School of Electrical Engineering, Computing and Mathematical Sciences

Advanced Deep Learning Methods for Vibration-based Structural Damage Identification

RUHUA WANG
0000-0001-5798-8118

This thesis is presented for the degree of
Doctor of Philosophy
of
Curtin University

May 2021

Declaration

I hereby declare that, to the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made. This thesis contains no material which has been accepted for the award of any other degree or diploma in any university. I have obtained permission from the copyright owners to use any third-party copyright material reproduced in the thesis, or to use any of my own published work in which the copyright is held by another party.

Signature: _____ Date: _____

Abstract

Vibration-based structural damage identification has been a challenging task in structural health monitoring (SHM). The main difficulty lies on the reliable correlation between the measured vibration characteristics and the damage states (*e.g.*, stiffness reductions) of structures. Such states can ideally indicate the presence, locations, and severities of structural damages. The damage identification procedure can hence be considered as a feature extraction process from the input measurement, mapping the selected features to damage states.

Some traditional machine learning methods, such as Principal Component Analysis, Support Vector Machine, and shallow Artificial Neural Networks, have been widely used in SHM during the past decades. However, such traditional methods are insufficient to learn complex models, especially when the dimension of data is very high. It should be noted that the vibration signals are usually high-dimensional, noise-contaminated, and complex (*e.g.*, in multiple scales). Deep learning, which often refers to deep neural networks, can be used to overcome these obstacles. It has achieved great success in a variety of artificial intelligence research areas and recently has also been extensively applied to vibration-based structural damage identification.

In this thesis, we propose advanced deep learning frameworks for vibration-based damage identification applications. Vibration signals such as frequencies, mode shapes, and acceleration responses are utilized as the input while the structural properties such as stiffness parameters are utilized as the output. By capturing the underlying relationship between the input and output, one can perform damage identification. As discussed above, the vibration data is usually very complex. Therefore, we investigate how to achieve effective and efficient damage identification by using several advanced deep learning methods. We also investigate a novel type of neural networks known as invertible neural networks, which might be used to improve the robustness and security of deep learning-based SHM frameworks. Next, the major tasks of this thesis are summarised below.

Firstly, we propose a parallel sparse autoencoder framework for structural damage identification, which can deal with multi-scale datasets. This framework consists of two main components: a parallel architecture-based dimensionality reduction component followed by a

relationship learning component. We emphasize the importance of processing the multi-scale vibration data separately in the dimensionality reduction stage. Experiments are conducted on both the clean dataset and the noisy dataset to evaluate the performance of the proposed method. This proposed framework significantly improves the effectiveness and robustness of structural damage identification in comparison with the state-of-the-art approach.

Next, we investigate convolutional neural network-based approaches, which are capable of constructing end-to-end learning methods for vibration-based structural damage identification. We first propose a deep residual network framework which is composed of purely residual blocks operating as feature extractors and a fully connected layer as a regressor. It learns the damage-related features from the vibration characteristics, such as mode shapes, and maps them into the damage index labels, *e.g.*, stiffness reductions of structures. Extensive performance evaluations are conducted with both numerical and experimental studies. The comparison between the proposed approach and the state-of-the-art models, including a sparse autoencoder neural network, a convolutional neural network, is conducted. The proposed deep residual network framework consistently outperforms the compared approaches.

After that, we propose a densely connected convolutional network framework that implements dense connectivity in the convolutional neural network architecture, which fits well for the study using time-domain responses, *e.g.*, acceleration. Both low-level and high-level features are learned and reused during training. It not only strengthens feature propagation through the network but also substantially reduces the number of parameters. The performance of the proposed approach is evaluated through both numerical and experimental verifications. The results demonstrate that the damage localization and quantification are achieved with better performance, in comparison with the deep residual network framework.

Last but not least, we investigate the invertibility of residual networks. We first propose a novel fixed-point algorithm that can be used to invert residual units under weak weight constraints. Then, we propose a sufficient and necessary condition under which the residual block is invertible. Several experiments have been conducted using both image and SHM datasets to evaluate the discrimination performance of the proposed invertible residual networks. A promising direction of using the proposed invertible residual networks to increase the robustness and security of deep learning-based SHM frameworks is discussed.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisors Dr. Senjian An, Prof. Ling Li, Prof. Wanquan Liu, and A/Prof. Jun Li, for their continuous support during my Ph.D. research studies, both mentally and technically. As a supervisor, they are always available to talk with me and put their students' interest at the first place. I still remember when I first came here and Wanquan told me that I should enjoy this long journey in Perth. Even at difficult times, I was trying my best to make it. I did have many enjoyable and memorable days and nights working with Senjian before the submission deadlines, which I have learned a great deal from him. I have also learned a lot from Jun, who has given me many practical suggestions and helped me get through many hurdles in my structural health monitoring research projects. I am appreciated much to Ling, who always encourages me, strengthens my confidence and guides me in the right direction, and keeps me on track, for those tough days when I felt miserable about my future. It has been such a great honor to be a student of my supervisors. Their wisdom, academic insight, and rigorous way of thinking have greatly influenced me in different ways and truly enlightened me to become a better researcher.

I would like to express a special thanks to the School of Electrical Engineering, Computing and Mathematical Sciences and Prof. Hong Hao and Prof. Jun Li from the School of Civil and Mechanical Engineering for providing sponsorship to my Ph.D. study program. My sincere thanks also go to their excellent and professional guidance in the Civil Engineering field.

I would like to thank all my fellow research students and researchers (Antoni Liang, Nadith Pathirage, Qilin Li, Xianchao Xiu, Shichu chen, Lu Tan, Hongyan Feng, Huizhu Pan, Bradley Ezard, Chencho, Xi Zhang, Jiayi Zhu, Yanda Shao and Jie Liu) for many discussions and inspirations. I am very grateful to Bradley for his patience and kindness, who helped me out many times to fix the server issues when I was stuck running experiments on the GPUs. Besides, thanks to Qilin and Yanda, it has been a great memory of those traveling and fishing times with them all over WA.

I am much indebted to my best friend, Dr. Wing Yan Ho, who is also a mentor to me. Thanks for her companionship, encouragement, and great support through these years, and fills my life with laughter and warm memories that I treasure very much.

Finally yet importantly, my most profound appreciation is given to my family. Thanks to my parents, Mr. Xinsheng Wang and Mrs. Guizhen Yuan, for their kind understanding and their warmest support. Thanks to my twin brother, Mr. Zehua Wang, for his encouragement and those game nights for relieving my pressure from my studies. Thanks to my lovely puppy, Xiao Bao, who always brings happiness to us, I am missing her.

Publications

This is a list of works that have been published over the course of the author's PhD Degree in chronological order:

- **Wang, R.**, An, S., Liu, W., & Li, L. (2021). Invertible Residual Blocks in Deep Learning Networks. *Submitted to: IEEE Transactions on Neural Networks and Learning Systems.*

- **Wang, R.**, Li, J., An, S., Hao, H., Liu, W., & Li, L. (2021). Densely connected convolutional networks for vibration based structural damage identification. *Engineering Structures*, 245, 112871.

- **Wang, R.**, An, S., Liu, W., & Li, L. (2021). Fixed-point algorithms for inverse of residual rectifier neural networks. *Mathematical Foundations of Computing*, 4(1), p.31.

- **Wang, R.**, Chencho, An, S., Li, J., Li, L., Hao, H., & Liu, W. (2020). Deep residual network framework for structural health monitoring. *Structural Health Monitoring*, p.1475921720918378.

- **Wang, R.**, Li, L., & Li, J. (2018). A novel parallel auto-encoder framework for multi-scale data in civil structural health monitoring. *Algorithms*, 11(8), p.112.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Structural health monitoring (SHM) [30] refers to the process of implementing a damage identification strategy to assess and predict structural performance under operational conditions. Civil infrastructures such as buildings, bridges, roads and tunnels may continuously accumulate damages during their service life caused by the environmental, operational and human-induced influences. Such damages like material deterioration, long-term fatigues, corrosion and degradation will adversely affect the current or future performance of structures. The main objective of SHM is to detect, locate and quantify the damages of civil infrastructures at an early stage, and to make decisions for the maintenance of the monitored structures.

Structural damage identification is a key component of SHM with numerous research efforts having been made using vibration-based methods [11, 29, 58, 64, 76, 106, 107]. As the vibration characteristics of the structure contain useful information of its state, the changes in the information between the intact and damaged states can be used to quantify structural damages. In other words, by extracting and analyzing damage-sensitive features from the measured vibration characteristics, one can determine the current state of structural health. Therefore, the vibration-based damage identification problem in SHM can be formulated as a pattern recognition problem.

Pattern recognition (PR) [31] covers a wide variety of activities and humans are particularly good at most of them. For example, many of us can distinguish dogs from cats at a glance, identify different species of fruits, recognize voices over a phone call even when the connection is unstable, etc. Similar tasks have been brought to us with the rapid develop-

ment of science, technology, and business, such as diagnosing diseases, detecting anomalies of infrastructures, identifying risks of investments, and so on. Although humans can do some of the tasks quite well, it is of great interest and demand to develop algorithms to perform such tasks in more accurate, efficient, and cheaper ways. PR is the discipline of developing such algorithms that can automatically identify patterns in data, and then utilize these patterns for decision-making tasks. Traditionally, SHM enables civil engineers to conduct field measurement and monitor the structural conditions, which is often subjective, inefficient and risky [17]. Therefore, covert such engineering goals to pattern recognition tasks would be beneficial for SHM.

Recently, machine learning (ML) algorithms gain much popularity in the research domain of PR owing to the increased availability of big data and computational power. Machine learning [74] refers to the study of computational models and algorithms that can automatically learn from experience without being explicitly programmed. It has become a main branch of artificial intelligence (AI), and has been applied in a very broad set of practical applications, including computer vision, natural language processing, speech processing, and structural health monitoring.

