Ogbechie, C. E. Puerto-Santana and C. Bielza, Industrial applications of machine learning, Boca Raton, Florida, USA: CRC Press, 2018.

[25] F. Jia, Y. Lei, J. Lin, X. Zhou and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," *Mechanical Systems and Signal Processing,* vol. 72, pp. 303-315, 2016.

[26] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, Cambridge, Massachusetts, USA: MIT press, 2016.

[27] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[28] A. Voulodimos, N. Doulamis, A. Doulamis and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience,* vol. 2018, no. 7068349, pp. 1-13, 2018.

[29] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine,* vol. 13, no. 3, pp. 55-75, 2018.

[30] A. E. Sallab, M. Abdou, E. Perot and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging,* vol. 2017, no. 19, pp. 70-76, 2017.

[31] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk and J. Faubert, "Deep learning-based electroencephalography analysis: A systematic review," *Journal of Neural Engineering,* vol. 16, no. 051001, pp. 1-33, 2019.

[32] Y. Chen, M. Rao, K. Feng and M. J. Zuo, "Physics-Informed LSTM hyperparameters selection for gearbox fault detection," *Mechanical Systems and Signal Processing,* vol. 171, p. 108907, 2022.

[33] C. Hu, P. Wang, K. Goebel, B. D. Youn, D. Howey, Z. Peng and D. Wang, "Physics-informed machine learning enabling fault feature extraction and robust failure prognosis,"

21 10 2021. [Online]. Available: https://www.sciencedirect.com/journal/mechanical-systems-and-signal-processing/special-issue/10MTCZR08FP. [Accessed 25 5 2022].

[34] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics,* vol. 3, no. 6, pp. 422-440, 2021.

[35] G. A., Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems, Sebastopol, California, USA: O'Reilly Media, 2019.

[36] M. Yu and S. Shiliang, "Policy-based reinforcement learning for time series anomaly detection," *Engineering Applications of Artificial Intelligence,* vol. 95, p. 103919, 2020.

[37] D. Kozjek and A. Malus, "Multi-objective adjustment of remaining useful life predictions based on reinforcement learning," *Procedia CIRP,* vol. 93, pp. 425-430, 2020.

[38] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li and A. K. Nandi, "Applications of machine learning to machine fault diagnosis: A review and roadmap," *Mechanical Systems and Signal Processing,* vol. 138, p. 106587, 2020.

[39] H. Cao, H. Shao, X. Zhong, Q. Deng, X. Yang and J. Xuan, "Unsupervised domain-share CNN for machine fault transfer diagnosis from steady speeds to time-varying speeds," *Journal of Manufacturing Systems,* vol. 62, pp. 186-198, 2022.

[40] G. Michau and O. Fink, "Unsupervised fault detection in varying operating conditions," in *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, San Francisco, California, USA, 2019.

[41] Y. Xu, Y. Sun, X. Liu and Y. Zheng, "A digital-twin-assisted fault diagnosis using deep transfer learning," *IEEE Access,* vol. 7, pp. 19990-19999, 2019.

[42] C. Li, S. Zhang, Y. Qin and E. Estupinan, "A systematic review of deep transfer learning for machinery fault diagnosis," *Neurocomputing,* vol. 407, pp. 121-135, 2020.

[43] W. Li, R. Huang, J. Li, Y. Liao, Z. Chen, G. He, R. Yan and K. Gryllias, "A perspective survey on deep transfer learning for fault diagnosis in industrial scenarios: Theories, applications and challenges," *Mechanical Systems and Signal Processing,* vol. 167, p. 108487, 2022.

[44] W. Yang, Z. Peng, K. Wei and W. Tian, "Structural health monitoring of composite wind turbine blades: Challenges, issues and potential solutions," *IET Renewable Power Generation,* vol. 11, no. 4, pp. 411-416, 2017.

[45] A. P. J. G. Prisacaru, M. B. Jeronimo, B. Han and G. Q. Zhang, "Prognostics and health monitoring of electronic system: A review," in *18th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*, Dresden, Germany, 2017.

[46] R. Bigret, "Rotating machinery essential features," *Encyclopedia of Vibration,* pp. 1064-1069, 2001.