Machine learning was originated in the computer science field and is now closely related to pattern recognition. In ML, PR can be seen as the assignment of a label to a given input value, which can be categorized into different tasks. Classification is one of the typical PR tasks that assigns objects to a set of given classes. For example, separating "red apples" from "oranges" based on their distinctive features like the shape and the color is a classification problem. Another type of task, known as regression, assigns a real-valued output, instead of a class label, to each input. For example, predicting house prices based on its past information is a regression problem. Machine learning can be categorized into supervised learning, unsupervised learning, and semi-supervised learning based on the availability of labeled data. Here, both classification and regression are in the category of supervised learning. Unsupervised learning such as dimensionality reduction, clustering, and ranking are also widely studied in ML.

Some traditional ML algorithms such as Principle Component Analysis (PCA) [104], Support Vector Machine (SVM) [82], and shallow Artificial Neural Networks (ANNs) [42] are insufficient to learn complex models when the dimensionality of data is very high. This phenomenon is known as "the curse of dimensionality" [34]. Besides, the civil data, *e.g.*,

the measured vibration characteristics of structures, often include various types of noises, thereby requiring highly complicated models to learn the robust representations. Deep learning techniques were designed to overcome these obstacles and has become the dominant methods in machine learning in recent years. Deep learning, often refers to deep neural networks, extract higher-level features progressively from the raw input through multiple layers. For instance, in image classification, lower layers may identify edges while higher layers may identify objects as combinations of these edges. Deep learning has achieved great success in a variety of AI research areas and recently has also been extensively applied to vibration-based structural damage identification. Pathirage *et al.* [85, 86] developed deep learning-based autoencoder models to quantify the damage of structures using vibration characteristics, such as natural frequencies and mode shapes. This kind of damage identification problems are generally regression problems which learn the underlying relationship between the measured vibration signals and the ground truth, *e.g.*, a stiffness reduction vector that indicates the severity and location of damages.

In this thesis, we focus on pattern recognition for structural health monitoring problems, in which the data is usually high-dimensional and noise-contaminated, and sometimes is in multiple scales of various physical meanings. A promising direction for improving the robustness and security of deep learning SHM frameworks is also explored and discussed.

1.1 Research Objectives

This thesis investigates methods to capture the underlying relationship between the measured vibration signals, and the structural damage indices, *e.g.*, stiffness reductions. By exploiting recent advanced deep learning networks, SHM frameworks can be bulit that facilitate efficient and effective identification of structural damages. Besides, these frameworks are flexible to deal with high-dimensional, noise-contaminated, and multi-scale datasets. Furthermore, the conditions under which a residual block is invertible are investigated. An invertible residual network is developed that can potentially be used to make the SHM frameworks more secure and robust to adversarial attacks. After conducting in-depth studies on the literature review, several research gaps and drawbacks of existing methods are observed. We present our research objectives separately to address each of these gaps as below.

- Deep neural networks, such as deep autoencoder, has been utilized successfully for many supervised learning tasks. It was observed the greedy layer-wise unsupervised pre-training can yield substantial improvements for supervised learning tasks, which was seen as a breakthrough in deep learning in 2006 [13, 40, 89]. Pathirage *et al.* [85, 86] developed deep autoencoders based frameworks that target vibration-based structural damage identification applications, carrying out dimensionality reduction and relationship learning via two stages. However, these proposed models are basically processing the entire input data together which include different types of features, *e.g.*, natural frequencies and mode shapes. This will increase the difficulty in learning a robust framework due to several factors: 1) The natural frequencies and mode shapes are in different magnitude scales, *e.g.*, 7.6 and 0.9, and of different physical meanings. It is not logical to normalize them in the same scale and process them as a whole vector; 2) Each mode shape is associated with one frequency, and different mode shapes are unrelated to each other. It is better to deal with them separately in the dimensionality reduction stage; 3) The SHM datasets are vulnerable to diverse types of noises, such as measurement noises and uncertainties. These noises may not be well reduced if the input is formed by all frequencies and mode shapes as one high dimensional vector. To tackle these issues, we propose a parallel architecture based framework that can learn a robust representation of the input.

- The deep autoencoder-based approach mentioned above can be used for regression tasks with inputs of multi-scale elements. However, the training of deep autoencoders heavily relies on layer-wise pretraining. Also, the optimizations of two components (*i.e.*, the dimensionality reduction and the relationship learning) in the approach are separate. To develop more effective and efficient approaches, we first investigate the convolutional neural networks (CNNs), which leverages three key ideas: sparse interactions, parameter sharing, and equivariant representations. It has been used successfully in image classification to extract features from high-dimensional data. However, its performance may degrade due to vanishing gradients. Deep residual networks is proposed to solve this issue by introducing skip connections in the CNNs. We exploit the advantages of CNNs and deep residual networks to build our own framework for structural damage identification applications which often involve high-dimensional and noise-contaminated data.

- Modal information such as natural frequencies and mode shapes are extensively investigated to identify structural damages in recent studies. However, a sufficient number of sensor measurements are usually required for covering the whole structure and a large number of channels in data acquisition systems are required accordingly to obtain such modal information. This may not be practical in the rough environment for large-scale structures. On the contrary, only a certain number of sensors for measuring acceleration responses are required for those time-domain methods for structural damage identification. Also, sufficient data points can be sampled using fewer sensors. More importantly, it is straightforward to use raw time-domain responses without calculating the mode shapes of structures. Thus, it is of practical values to directly employ time-domain responses for damage identification of structures. On the other hand, the sensor data always contain significant effects of noise, thereby making it not easy to interpret vibration response information. To find an efficient and effective approach to learn the unknown relationship between sensor measurements and damage patterns, we develop a densely connected convolutional network framework that fits well to the time-domain response for structural damage identification.

- The robustness and security of SHM frameworks against adversarial attacks are of the utmost importance if they are to be implemented to real-world SHM applications. A nice property of invertible neural networks is that it defines a bijective mapping from the input domain to the output domain and one can recover the input from the output. With this property, we can investigate the difference between legitimate and adversarial examples. This may be used to improve the robustness and security of SHM frameworks against adversarial examples. Many state-of-the-art works on invertible neural networks focus on the invertibility of residual blocks. Thus, we first investigate the invertible conditions of residual networks. The invertible densely connected convolutional networks will be investigated for future studies.

1.2 Thesis Structure and Contribution

In this section, we briefly describes the content of each chapter along with the contributions.

- **Chapter 2** introduces the preliminary knowledge related to this thesis. We begin with the concepts of three broad categories of machine learning, *e.g.*, supervised learning,

unsupervised learning, and semi-supervised learning. We then discuss the major nonlinear algorithm in machine learning, *i.e.*, neural networks, which make up the backbone of deep learning algorithms. Next, a key component known as activation function that introduces nonlinearity to neural networks is described. After that, several types of deep learning neural networks are introduced, *e.g.*, autoencoders, convolutional neural networks, and invertible neural networks. At last, we introduce structural health monitoring and discuss the structural models studied in this thesis in detail.

- **Chapter 3:** We propose a novel parallel autoencoder framework that can be applied to structural damage identification, where the input data could be in multiple magnitude scales or have multiple physical meanings. This framework is composed of two components. The first component aims to achieve parallel dimensionality reduction and feature extraction for each subset of the input's elements. The learned features from each subset are concatenated into a new input vector that forms a lower dimensional representation of the original input. The second component learns the relationship between the learned representation and the output. Furthermore, we introduce sparsity constraint in the first component for performance improvement. Two experiments are conducted and our results show the significant advantages of the proposed parallel architecture based model in comparison with the existing state-of-the-art approaches.

- **Chapter 4:** We propose a novel structural health monitoring framework based on deep residual networks to perform effective and efficient feature extraction damage identification. This framework can be applied to perform end-to-end learning between the vibration characteristics such as modal information and the structural damage indices such as stiffness reductions. Also, it is feasible to build a very deep model with our framework and it can be applied to more complex civil structures. In the numerical studies, a seven-strorey frame is investigated and several experiments are conducted on various datasets to investigate the effects of noise in the measurement data and uncertainties in finite element modelling. In the experimental studies, a laboratory T-section reinforced concrete bridge model is used to further demonstrate the effectiveness of the proposed approach. It is observed that this framework is superior to the state-of-the-art models for this study.

- **Chapter 5:** We develop a densely connected convolutional network framework for structural damage identification, termed as SDI-DenseNet. Diverse levels of features are preserved and reused in training, which fits well for the study using acceleration responses. Time-domain acceleration responses are used in this study to identify the damages of structures. Fewer sensors are required to measure the acceleration responses used in this study, while a large number of sensors are needed to calculate the mode shapes used in Chapters 3 and 4. It is more practical to directly use the acceleration data. Extensive experiments including numerical studies and experimental studies are conducted to validate the improvements of the proposed SDI-DenseNet. The results reveal that, in terms of both effectiveness and accuracy, the proposed SDI-DenseNet outperforms existing state-of-the-art approaches.

- **Chapter 6:** We investigate the conditions under which the hidden layers of rectifier neural networks are invertible. An novel fixed-point algorithm is first proposed and the experimental results show that it is more widely applicable than the existing fixed-point algorithm presented in [12]. Next, a sufficient and necessary condition for a residual block to be invertible is presented and the constraint is further relaxed. A new invertible residual network is proposed which shows good discriminative performance on digit classification and structural damage identification. The potential benefits of using the proposed invertible residual network to improve the robustness and security of the deep learning-based SHM frameworks against adversarial attacks is discussed.

- **Chapter 7:** We review the main content and concludes the whole thesis. Some promising directions are also discussed for future study.