[47] Q. Lv, X. Yu, H. Ma, J. Ye, W. Wu and X. Wang, "Applications of machine learning to reciprocating compressor fault diagnosis: A review," *Processes,* vol. 9, no. 6, p. 909, 2021.

[48] J. P. Salameh, S. Cauet, E. Etien, A. Sakout and L. Rambault, "Gearbox condition monitoring in wind turbines: A review," *Mechanical Systems and Signal Processing,* vol. 111, pp. 251-264, 2018.

[49] M. O. Abdalla and T. M. Al-Jarrah, "Fuzzy logic control of an electrical traction elevator," *JJMIE: Jordan Journal of Mechanical and Industrial Engineering,* pp. 97-106, 2011.

[50] S. Martin-del-Campo, F. Sandin and D. Strömbergsson, "Dictionary learning approach to monitoring of wind turbine drivetrain bearings," *arXiv preprint arXiv,* no. 1902.01426, 2019.

[51] J. Lin and M. Zhao, "A review and strategy for the diagnosis of speed-varying machinery," in *2014 International Conference on Prognostics and Health Management*, 2014.

[52] X. Yang, P. Zhou, M. J. Zuo and Z. Tian, "Normalization of gearbox vibration signal for tooth crack diagnosis under variable speed conditions," *Quality and Reliability Engineering International,* pp. 1-27, 2021.

[53] M. Pineda-Sanchez, M. Riera-Guasp, J. A. Antonino-Daviu, J. Roger-Folch, J. Perez-Cruz and R. Puche-Panadero, "Instantaneous frequency of the left sideband harmonic during the start-up transient: A new method for diagnosis of broken bars," *IEEE Transactions on Industrial Electronics,* vol. 56, no. 11, pp. 4557-4570, 2009.

[54] Y. Li, F. Gu, G. Harris, A. Ball, N. Bennett and K. Travis, "The measurement of instantaneous angular speed," *Mechanical Systems and Signal Processing,* vol. 19, no. 4, pp. 786-805, 2005.

[55] C. Peeters, Q. Leclère, J. Antoni, P. Lindahl, J. Donnal, S. Leeb and J. Helsen, "Review and comparison of tacholess instantaneous speed estimation methods on experimental vibration data," *Mechanical Systems and Signal Processing,* vol. 129, pp. 407-436, 2019.

[56] K. Dziedziech, A. Jablonski and Z. Dworakowski, "A novel method for speed recovery from vibration signal under highly non-stationary conditions," *Measurement,* vol. 128, pp. 13-22, 2018.

[57] F. Bonnardot, M. El Badaoui, R. B. Randall, J. Daniere and F. Guillet, "Use of the acceleration signal of a gearbox in order to perform angular resampling (with limited speed fluctuation)," *Mechanical Systems and Signal Processing,* vol. 19, no. 4, pp. 766-785, 2005.

[58] F. Combet and L. Gelman, "An automated methodology for performing time synchronous averaging of a gearbox signal without speed sensor," *Mechanical systems and signal processing,* vol. 21, no. 6, pp. 2590-2606, 2007.

[59] M. D. Coats and R. B. Randal, "Order-Tracking with and without a tacho signal for gear fault diagnostics," in *2012 Australian Acoustical Society Conference*, Fremantle, Australia, 2012.

[60] J. Urbanek, T. Barszcz and J. Antoni, "A two-step procedure for estimation of instantaneous rotational speed with large fluctuations," *Mechanical Systems and Signal Processing,* vol. 38, no. 1, pp. 96-102, 2013.

[61] A. Boudraa, J. Cexus, F. Salzenstein and L. Guillon, "IF estimation using empirical mode decomposition and nonlinear Teager energy operator," in *First International Symposium on Control, Communications and Signal Processing*, Hammamet, Tunisia, 2004.

[62] M. Feldman, "Time-varying vibration decomposition and analysis based on the Hilbert transform," *Journal of Sound and Vibration,* vol. 295, pp. 518-530, 2006.

[63] R. Randall and W. Smith, "Use of the Teager Kaiser Energy Operator to estimate machine speed," in *Proceedings of the European Conference of the PHM Society*, Bilbao, Bizkaia, Spain, 2016.

[64] J. Urbanek, T. Barszcz, N. Sawalhi and R. B. Randall, "Comparison of amplitude-based and phase-based method for speed tracking in application to wind turbines," *Metrology and measurement systems,* vol. 18, no. 2, pp. 295-303, 2011.

[65] Z. K. Peng, G. Meng, F. L. Chu, Z.-Q. Lang, W. M. Zhang and Y. Yang, "Polynomial chirplet transform with application to instantaneous frequency estimation," *IEEE Transactions on Instrumentation and Measurement ,* vol. 60, no. 9, pp. 3222-3229, 2011.

[66] K. C. Gryllias and I. A. Antoniadis, "Estimation of the instantaneous rotation speed using complex shifted Morlet wavelets," *Mechanical Systems and Signal Processing,* vol. 38, no. 1, pp. 78-95, 2013.

[67] S. Schmidt, P. Heyns and J. De Villiers, "A tacholess order tracking methodology based on a probabilistic approach to incorporate angular acceleration information into the maxima tracking process," *Mechanical Systems and Signal Processing,* vol. 100, pp. 630-646, 2018.

[68] D. Iatsenko, P. McClintock and A. Stefanovska, "Extraction of instantaneous frequencies from ridges in time–frequency representations of signals," *Signal Processing,* vol. 125, pp. 290-303, 2016.

[69] S. Lu, R. Yan, Y. Liu and Q. Wang, "Tacholess speed estimation in order tracking: A review with application to rotating machine fault diagnosis," *IEEE Transactions on Instrumentation and Measurement,* vol. 68, no. 7, pp. 2315-2332, 2019.

[70] H. Yousuf, M. Lahzi, S. A. Salloum and K. Shaalan, "A systematic review on sequence-to-sequence learning with neural network and its models," *International Journal of Electrical & Computer Engineering,* vol. 11, no. 3, pp. 2315-2326, 2021.

[71] Z. C. Lipton, J. Berkowitz and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv,* no. 1506.00019, pp. 1-37, 2015.

[72] R. Prabhavalkar, K. Rao, T. Sainath, B. Li, L. Johnson and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition".

[73] I. Sutskever, O. Vinyals and Q. V. Le, "equence to sequence learning with neural networks," *Advances in Neural Information Processing Systems,* vol. 27, pp. 1-9, 2014.

[74] C. Saiprasert, T. Pholprasit and S. Thajchayapong, "Detection of driving events using sensory data on smartphone," *International Journal of Intelligent Transportation Systems Research,* vol. 15, no. 1, pp. 17-28, 2017.

[75] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya and M. C. González, "Safe driving using mobile phones," *IEEE Transactions on Intelligent Transportation Systems,* vol. 13, no. 3, pp. 1462-1468, 2012.

[76] S. Sattar, S. Li and M. Chapman, "Road surface monitoring using smartphone sensors: A review," *Sensors,* vol. 18, no. 11, pp. 1-21, 2018.

[77] L. T. Hsu, Y. G. Gu and S. Kamijo, "Intelligent viaduct recognition and driving altitude determination using GPS data," *IEEE Transactions on Intelligent Vehicles,* vol. 2, no. 3, pp. 175-184, 2017.

[78] Y. Gu, Q. Wang and S. Kamijo, "Intelligent driving data recorder in smartphone using deep neural network-based speedometer and scene understanding," *IEEE Sensors Journal,* vol. 19, no. 1, pp. 287-296, 2018.

[79] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane and G. Nenadic, "Machine learning methods for wind turbine condition monitoring: A review," *Renewable Energy,* vol. 133, pp. 620-635, 2019.

[80] M. A. Pimentel, D. A. Clifton, L. Clifton and L. Tarassenko, "A review of novelty detection," *Signal Processing,* vol. 99, pp. 215-249, 2014.

[81] H. Zhou, X. Huang, G. Wen, Z. Lei, S. Dong, P. Zhang and X. Chen, "Construction of health indicators for condition monitoring of rotating machinery: A review of the research," *Expert Systems with Applications,* vol. 203, p. 117297, 2022.