Chapter 2

Background

Structural health monitoring (SHM) is essential for maintaining the service life of civil structures. Over the past decades, considerable effort has been made toward developing vibration-based methods that utilize the vibration response of structures to assess their condition and discern structural damage. Meanwhile, with great success in various artificial intelligence (AI) research areas, machine learning (ML) and deep learning (DL) algorithms have also been extensively applied to vibration-based structural damage identification. In this chapter, we first present some background knowledge and related techniques on ML and DL. Afterward, a brief introduction to SHM and vibration-based damage identification is presented. The structural models we used in this thesis are then described in detail. Finally, we summarize the content of this chapter.

2.1 Machine Learning

Machine learning aims at designing efficient and robust algorithms that generate accurate predictions for unseen data. A common recipe [34] to build an ML algorithm is to combine several essential components, such as a dataset, an objective function, a model, and an optimization procedure. ML algorithms can be broadly categorized into several scenarios according to the types of available training data and how the training/test data are used during the learning process. Common machine learning methods such as supervised learning, unsupervised learning, and semi-supervised learning which are related to our research are briefly introduced in the following sections.

2.1.1 Supervised Learning

Supervised learning algorithms use a set of labeled examples as training data, and make predictions for the unseen(test) data. For example, a set of emails are labeled as "spam" or "non-spam". A supervised learning algorithm can extract useful features from the emails and classify them into two different categories. Sometimes it is not practical to obtain a large amount of labeled data for learning. Supervised learning is the most common scenario associated with classification, regression, and structured output problems.

2.1.2 Unsupervised Learning

Unsupervised learning algorithms use unlabeled training data and make predictions for the unseen data. As the training samples have no labels, the algorithms learn useful properties of the structure of the dataset instead of assigning labels to the input. It is difficult to quantify the evaluation performance when labeled examples are not available. Dimensionality reduction and clustering are examples of unsupervised learning [41].

2.1.3 Semi-supervised Learning

Semi-supervised learning algorithms use both labeled and unlabeled training data and make predictions for the unseen data. It is a special instance of weak supervision [14] which falls between unsupervised learning and supervised learning. Using semi-supervised learning is a common practice when the labeled samples are scarce or expensive while the unlabeled samples are easily accessible. Also, it helps people to understand how the learning behavior may change in the presence of both labeled and unlabeled data, and benefit from such a combination [112]. Thus, semi-supervised learning is of great interest in ML as it can use readily available unlabeled data to improve supervised learning tasks, including classification, regression, or ranking tasks.

2.2 Neural Networks

Artificial neural networks (ANNs), also known as neural networks, is a machine learning algorithm inspired by the biological human brain [93]. Deboeck *et al.* [22], also described ANNs as a collection of mathematical techniques that can be used for signal processing, forecasting, and clustering and termed it as nonlinear, multi-layered, and parallel regression

techniques. ANNs have emerged in the past decades and are widely used for engineering purposes, such as nonlinear pattern recognition and regression. Several pattern recognition problems in the structural health monitoring research domain have been solved using ANNs. It was observed by Yun *et al.*. [109] that the joint damages occurring at the beam-to-column connections of a steel frame structure can be predicted from modal data via the ANN approach. Based on back-propagation, a trained neural network in [77] performs well in estimating the location and severity of damages in a bridge structure. Also, a statistical approach is presented in [9] to consider the effect of uncertainties in developing an ANN model for structural damage detection.

A standard neural network is organized in layers made up of a number of connected nodes called neurons, with each of them followed by an activation function. The connections between the neurons are attached with some coefficients called weights. Each neuron receives a signal as input, processes it, and transmits an output through the connection. A typical architecture of an ANN consists of an input layer, one or two hidden layers, and an output layer. As defined in [34], the output of a neuron in each layer (input layer excluded) is computed by a two-step process: first the input of each neuron is computed by a weighted sum that is an affine mapping from $\mathbb{R}^d$ to $\mathbb{R}$, then the output of the neuron is computed by the activation function which is an element-wise mapping from $\mathbb{R}$ to $\mathbb{R}$. Input signals are propagated layer by layer and finally the output is compared with the targets to compute the errors by the objective function, *i.e.*, the cost function. Once a cost function is selected for a specific task, one can train the ANN with gradient descent back-propagation(BP) algorithms. However, as the number of hidden layers increased, the "gradient values" might be vanished during the process of back-propagation. Hence, it is difficult to optimize the weights in neural networks with a deep architecture. Before we introduce more advanced approaches, *e.g.*, a branch of ML called deep learning, a key component of ML is described below which is highly related to this thesis, *i.e.*, activation functions.

2.2.1 Activation Functions

We have introduced above the basic structure of a neural network, where the general form of a hidden layer is a linear transformation followed by an activation function $g(\cdot)$. Activation functions play a major role in the success of training neural networks because they can improve the performance of feature learning and representation. A neural network without

an activation function just acts as a linear regression model with limited capacities, which fails to model some complex types of data like images, videos, or other nonlinear/high-dimensional numerical data. Thus, to enable the network to represent complex (nonlinear) mappings from inputs to outputs, we need to apply nonlinear activation functions in the network. Another important feature of an activation function is the differentiability as it allows us to implement the back-propagation optimization techniques to reduce the loss/error when training to optimize the weights. In our study, several popular activation functions are utilized in different networks as described below.

sigmoid Function

The *sigmoid* function, sometimes also referred to as the logistic function [100], is one of the most widely used nonlinear activation function. Its mathematical form is denoted as:

$$g(x) = \frac{1}{1 + e^{-x}}. \tag{2.1}$$

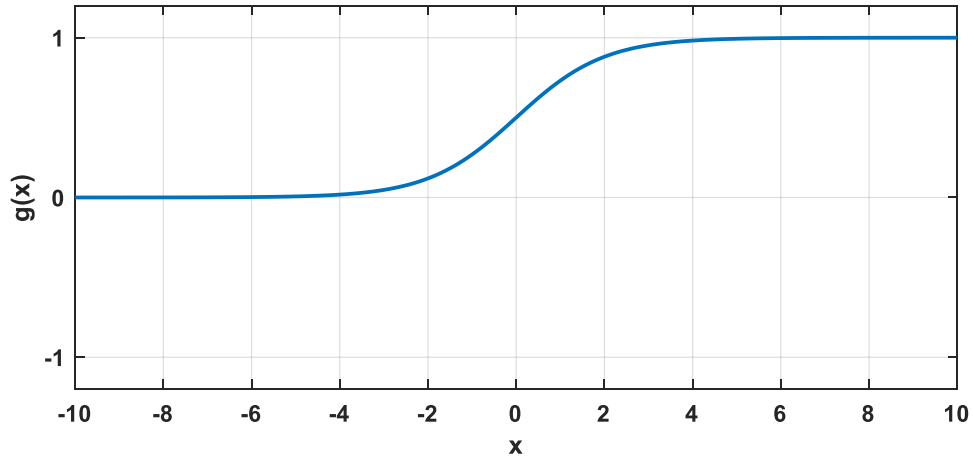

Figure 2-1: Logistic sigmoid function.

As seen in Figure 2-1, the characteristic S-shape gave the *sigmoid* function its name. It is continuously differentiable, making it possible to train a neural network by back-propagation algorithms.

The *sigmoid* function maps the real values in range $(-\infty, \infty)$ onto (0, 1). It saturates to 0 when its argument is very small whereas saturates to 1 when its argument is very large, which means the function becomes very flat and insensitive to small changes when

the input is very large or very small. Furthermore, the *sigmoid* function is non-centered which means the signs of all outputs are the same, either all positive or all negative. It has been reported that the saturated and non-centered properties may leads to the gradients vanished quickly during error back-propagation, especially when multiple layers are used in the neural network.

Hyperbolic Tangent Function

The hyperbolic tangent function, also known as *tanh* function, is similar to *sigmoid* function but it is zero-centered. This results in different signs of output values where its output range lies between -1 to 1, as shown in Figure 2-2. It can be described by the mathematical form:

$$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.2}$$

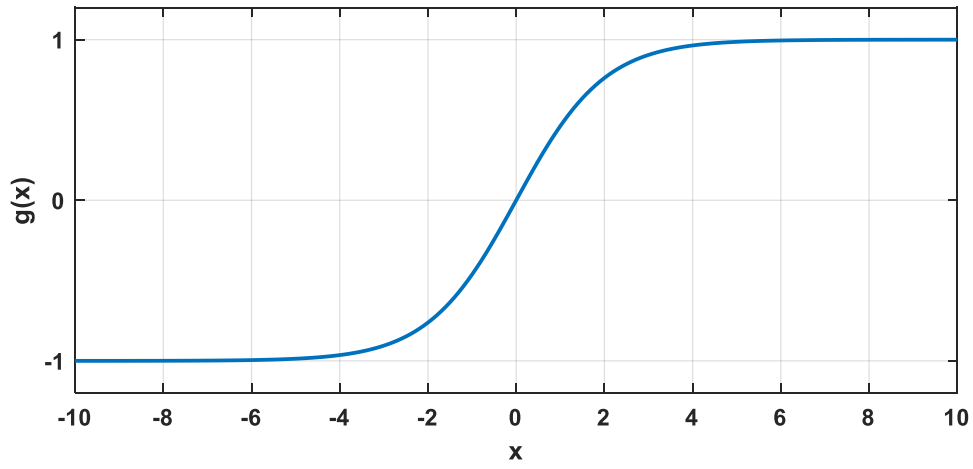

Figure 2-2: Hyperbolic tangent function.

As the output of this function is zero-centered thereby it helps the training with back-propagation. Thus *tanh* is preferred over *sigmoid* function, and it is observed that the *tanh* function performs better than *sigmoid* function when training a neural network with multiple layers [53,80]. However, the gradient vanishing problem could not be solved as well by the *tanh* function.

Rectified Linear Unit Function

The rectified linear unit ($ReLU$) function has achieved great success in deep learning (See Section 2.3) with state-of-the-art results since it was proposed by by [78].