[82] K. K. Reddy, S. Sarkar, V. Venugopalan and M. Giering, "Anomaly detection and fault disambiguation in large flight data: a multi-modal deep auto-encoder approach," in *2016 Annual Conference of the Prognostics and Health Management Society*, Denver, Colorado, USA, 2016.

[83] E. V. F. Marchi, S. Squartini and B. Schuller, "Deep recurrent neural network-based autoencoders for acoustic novelty detection," *Computational Intelligence and Neuroscience ,* 2017.

[84] A. Chandra and R. Kala, "Regularised encoder-decoder architecture for anomaly detection in ECG time signals," in *IEEE Conference on Information and Communication Technology*, 2019.

[85] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv,* no. 1607.00148, 2016.

[86] S. Maya, K. Ueno and w. Nishika, "dLSTM: a new approach for anomaly detection using deep learning with delayed prediction," *International Journal of Data Science and Analytics,* vol. 8, no. 2, pp. 137-164, 2019.

[87] N. Dervilis, M. Choi, S. Taylor, R. Barthorpe, G. Park, C. Farrar and K. Worden, "On damage diagnosis for a wind turbine blade using pattern recognition," *Journal of sound and vibration,* vol. 33, no. 6, pp. 1833-1850, 2014.

[88] G. Michau, P. Thomas and O. Fink, "Deep feature learning network for fault detection and isolation," in *Annual Conference of the Prognostics and Health Management Society*, St. Petersburg, Florida, USA, 2017.

[89] G. Michau, Y. Hu, T. Palme and O. Fink, "Feature learning for fault detection in high-dimensional condition monitoring signals," *Proceedings of the Institution of Mechanical Engineers Part O: Journal of Risk and Reliability,* vol. 234, no. 1, pp. 104-115, 2020.

[90] W. Mao, J. Chen, X. Liang and X. Zhang, "A new online detection approach for rolling bearing incipient fault via self-adaptive deep feature matching," *IEEE Transactions on Instrumentation and Measurement,* vol. 69, no. 2, pp. 443-456, 2020.

[91] P. Chen, Y. Li, K. Wang, M. J. Zuo, P. S. Heyns and S. Baggeröhr, "A threshold self-setting condition monitoring scheme for wind turbine generator bearings based on deep convolutional generative adversarial networks," *Measurement,* vol. 167, p. 108234, 2021.

[92] S. Shmidt and P. S. Heyns, "Normalisation of the amplitude modulation caused by time-varying operating conditions for condition monitoring," *Measurement ,* vol. 149, p. 106964, 2020.

[93] K. R. Fyfe and E. D. S. Munck, "Analysis of computed order tracking," *Mechanical Systems and Signal Processing,* vol. 11, no. 2, pp. 187-205, 1997.

[94] M. Rao and M. J. Zuo, "A new strategy for rotating machinery fault diagnosis under varying speed conditions based on deep neural networks and order tracking," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, Florida, USA, 2018.

[95] X. Liang, F. Duan, I. Bennett and D. Mba, "A sparse autoencoder-based unsupervised scheme for pump fault detection and isolation," *Applied Sciences,* vol. 10, no. 19, p. 6789, 2020.

[96] G. Jiang, P. Xie, H. He and J. Yan, "Wind turbine fault detection using a denoising autoencoder with temporal information," *IEEE/ASME Transactions on Mechatronics,* vol. 23, no. 1, pp. 89-100, 2018.

[97] B. Luo, H. Wang, H. Liu, B. Li and F. Peng, "Early fault detection of machine tools based on deep learning and dynamic identification," *IEEE Transactions on Industrial Electronics,* vol. 66, no. 1, pp. 509-518, 2018.

[98] H. Zhao, H. Liu, W. Hu and X. Yan, "Anomaly detection and fault analysis of wind turbine components based on deep learning network," *Renewable Energy,* vol. 127, pp. 825-834, 2018.

[99] M. Cerrada, R.-V. Sánchez, C. Li, F. Pacheco, D. Cabrera, J. V. d. Oliveira and R. E. Vasquez, "A review on data-driven fault severity assessment in rolling bearings," *Mechanical Systems and Signal Processing,* vol. 99, pp. 169-196, 2018.