Mathematically, it can be denoted as:

$$g(x) = max(0, x) = \begin{cases} x & if \quad x \geq 0 \\ 0 & if \quad x < 0 \end{cases}. \tag{2.3}$$

$ReLU$ is shown to be faster [61] than the conventional nonlinear activation functions, *e.g.*, *sigmoid* and *tanh* functions, as it is cheap to compute without complicated math function such as exponential. Also, it is the most widely used activation function in recent years [88]. If the input value x is less than zero, the output will be set to zero. When x is great than or equal to zero, the output will be identical to the input. As seen in Figure 2-3, although the $ReLU$ function is nonlinear, it still preserves the linearity when the input is positive. Its derivative will always be 1 even when x is extremely large, thereby alleviating the vanishing gradient problem suffered by *sigmoid* or *tanh*.

Another advantage of $ReLU$ is that it introduces sparsity in the network as it outputs zeros for all negative inputs. However, $ReLU$ function has some issues as well, such as the "dying" neurons problem, which occurs when a large negative bias is learned causing the activation of the neuron to be always zero. As a result, those dead neurons could learn nothing regardless of the inputs.

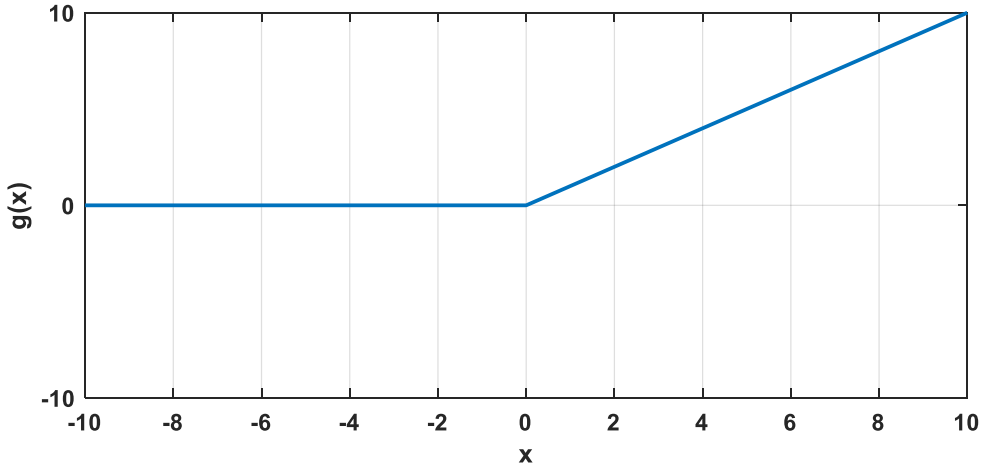

Figure 2-3: Rectified linear unit function.

Identity Function

An identity function is a linear function that outputs signal identical to the input as shown in Figure 2-4. It can be mathematically expressed as:

$$g(x) = x. \tag{2.4}$$

A neural network with only identity or linear activation functions is equivalent to a linear regression model. Thus, it could be used as the activation function of the output layer for a regression task.

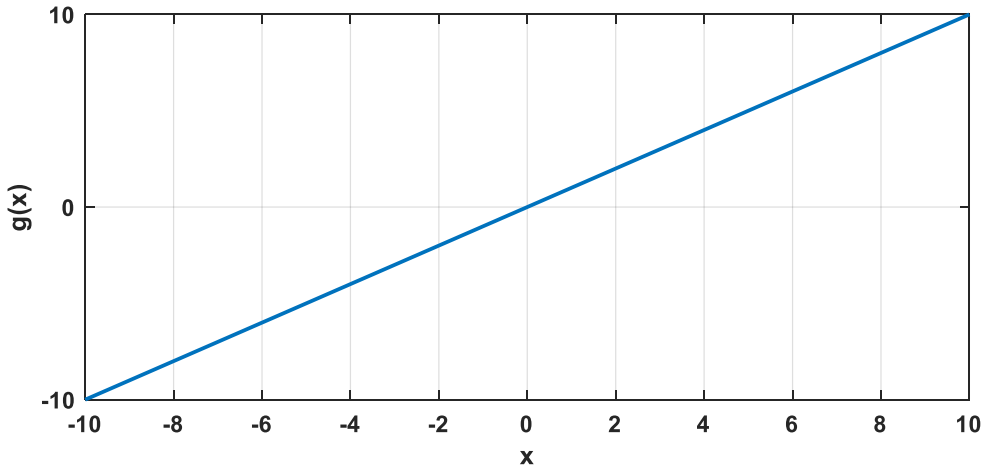

Figure 2-4: Hyperbolic tangent function.

2.3 Deep Learning

Deep learning (DL) methods are mutil-level representation learning methods that allows simple nonlinear processing modules to transform the representations from the raw input into higher levels, with each level of the representation becoming slightly more abstract [61]. With enough amount of such transformations, one can learn complex functions. The DL research was motivated to overcome the limitations of the conventional machine learning (ML) algorithms and improve generalization performances on some complicated artificial intelligence (AI) tasks, such as object or speech recognition [34]. In particular, conventional ML algorithms are limited to processing data in their raw form [61], and they are insufficient to learn complex functions to model the high-dimensional data [34]. DL was designed

to overcome such obstacles and has achieved many significant breakthroughs in the AI community in recent years. Most modern deep learning models are developed based on artificial neural networks with a deep architecture, thus are also known as deep neural networks. Depending on the learning tasks, deep neural networks can be applied to both supervised learning and unsupervised learning.

2.3.1 Autoencoders

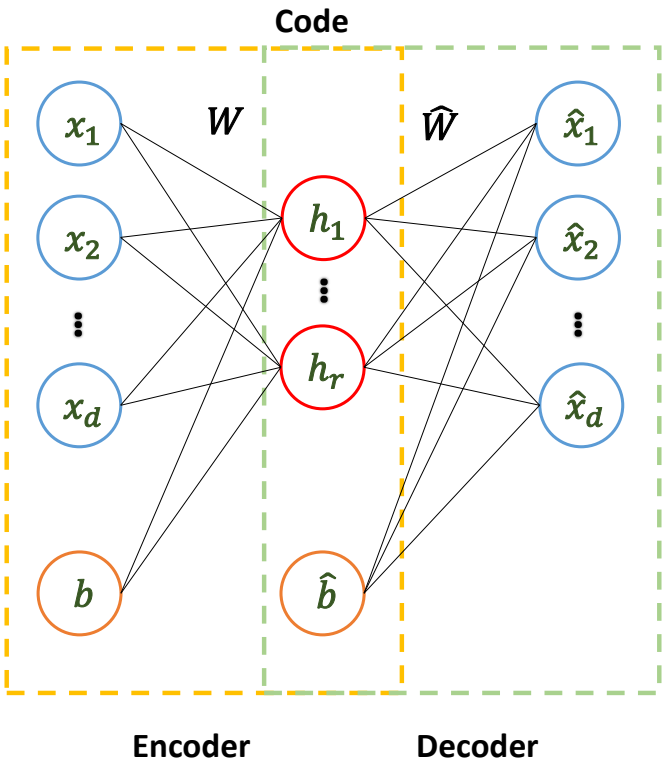

Figure 2-5: The structure of an autoencoder neural network.

An autoencoder [34] is a neural network that is trained to reconstruct its input to its output in an unsupervised manner. As shown in Figure 2-5, it includes two core components: encoder and decoder with a hidden layer **h** that describes a code used to compress the representation of the input.

Encoder: The deterministic function $f(x)$ that maps a input vector **x** of dimension d to a hidden representation **h** of dimension r, is referred as to an encoder. Its typical form is an affine transformation followed by an nonlinear transformation, which can be defined as follows:

$$\mathbf{h} = f(\mathbf{x}) = \phi(W\mathbf{x} + \mathbf{b}) \tag{2.5}$$

where $W \in \mathbb{R}^{r \times d}$ is the weight matrix of the affine transformation and $\mathbf{b} \in \mathbb{R}^d$ is the bias vector. $\phi(\cdot)$ denotes an element-wise nonlinear activation function, which could be a *sigmoid* function $\phi(\mathbf{x}) = \frac{1}{1+e^{-x}}$ function or a hyperbolic tangent function $\phi(\mathbf{x}) = \frac{e^{2x}-1}{e^{2x}+1}$.

Decoder: The function $g(\mathbf{h})$ that transforms the hidden representation $\mathbf{h}$ back into a reconstructed vector $\widehat{\mathbf{x}}$ in the input space, is called a decoder. Similarly, the decoder can be represented in the same fashion, an affine mapping followed by a nonlinearity:

$$\widehat{\mathbf{x}} = g(\mathbf{h}) = \psi(\widehat{W}\mathbf{h} + \widehat{\mathbf{b}}) \tag{2.6}$$

where $\widehat{W} \in \mathbb{R}^{d \times r}$ and $\widehat{\mathbf{b}} \in \mathbb{R}^r$ are the weight matrix and the bias vector of the decoder, respectively. $\psi(\cdot)$ denotes an element-wise nonlinear activation function, which is similar to $\phi(\cdot)$ described above.

Autoencoders can be treated as a special case of feedforward neural networks that often trained with gradient descent optimization methods with the gradients computed by back-propagation. The training of autoencoders aims to minimize the reconstruction errors by ensuring the hidden neurons capture the most appropriate features of the data.

2.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) is a specialized kind of neural network designed to process data that has a known grid-like topology. For instance, the time-series data can be considered as a 1-D grid sampling at regular time intervals, while the image data can be considered as 2-D grid of pixels. Applying CNNs to those gird-like data, one can learn the robust features from the input signal which involves the time or space correlations. A typical building block of a convolutional neural network is composed of three layers, which are the convolution layer, nonlinear activation layer, and pooling layer. Recently, batch normalization (BN) proposed in [49] is widely used for faster and more stable training of deep neural networks. In practice, the BN layer is typically inserted after the convolutional layers. These layers process the input in four successive stages: 1) Convolution; 2) Batch normalization; 3) Nonlinearity; and 4) Downsampling, to perform feature extraction. Details of these stages are presented in the following sections.

Convolution

In the first stage, the convolutional layer produces a set of linear operations by performing several convolutions in parallel in place of general matrix multiplication in traditional fully-connected layers. The convolution operation is known as a feature detector, which is often interpreted as a kernel that filters the input data for certain kinds of features. For example, an edge kernel only filters the edge information of an input image. Figure 2-6 shows the convolution operation and demonstrate how the kernel slides over the input to produce the convolved features output, also called a feature map. It can be formulated as *Output=Kernel* $*$ *Input*, where '$*$' denotes the convolution operator. At each location, the kernel operates a dot product between the input values within its local region and sums the results. This sum goes to a single entry in the output feature map. Finally, the feature map will be fed into the next layer as the input. In typical CNNs, a number of kernels will be learned in each convolutional layer, and the parameters in the kernels are known as weights which are optimized by gradient descents with back-propagation.

Batch normalization

In the second stage, every mini-batch of input is normalized by firstly subtracting the batch mean and then divided by the batch standard deviation. Supposing that we have a mini-batch of inputs: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$, the variance σ^2 and mean μ of this mini-batch are computed to perform normalization as below:

$$\mathbf{y}_i = \gamma \frac{\mathbf{x}_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{2.7}$$

where γ and β perform scaling and shifting on the standardized $\mathbf{x}_i$, respectively. ϵ is added to the mini-batch variance to avoid dividing by zero, which is typically a small constant. BN stabilizes and accelerates the training of neural networks, especially for the deep models.

Nonlinearity

In the third stage, the input is run through a nonlinear activation function, such as *sigmoid*, *tanh*, or *ReLU*. *ReLU* has become the standard activation function when developing the CNNs after it is shown to converge faster with superior performance in AlexNet. Unlike *sigmoid* and *tanh* activation function, it is linear when the input values are positive, and

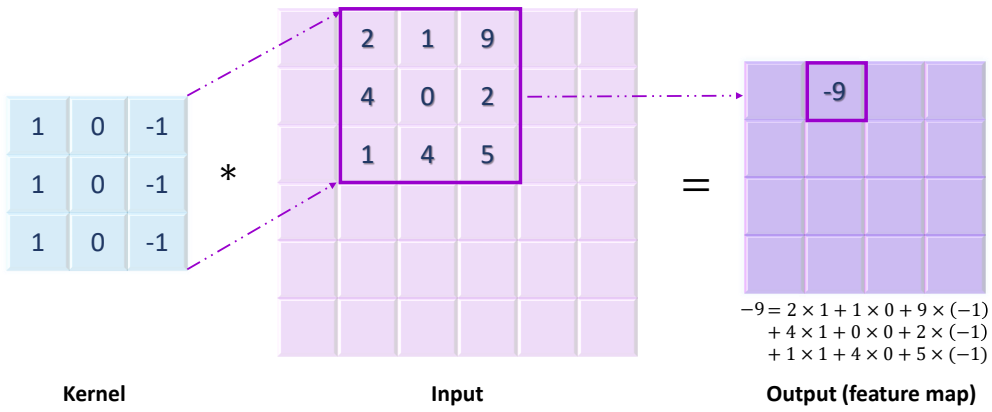

Figure 2-6: Convolution operation.

gradients will always be 1. Thus it helps to reduce the gradient vanishing effect like *sigmoid* or *tanh*, making the error back-propagation easier in the network. Besides, when the input is negative, $ReLU$ will convert it to zero, and the neuron will be turned off during training. It enforces a nice property in feature learning called sparsity, which can accelerate learning and improve the network generalization performance.

Subsampling versus Strided Convolution

In the last stage, it is common to insert a pooling layer in between the successive convolutional layers. Pooling layers are also called subsampling layers as they perform downsampling on the input feature maps to reduce their dimensionality in both width and height. As a consequence, the number of parameters is reduced, which can avoid overfitting and boost the training speed. The most commonly used pooling layer is max-pooling and average pooling, where the former one uses max operation, while the latter applies an average operation on the input.

It should be noted that there is no learning done in the pooling layer as it just downsampling operation. Recently, many researchers replace pooling layers with strided convolution layers. The stride of the convolutional layer is set to be greater than 1 (*e.g.*, stride = 2), which can be considered as learning the pooling operation. It is optional to use pooling layer or strided convolution in the convolutional neural networks.

2.3.3 Invertible Neural Networks

A deep neural network with invertible hidden layers has a nice property of preserving all the information in the feature learning stage. Invertible neural networks (INNs) which refers to a neural network with such invertible property, have received considerable attention recently in both supervised learning and unsupervised learning, and many invertible neural networks [6, 8, 19, 24–26, 60, 71, 79, 96, 99, 103, 113] have been proposed with wide applications such as adversarial attack, image generation, image classification, speech recognition and compressed sensing. Mathematically, INNs can be seen as bijective function approximators [12], which have a forward mapping and an inverse mapping. Consider an invertible neural network, the forward mapping is denoted as $F_\theta : x \in \mathbb{R}^d \mapsto z \in \mathbb{R}^d$, while the inverse mapping is denoted as $F_\theta^{-1} : z \in \mathbb{R}^d \mapsto x \in \mathbb{R}^d$. Through the inverse mapping, the input x can be recovered via output z without any information loss. With this property, it can implicitly define a normalized density model which can be directly trained by maximum likelihood [96]. This is particularly useful for generative modelling in unsupervised learning. Moreover, an invertible neural network for feature learning can help analyze the learnt latent features and improve the interpretability and explainability of discriminative models for classification or regression.

The existing invertible neural networks can be separated into two categories: one uses non-standard network architectures with specially designed structures while the other uses standard neural networks such as ResNet [39] but with imposed restrictive constraints. Typical invertible neural networks in the first category includes NICE [24] and Real-NVP [25] which uses dimension partitioning and coupling layers. [12] and [96] are typical invertible neural networks in the second category. Both of them use ResNet, the state-of-the-art neural network for image classification, as the basic neural network architecture. [12] imposes a contractive condition on the Lipschitz-constant of the convolution path so that the residual block is invertible, while [96] uses masked convolutions and a set of composition rules.

2.4 Structural Health Monitoring

Structural damage identification has gained increasing attention from both society and community due to the unexpected structural failure may lead to catastrophic, economic, and human life loss. The process of implementing a structural damage identification strategy to

monitor the structural performance under operational conditions is referred as Structural Health Monitoring (SHM) [30]. The main objective of SHM is to identify the presence, location, and severity of damages in civil infrastructures at an early stage, and then to predict the remaining life and make the future maintenance decision of the monitored structures.

In the past decades, extensive research has been carried out in vibration-based structural damage identification, and significant progress has been achieved in this area [64]. The fundamental idea behind the vibration-based structural damage identification is that the damage-induced changes in the physical properties (stiffness, mass, damping ratio) will cause detectable changes in vibration characteristics (natural frequencies, mode shapes, and acceleration responses) of the structure [28]. Therefore, by extracting and analyzing damage-sensitive features from the measured vibration characteristics, one can determine the current state of structural health.

2.5 Numerical and experimental models

Different structures have been investigated in SHM research domain, from simple structural components (*e.g.*, beams [55]) to complex structural systems (*e.g.*, buildings [69] and bridges [47]). The following sections describe several structural components/systems that are used in this thesis, including both numerical and experimental models. Besides, a summary of the datasets simulated/collected from these structural models is presented.

2.5.1 Numerical Models

In this thesis, we proposed several novel deep learning-based frameworks for structural damage identification, which are first evaluated by the simulation data generated from the numerical finite element model (FEM). Two numerical models, a simply supported beam and a seven-storey steel frame are presented in the following sections. While the details of the experiments including data generation, and data pre-processing will be described separately in the experiment section of each chapter.

Numerical Structure and Finite Element Model - A Beam Structure

A simply supported beam with a width of 0.6 m, a height of 1 m, and a length of 20 m is used for the numerical study. The mass, Young's modulus, and cross-sectional moment

of inertia of the beam are 2500 kg/m^3, 3.3×104 MPa, and 0.05 m^4, respectively. The first five natural frequencies of the undamaged beam model are respectively 4.12 Hz, 16.48 Hz, 37.10 Hz, 66.04 Hz, and 103.425 Hz. Figure 2-7 shows the finite element model of this beam structure. There are ten elements and eleven nodes. Each node has two degrees of freedom, namely, a vertical transitional displacement and one rotational displacement. Acceleration responses in the vertical direction measured from four nodes at No. 2, 5, 8, and 9 are used for training and testing of the proposed network. The acceleration responses are generated using a random impact force applied at node 6. It should be noted that random impact force is generated with a Gaussian distribution with a mean value of 8000 N and a standard deviation of 50.

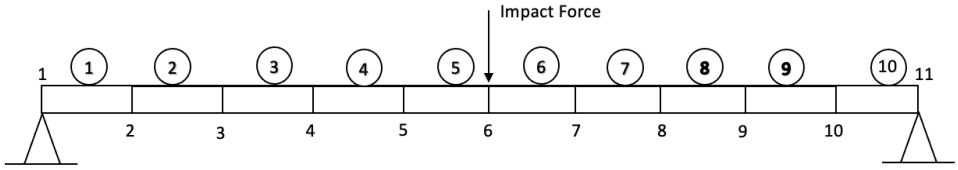

Figure 2-7: A simply supported beam model.

Numerical Structure and Finite Element Model - A Frame Structure

A seven-storey steel frame structure has been built in the laboratory and its dimensions are illustrated in Figure 2-8. Each storey is 0.3m in height, composing up to 2.1m for the total column height of the steel frame whereas the beam length of the steel frame is 0.5m. The cross-sections of the column and beam elements are shown with dimensions of 49.98 mm×4.85 mm and 49.89 mm×8.92 respectively, while the corresponding measured mass densities of the column and beam elements are 7850 kg/m^3 and 7734.2 kg/m^3. Initial Young's modulus of 210 GPa is applied on all members. The column and beam elements are connected continuously by welding at the top and bottom of the beam sections. The two columns at the bottom of the steel frame are welded onto a thick and solid steel plate which is fixed to the ground. In order to simulate the mass from the floor of a building structure, two pairs of mass blocks with each around 4 kg in weight are fixed at the quarter and three-quarter lengths of the beam in each storey.