[100] T. Ince, S. Kiranyaz, L. Eren, M. Askar and M. Gabbouj, "Real-time motor fault detection by 1-D convolutional neural networks," *IEEE Transactions on Industrial Electronics,* vol. 63, no. 11, pp. 7067-7075, 2016.

[101] H. Shao, H. Jiang, H. Zhao and F. Wang, "A novel deep autoencoder feature learning method for rotating machinery fault diagnosis," *Mechanical Systems and Signal Processing,* vol. 95, pp. 187-204, 2017.

[102] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing,* vol. 234, pp. 11-26, 2017.

[103] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Paheding , M. S. Nasrin, H. Hasan, B. C. Van Essen, A. A. Awwal and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics ,* vol. 8, no. 3, p. 292, 2019.

[104] O. Fink, Q. Wang, M. Svensén, P. Dersin, W.-J. Lee and M. Ducoffe, "Potential, challenges and future directions for deep learning in prognostics and health management applications," *Engineering Applications of Artificial Intelligence,* vol. 92, no. 103678, pp. 1-15, 2020.

[105] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, R. V. d. Walle and S. V. Hoecke, "Convolutional neural network based fault detection for rotating machinery," *Journal of Sound and Vibration,* vol. 377, pp. 331-345, 2016.

[106] S. Ma, F. Chu and Q. Han, "Deep residual learning with demodulated time-frequency features for fault diagnosis of planetary gearbox under nonstationary running conditions," *Mechanical Systems and Signal Processing,* vol. 127, pp. 190-201, 2019.

[107] D. Wei, K. Wang, S. Heyns and M. J. Zuo, "Convolutional neural networks for fault diagnosis using rotating speed normalized vibration," in *International Conference on Condition Monitoring of Machinery in Non-Stationary Operation*, Santander, Spain, 2018.

[108] Y. Du, W. Aiming, S. Wang, B. He and G. Meng, "Fault diagnosis under variable working conditions based on STFT and transfer deep residual network," *Shock and Vibration ,* vol. 2020, p. 1274380 , 2020.

[109] D. Cabrera, F. Sancho, C. Li, M. Cerrada, R.-V. Sánchez, F. Pacheco and J. V. d. Oliveira, "Automatic feature extraction of time-series applied to fault severity assessment of helical

gearbox in stationary and non-stationary speed operation," *Applied Soft Computing,* vol. 58, pp. 53-64, 2017.

[110] Y. Han, B. Tang and L. Deng, "Multi-level wavelet packet fusion in dynamic ensemble convolutional neural network for fault diagnosis," *Measurement,* vol. 127, pp. 246-255, 2018.

[111] B. Yuan, C. Wang, F. Jiang, M. Long, P. S. Yu and Y. Liu, "WaveletFCNN: A deep time series classification model for wind turbine blade icing detection," *arXiv preprint arXiv,* no. 1902.05625, 2019.

[112] M. Zhao, M. Kang, B. Tang and M. Pecht, "Multiple wavelet coefficients fusion in deep residual networks for fault diagnosis," *IEEE Transactions on Industrial Electronics,* vol. 66, no. 6, pp. 4696-4796, 2019.

[113] Z. An, S. Li, J. Wang and X. Jiang, "A novel bearing intelligent fault diagnosis framework under time-varying working conditions using recurrent neural network," *ISA Transactions,* 2019.

[114] R. Liu, F. Wang, B. Yang and S. J. Qin, "Multi-scale kernel based residual convolutional neural network for motor fault diagnosis under non-stationary conditions," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 6, pp. 3797-3806, 2020.

[115] K. Weiss, T. M. Khoshgoftaar and D. Wang, "A survey of transfer learning," *Journal of Big data,* vol. 3, no. 1, pp. 1-40, 2016.

[116] A. Farahani, S. Voghoei, K. Rasheed and H. R. Arabnia, "A brief review of domain adaptation," *Advances in Data Science and Information Engineering,* pp. 877-894, 2021.

[117] H. Shao, M. X. Xia, G. Han, Y. Zhang and J. Wan, "Intelligent fault diagnosis of rotor-bearing system under varying working conditions with modified transfer convolutional neural network and thermal images," *IEEE Transactions on Industrial Informatics,* vol. 17, no. 5, pp. 3488-3496, 2020.