The finite element model of the whole frame structure is shown in Figure 2-9 which includes 65 nodes and 70 planar frame elements. The weights of steel blocks are loaded

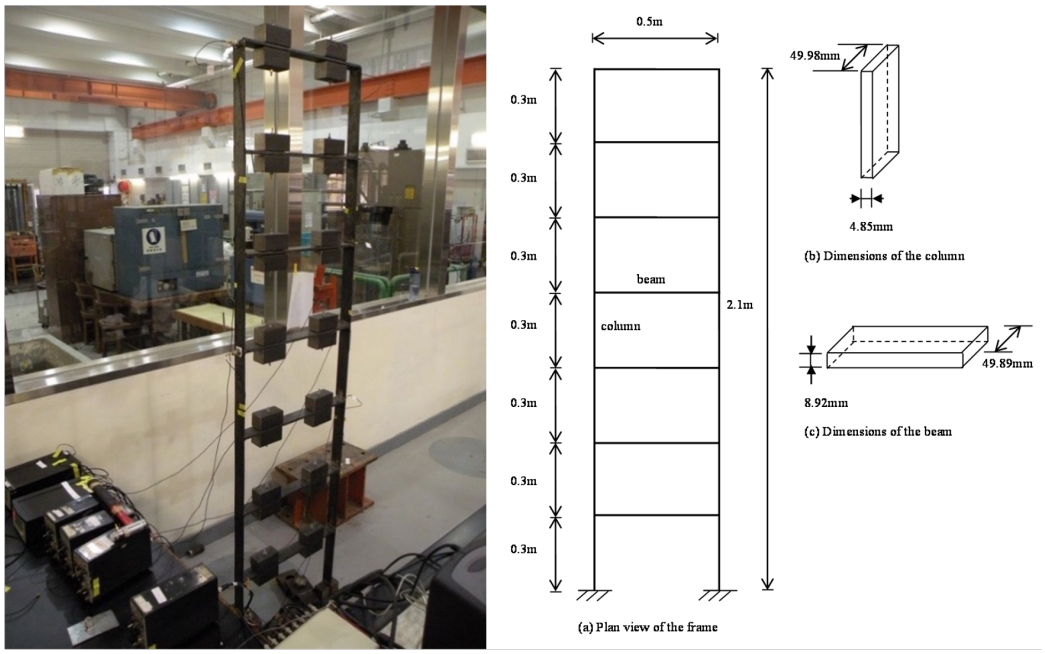

Figure 2-8: Laboratory steel frame model and the dimensions of the steel frame structure.

as concentrated masses at the corresponding nodes of the finite element model. Each node has three Degrees-of-Freedom (DOFs), including two translational displacements and a rotational displacement. The structure has 195 DOFs in total. The translational and rotational restraints at the supports (Nodes 1 and 65) are expressed initially by the large stiffness of 3×10^9N/m and 3×10^9N.m/rad, respectively. The initial finite element model updating has been executed to minimize the discrepancies between the analytical finite element model and the experimental model in the laboratory. This updated finite element model is adopted as the baseline model for generating the training, validation and testing data. The detailed model updating process and results can be referred to [77].

2.5.2 Experimental Models

Experimental studies are conducted to further validate the effectiveness of the proposed deep learning-based frameworks for structural damage identification. The following laboratory models are utilized in this thesis and the data is collected from the installed sensors. Details of the data generation and experimental settings will be described in the experiment section of each chapter.

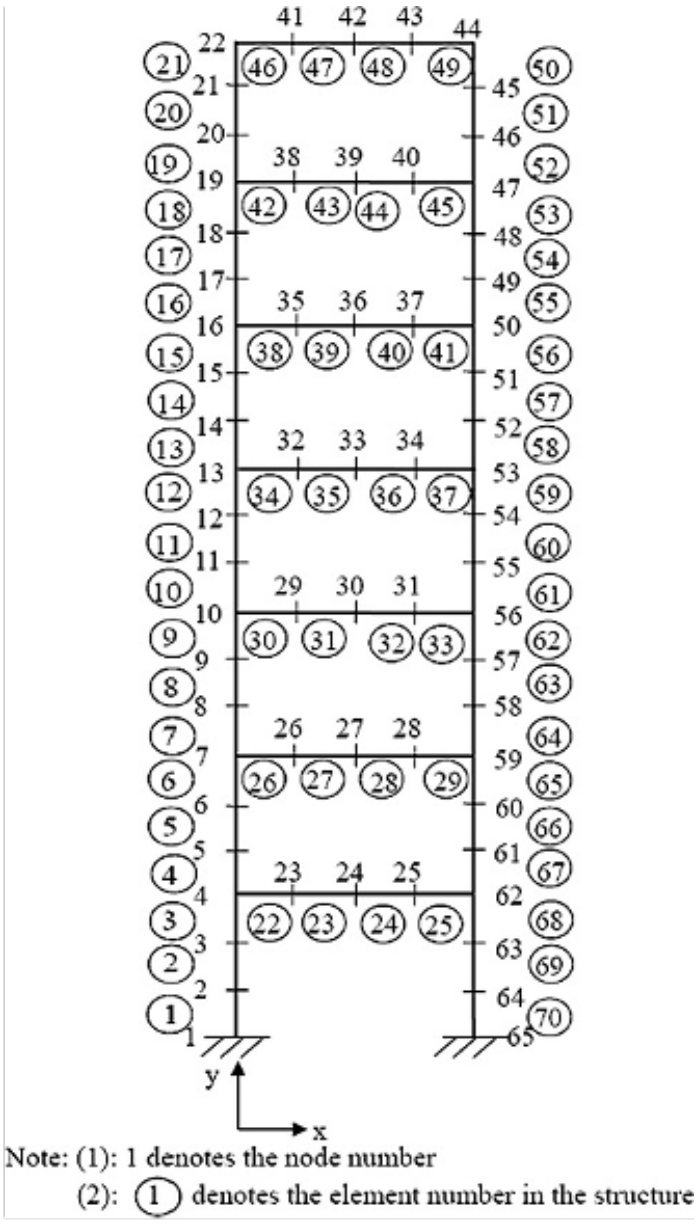

Figure 2-9: Finite element model of the steel frame structure.

A T-section Pre-stressed Concrete Bridge Model

A T-section pre-stressed concrete bridge model, as shown in Figure 2-10, is fabricated in the laboratory and tested to validate the effectiveness of the proposed approach for the identification of structural damage. Figure 2-11 shows the dimensions of the bridge model and the location of sensors (accelerometers) on the structure. The length, slab, web widths and the height of the bridge are 5m, 0.65m, 0.15m and 0.415m, respectively. Initial Young's modulus of 2.6×104 MPa is applied while the density is 2707.7 kg/m^3. Three pre-stressing tendons are included in the bridge with a combined pre-stressing strength of 140kN. Each tendon has an area of 99.8 mm^2 and it has the tensile strength of 1949N/mm^2. More information about the bridge can be found in [66]. To measure dynamic vibration responses in the vertical direction, seven sensors (accelerometers) are placed on the top of the bridge structure.

Figure 2-10: The experimental testing model.

The model updating is conducted to prepare an accurate baseline model for generating the training data. Using a modal hammer to excite the model, dynamic tests are performed to identify the model's vibration properties, *i.e.*natural frequencies and mode shapes. The sampling rate is set to 2000 Hz to accommodate the bridge model's frequency range of

excited modes. As shown in Figure 2-12, an initial finite element model of the bridge is built with flat shell elements. The finite element model consists of 90 elements and 114 nodes. Each node has six DOFs. The initial updating of the model is carried out to modify the finite element model to serve as the baseline model. The updating process is carried out by minimising the difference between the first three natural frequencies and calculated mode shapes from the finite element model and measurements from the dynamic tests. Young's modulus of slab and beam web and the support stiffness are selected as parameters to be updated in the initial model update. The comprehensive updating method and results can be found in a previous study [65].

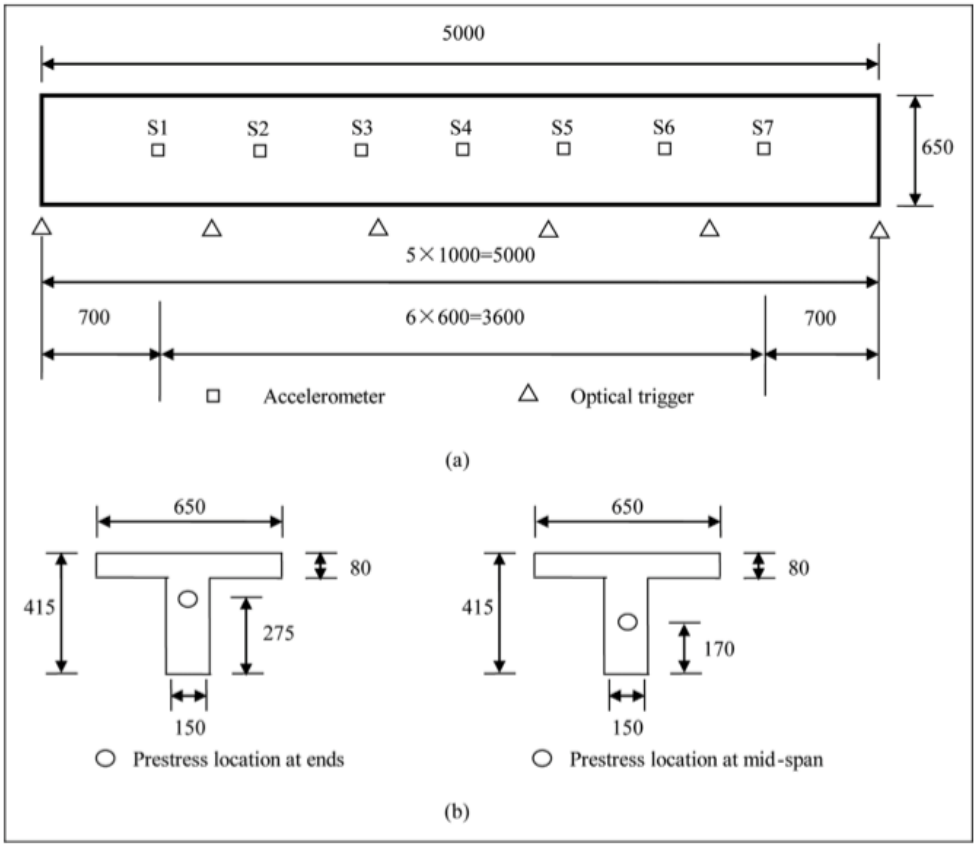

Figure 2-11: Sensor placement and Dimensions of the testing model.