[118] L. Guo, Y. Lei, S. Xing, T. Yan and N. Li, "Deep convolutional transfer learning network: A new method for intelligent fault diagnosis of machines with unlabeled data," *IEEE Transactions on Industrial Electronics,* vol. 66, no. 9, pp. 7316-7325, 2019.

[119] T. Hastie, R. Tibshirani and J. Friedman, The elements of statistical learning: Data mining, inference, and prediction (2nd edition), New York, USA: Springer, 2009.

[120] S. S. Rao, Engineering Optimization Theory and Practice (5th edition), Hoboken, New Jersey, USA: John Wiley & Sons, 2019.

[121] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science,* vol. 313, no. 5786, pp. 504-507, 2006.

[122] K. You, M. Long, J. Wang and M. I. Jordan, "How does learning rate decay help modern neural networks," *arXiv preprint arXiv,* no. 1908.01878, pp. 1-15, 2019.

[123] Z. Wang, W. Yan and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International Joint Conference on Neural Networks*, Anchorage, Alaska, USA, 2017.

[124] K. He, X. Zhuang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, Nevada, USA, 2016.

[125] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *27th International Conference on Machine Learning*, Haifa, Israel, 2010.

[126] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[127] "Keras Tutorial," 2019. [Online]. Available: https://www.tutorialspoint.com/keras/index.htm. [Accessed 20 6 2022].

[128] M. Rao, Q. Li, D. Wei and M. J. Zuo, "A deep bi-directional long short-term memory model for automatic rotating speed extraction from raw vibration signals," *Measurement,* no. 107719, 2020.

[129] M. Rao, Q. Li, D. Wei and M. J. Zuo, "Virtual rotating speed meter: extracting machinery rotating speed from vibration signals based on deep learning and transfer learning," in *2020 Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM)*, Vancouver, British Columbia, Canada, 2020.

[130] Q. Leclère, H. André and J. Antoni, "A multi-order probabilistic approach for Instantaneous Angular Speed tracking debriefing of the CMMNO14 ′ diagnosis contest," *Mechanical Systems and Signal Processing,* vol. 81, pp. 375-386, 2016.

[131] L. Huang, Y. Xu, W. Jiang, L. Xue, W. Hu and L. Yi, "Performance and complexity analysis of conventional and deep learning equalizers for the high-speed IMDD PON," *Journal of Lightwave Technology,* vol. 40, no. 14, pp. 4528-38, 2022.

[132] C. Laranjeira, A. Lacerda and E. R. Nascimento, "On modeling context from objects with a long short-term memory for indoor scene recognition," in *32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, Rio de Janeiro, Brazil, 2019.

[133] R. Buoy, N. Taing and S. Kor, "Khmer word segmentation using BiLSTM networks," in *4th Regional Conference on OCR and NLP for ASEAN Languages*, Phnom Penh, Cambodia, 2020.

[134] J. Jiang, S. Zou, Y. Sun and S. Zhang, "GL-BLSTM: a novel structure of bidirectional long-short term memory for disulfide bonding state prediction," *arXiv preprint arXiv,* no. 1808.03745, pp. 1-9, 2018.

[135] J. Feng, Y. Lei, J. Lin, X. Zhou and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," *Mechanical Systems and Signal Processing,* vol. 72, pp. 303-315, 2016.

[136] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent and S. Bengio, "Why does unsupervised pre-training help deep learning," *Journal of Machine Learning Research,* vol. 11, pp. 625-660, 2010.

[137] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv,* no. 1810.04805, 2018.

[138] R. B. Kurdikeri and A. B. Raju, "Comparative study of short-term wind speed forecasting techniques using artificial neural networks," in *2018 IEEE International Conference on Current Trends toward Converging Technologies*, Coimbatore, India, 2018.

[139] Y. Zhang, Y. Liu, Z. Liu and W. Liang, "Developing a Long Short-Term Memory-based signal processing method for Coriolis mass flowmeter," *Measurement,* vol. 14, p. 106896, 2019.