A Seven-storey Steel Frame Structure

A seven-storey steel frame structure has been fabricated in the laboratory used for validating substructure damage identification approaches [65]. This structure is introduced in Section

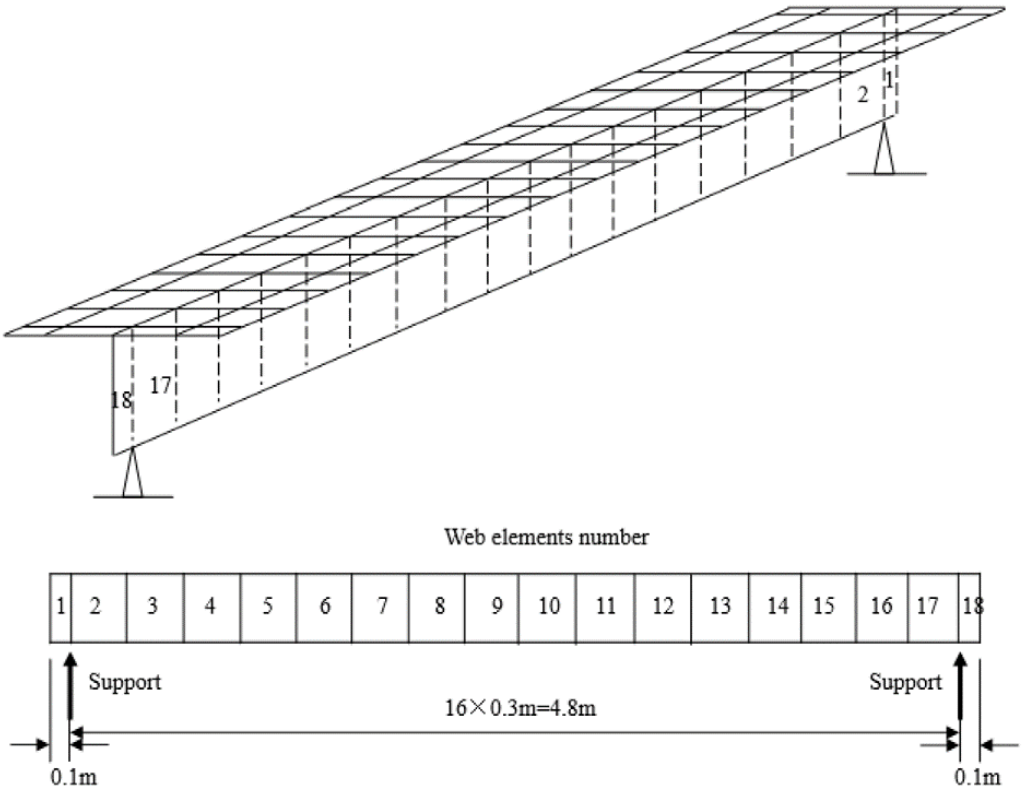

Figure 2-12: Finite element model of the testing bridge.

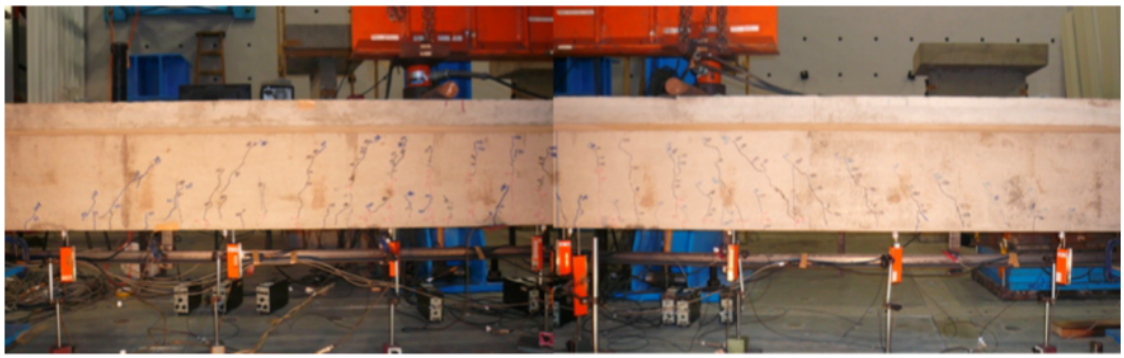

Figure 2-13: Introduced cracks in the structure during testing [66].

2.5.1 with a finite element model that is employed in the study in Chapters 3 and 4. It should be noted that although the same frame structure is also used for the research in Chapter 5, the data type and experimental settings are different. Thus in this section we briefly introduce this structure for experimental study with a different experimental setup.

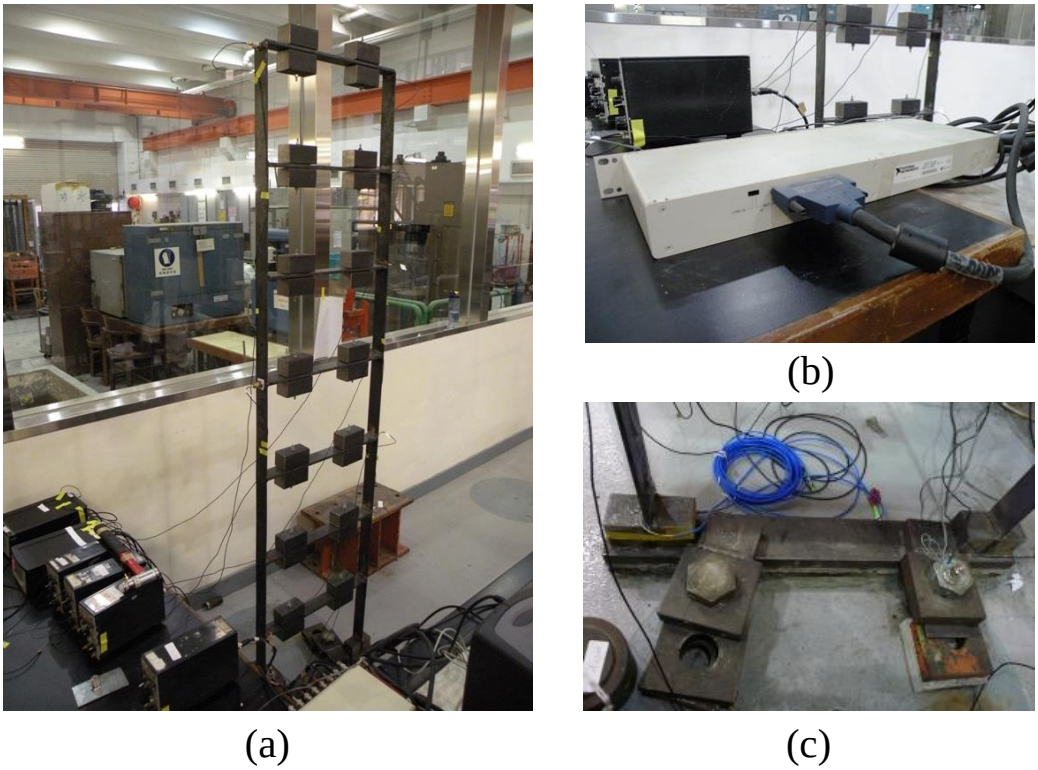

Figure 2-14: Experimental testing model: (a) a steel frame structure; (b) data acquisition system; (c) fixed bottom support.

Figure 2-14 shows the steel frame model and its experimental setup for vibration tests. An updated finite element model is obtained to minimize the difference between the built finite element model and the experimental model. There are 70 elements defined in the finite element model of the structure. Both numerical data and experimental data of this structure are generated for validating the efficacy and accuracy of the proposed deep learning approach in Chapter 5. Numerical data generated from the updated finite element model are used for training, validation, and testing. The experimental data collected from laboratory testing of damaged structures are only used for testing. Two damage cases are introduced in the frame structure in laboratory testings. The first one is a single-element damage case with a 12.5% stiffness reduction at the 6^{th} element, and the other one is a two-element damage case with a 12.5% stiffness reduction in both the 6^{th} and 12^{th} elements. Figure 2-15 shows

the introduced damage cases with a single damage and two damages.

(a) (b)

Figure 2-15: Damage cases in the frame structure: (a) Single damage; (b) Two damages.

2.5.3 Datasets

As discussed above, both numerical data simulated via FEMs and experimental data collected from installed sensors are investigated. As a quick overview of these datasets, a summary is shown in Table 2.1. Besides, several typical data samples, *i.e.*, acceleration responses of a beam structure, are illustrated in Figures 2-16 and 2-17 to provide an intuitive understanding of the nature of the data.

Table 2.1 outlines four main groups of data that are investigated in this thesis. Various types of noise effects, *e.g.*, measurement noises and modelling errors, are considered in the data generation/collection process. Details of the data generation process are discussed in

Table 2.1: Summary of the SHM datasets used in this thesis.