[140] T. Carneiro, R. Da Nóbrega, T. Nepomuceno, G. Bian, V. De Albuquerque and P. Reboucas Filho, "Performance analysis of google colaboratory as a tool for accelerating deep learning applications," *IEEE Access,* vol. 6, pp. 61677-61685, 2018.

[141] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv,* no. 1412.6980, 2014.

[142] S. J. Hunter, "The exponentially weighted moving average," *Journal of quality technology,* vol. 18, no. 4, pp. 203-210, 1986.

[143] C. Peng, K. Wang, Z. J. Ming and W. Dongdong, "An ameliorated synchroextracting transform based on upgraded local instantaneous frequency approximation," *Measurement,* vol. 148, pp. 1-15, 2019.

[144] Y. Chen, X. Yang, M. Rao and M. J. Zuo, "Experiment design and data collection on a fixed-axis gearbox under time-varying operation conditions," Department of Mechanical Engineering, University of Alberta, Edmonton, Canada, 2018.

[145] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, Hershey, Pennsylvania, USA, IGI Global, 2010, pp. 242-264.

[146] M. Rao, M. J. Zuo and Z. Tian, "A speed normalized autoencoder for rotating machinery fault detection under varying speed conditions," *Mechanical Systems and Signal Processing,* vol. 189, p. 110109, 2023.

[147] H. Chen, H. Liu, X. Chu, Q. Liu and D. Xue, "Anomaly detection and critical SCADA parameters identification for wind turbines based on LSTM-AE neural network," *Renewable Energy,* vol. 172, pp. 829-840, 2021.

[148] J. Yu, X. Liu and L. Ye, "Convolutional long short-term memory autoencoder-based feature learning for fault detection in industrial processes," *IEEE Transactions on Instrumentation and Measurement,* vol. 70, pp. 1-15, 2020.

[149] D. Jana, J. Patil, S. Kerkal, S. Nagarajaiah and L. Duenas-Osorio, "CNN and convolutional autoencoder based real-time sensor fault detection, localization, and correction," *Mechanical Systems and Signal Processing,* vol. 169, p. 108723, 2022.

[150] P. Spyridon and Y. S. Boutalis, "Generative adversarial networks for unsupervised fault detection," in *European Control Conference (ECC)*, Limassol, Cyprus, 2018.

[151] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv,* no. 1901.03407, 2019.

[152] Y. Hu, T. and O. Fink, "Fault detection based on signal reconstruction with auto-associative extreme learning machines," *Engineering Applications of Artificial Intelligence,* vol. 57, pp. 105-117, 2017.

[153] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE,* vol. 2, no. 1, pp. 1-18, 2018.

[154] T. Han, C. Liu, W. Yang and D. Jiang, "A novel adversarial learning framework in deep convolutional neural network for intelligent diagnosis of mechanical faults," *Knowledge-Based Systems,* vol. 165, pp. 474-487, 2019.

[155] S. Ahmad, A. Lavin, S. Purdy and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing,* vol. 262, pp. 134-147, 2017.

[156] J. Ren, X. Ma, C. Xu, H. Zhao and S. Yi, "Havana: Hierarchical and variation-normalized autoencoder for person re-identification," *arXiv preprint arXiv,* no. 2101.02568, 2021.

[157] Z. Rong, Q. Tan, L. Cao, L. Zhang, K. Deng, Y. Huang, Z. Zhu, Z. Li and K. Li, "NormAE: Deep adversarial learning model to remove batch effects in liquid chromatography mass spectrometry-based metabolomics data," *Analytical Chemistry,* vol. 92, no. 7, pp. 5082-90, 2020.

[158] S. Ariafar, J. Coll-Font and D. D. J. Brooks, "ADMMBO: Bayesian optimization with unknown constraints using ADMM," *Journal of Machine Learning Research,* vol. 20, pp. 1-26, 2019.

[159] H. Liu, K. Simonyan and Y. Yang, "DARTS: Differentiable architecture search," *arXiv:1806.09055,* 2019.

[160] J. N. Mandrekar, "Receiver operating characteristic curve in diagnostic test assessment," *Journal of Thoracic Oncology,* vol. 5, no. 9, pp. 1315-1316, 2010.

[161] Y. Chen, X. Liang and M. J. Zuo, "Sparse time series modeling of the baseline vibration from a gearbox under time-varying speed condition," *Mechanical Systems and Signal Processing,* vol. 134, p. 106342, 2019.

[162] Y. Kong, M. Rao, H. Fu, Q. Han, F. Chu and M. Zuo, "Experiment description and condition monitoring data collection on a planetary gearbox considering constant and varying speed conditions," Department of Mechanical Engineering, Tsinghua University, Beijing, China, 2021.

[163] H. Huang and N. Baddour, "Bearing vibration data collected under time-varying rotational speed conditions," *Data in Brief,* vol. 21, pp. 1745-1749, 2018.

[164] M. Rao, M. J. Zuo and Z. Tian, "Speed adaptive gate: Improving fault classification accuracy of deep learning models for rotating machinery under varying speed conditions," *Measurement,* vol. Submitted, 2023.

[165] M. Rao, M. J. Zuo, Z. Tian and D. Qin, "Speed adaptive gates for deep learning-based fault classification of rotating machinery under varying speed conditions: Comparison of implementations," in *IEEE Global Reliability & Prognostics and System Health Management Conference*, Yantai, China, 2022.

[166] M. Zhao, M. Kang, B. Tang and M. Pecht, "Deep residual networks with dynamically weighted wavelet coefficients for fault diagnosis of planetary gearboxes," *IEEE Transactions on Industrial Electronics,* vol. 65, no. 5, pp. 4290-4300, 2017.

[167] A. Ali, S. M. Shamsuddin and A. L. Ralescu, "Classification with class imbalance problem: A review," *International Journal of Advances in Soft Computing and its Applications,* vol. 7, no. 3, pp. 176-204, 2015.

[168] R. K. Srivastava, K. Greff and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:,* no. 1505.00387, pp. 1-6, 2015.

[169] F. Jia, Y. Lei, N. Lu and S. Xing, "Deep normalized convolutional neural network for imbalanced fault classification of machinery and its understanding via visualization," *Mechanical Systems and Signal Processing,* vol. 110, pp. 349-367, 2018.

[170] P. Chen, Y. Li, K. Wang and M. J. Zuo, " A novel knowledge transfer network with fluctuating operational condition adaptation for bearing fault pattern recognition," *Measurement,* vol. 158, no. 107739, pp. 1-12, 2020.

[171] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Progress in Artificial Intelligence,* vol. 4, no. 221-232, p. 5, 2016.

[172] H. Guo, Y. Li, J. Shang, M. Gu, H. Yuanyue and B. Gong, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications,* vol. 73, pp. 220-239, 2017.

[173] R. B. Randall, "Vibration-based diagnostics of gearboxes under variable speed and load conditions," *Meccanica,* vol. 51, no. 12, pp. 3227-3239, 2016.

[174] I. Antoniadou, G. Manson, W. J. Staszewski, T. Barszcz and K. Worden, "A time–frequency analysis approach for condition monitoring of a wind turbine gearbox under varying load conditions," *Mechanical Systems and Signal Processing,* vol. 64, pp. 188-216, 2015.

[175] M. S. Feki, F. Chaari, M. S. Abbes, F. Viadero, A. F. Rincon and M. Haddar, "Dynamic analysis of planetary gear transmission under time varying loading conditions," Heidelberg, Germany, Springer, 2013, pp. 311-318.

[176] G. James, H. D. T. Witten and R. Tibshirani, An introduction to statistical learning, Heidelberg, Germany: Springer, 2013.

[177] Y. Lei, N. Li, L. Guo, N. Li, T. Yan and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mechanical Systems and Signal Processing,* vol. 104, pp. 799-234, 2018.

[178] R. Gravina, P. Alinai, H. Ghasemzadeh and G. Fortino, "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges," *Information Fusion,* vol. 35, pp. 68-80, 2017.

[179] T. Li, Z. Zhao, C. Sun, R. Yan and X. Chen, "Adaptive channel weighted CNN with multisensor fusion for condition monitoring of helicopter transmission system," *IEEE Sensors Journal,* vol. 20, no. 15, pp. 8364-8363, 2020.