Structure	Data type	Model type	Format
A seven-story steel frame	Frequencies & Mode shapes	Numerical	1D
A simply supported beam	Acceleration responses	Numerical	2D
A T-section pre-stressed concrete bridge	Acceleration responses	Experimental	2D
A seven-story steel frame	Acceleration responses	Experimental	2D

the following chapters.

Figure 2-16 demonstrates a set of typical acceleration responses at four different sensor locations of a simply supported beam structure in its undamaged state. On the other hand, Figure 2-17 shows the acceleration responses measured at the same location in the undamaged and damaged states, respectively. The sampling rate is 1000 Hz and the measurement time is 0.2 seconds. In this thesis, the responses collected from multiple sensors are arranged into 2D arrays (measured responses $\times$ number of sensors) and presented as the input to the proposed deep learning models.

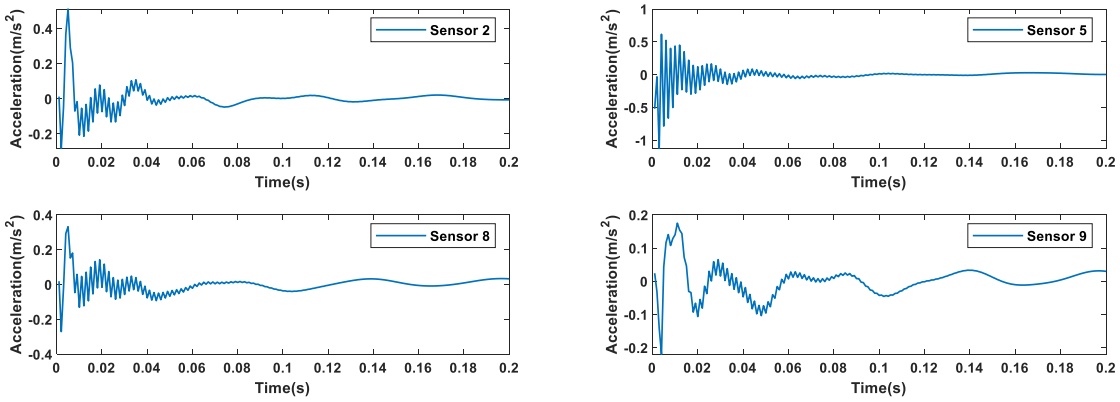

Figure 2-16: Acceleration responses at four sensors.

2.6 Summary

This chapter provides some background knowledge related to the topic of this thesis. Firstly, three main types of learning problems in machine learning: supervised, unsupervised, and semi-supervised learning are explained. Next, neural networks are described in detail followed by a key component, *i.e.*, activation functions, of the design of neural networks. Next, deep learning and several types of deep neural networks, *e.g.*, autoencoders and convolutional

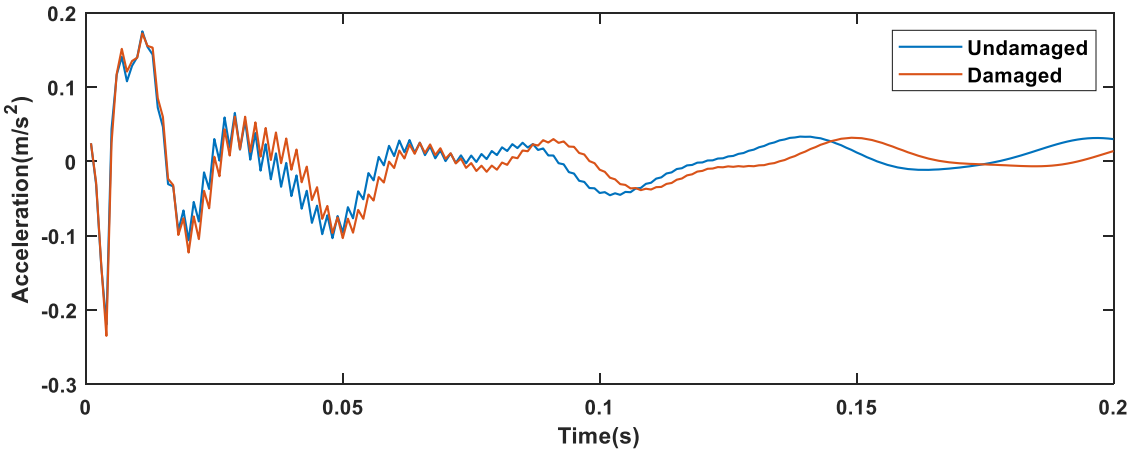

Figure 2-17: Acceleration responses of the simply supported beam structure in undamaged state versus damaged state.

neural networks, are introduced. In particular, a brief literature review of a novel class of networks called invertible network networks is presented. After that, a brief introduction to structural health monitoring and structural damage identification related to this thesis is provided. Finally, the numerical and experimental models, as well as their associated datasets, are presented.

The following chapters will describe several advanced deep learning based frameworks for structural damage identification and novel inverse algorithms for constructing invertible neural networks which can potentially be applied to structural health monitoring applications.

Chapter 3

Autoencoder Based Framework for Structural Health Monitoring

3.1 Introduction

Machine learning algorithms have been applied to solve a wide variety of important problems, including classification, regression, clustering, and dimensionality reduction for decades. Deep learning has recently been shown to be more efficient and effective for these tasks. Autoencoder is an unsupervised learning method that has usually been utilized to reduce the dimensionality of high-dimensional data while preserving useful information. The layer-wise pre-training strategy can also be employed in autoencoder based models. By stacking multiple hidden layers of autoencoders, a Deep Autoencoder (DAE) model can be established as a good classifier/regressor for pattern recognition in SHM. Recent works in [85, 86] show a much more promising future for deep autoencoder models to be applied in SHM in comparison with traditional ANNs. An autoencoder based deep learning framework is proposed in [85], namely AutoNet, which can achieve good performances for structural damage detection/identification via two stages: effective dimensionality reduction and accurate relationship learning. A sparse autoencoder based framework proposed in [86], namely SAF, further improves the performance via enforcing sparsity penalties in autoencoders. Both AutoNet and SAF are basically processing the entire input features together which include

[0]This chapter is reprinted, with permission, from [Wang, R., Li, L., & Li, J. A novel parallel auto-encoder framework for multi-scale data in civil structural health monitoring. *Algorithms*, 11(8), p.112. © 2018 MDPI. DOI: 10.3390/a11080112].

different types of features such as frequencies and mode shapes. This will increase the difficulty in training a robust model due to several issues: 1) As one mode shape is associated with one frequency and different mode shapes are unrelated to each other, it is better to deal with each of them specifically than put them together in dimension reduction; 2) The data for frequencies and mode shapes are in different magnitude scales, it is improper to put them together in the normalization process.

In this chapter, we introduce a novel parallel autoencoder framework (Para-AF) for SHM that achieves dimensionality reduction and feature extraction for the frequency data and mode shape data separately. The learnt features from different categories are fused for learning the relationship to the output, *e.g.*, stiffness of a seven-storey steel frame structure. In the proposed framework, sparsity constraint is also incorporated to enhance feature learning. The organization of this chapter is as follows: Section 3.2 describes the proposed parallel autoencoder framework (Para-AF); Section 3.3 evaluates the performances of the proposed framework with numerical studies; and Section 3.4 summarizes the work of this chapter.

3.2 The proposed parallel autoencoder framework.

In the steel frame structure shown in Section 2.5.1, the input data consists of seven frequencies and seven groups of mode shape parameters. As one mode shape is associated with one frequency and different mode shapes are unrelated to each other, it is more reasonable to perform dimension reduction on each group specifically. Also, frequencies and mode shapes are in different magnitude scales, for example, 7.6 and 0.9. It is improper to normalize them together in the pre-processing step as in [85] and [86]. Besides, vibration-based methods are generally vulnerable to a variety of noise effects in the damage identification process, such as the measurement noises in vibration data or uncertainties in the system. In consequence, the effects of some unnecessary information (*e.g.*, measurement noises, uncertainties, and redundant data) may not be well reduced at the dimensionality reduction stage in the high-dimensional features which are formed by all frequencies and mode shape parameters. Furthermore, introducing sparsity constraint in the training of an autoencoder model can achieve better de-noising performance.

Therefore, the proposed framework has been designed carefully as a parallel architecture

based on the sparse autoencoders, considering the difficulties mentioned above. As shown in Figure 3-1, the proposed framework includes the following two major components:

1. Dimensionality reduction component for
    - Scale-invariant, correlated, and noise-robust feature extraction from several groups of inputs, *e.g.*, frequencies and mode shapes, individually.
2. Relationship learning component for
    - Learning the relationship between the extracted features and the output, *i.e.*, elementary stiffness parameters.

Next, the generic building block of the proposed framework, namely sparse autoencoder, will be described in Section 3.2.1, and the two major components, namely parallel sparse dimension reduction and relationship learning, will be discussed in Section 3.2.2 and Section 3.2.3 respectively.

3.2.1 Sparse Autoencoders

An autoencoder is trained to reconstruct the input from its output as described in Section 2.3.1. However, if an autoencoder succeeds in simply copying its input to its output, it may not extract any useful features. A sparse autoencoder is an extension of the autoencoder whose training criterion involves a sparsity penalty term on the hidden neurons inspired by sparse coding [63, 84], along with the reconstruction error. By regularizing the autoencoder to be sparse, it must respond to essential statistical features of the training dataset, rather than purely acting as an identity function [34]. Hence, training a sparse autoencoder to perform reconstruction task can be used to learn useful features.

The detailed formulation of the sparsity penalty term [81] is explained as follows. Let $a_j(x^i)$ denotes the activation of hidden neuron j when the network is given the input x^i. Then, the mean activation of the hidden neuron j across the whole training dataset is

$$\widehat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} a_j(x^i) \tag{3.1}$$

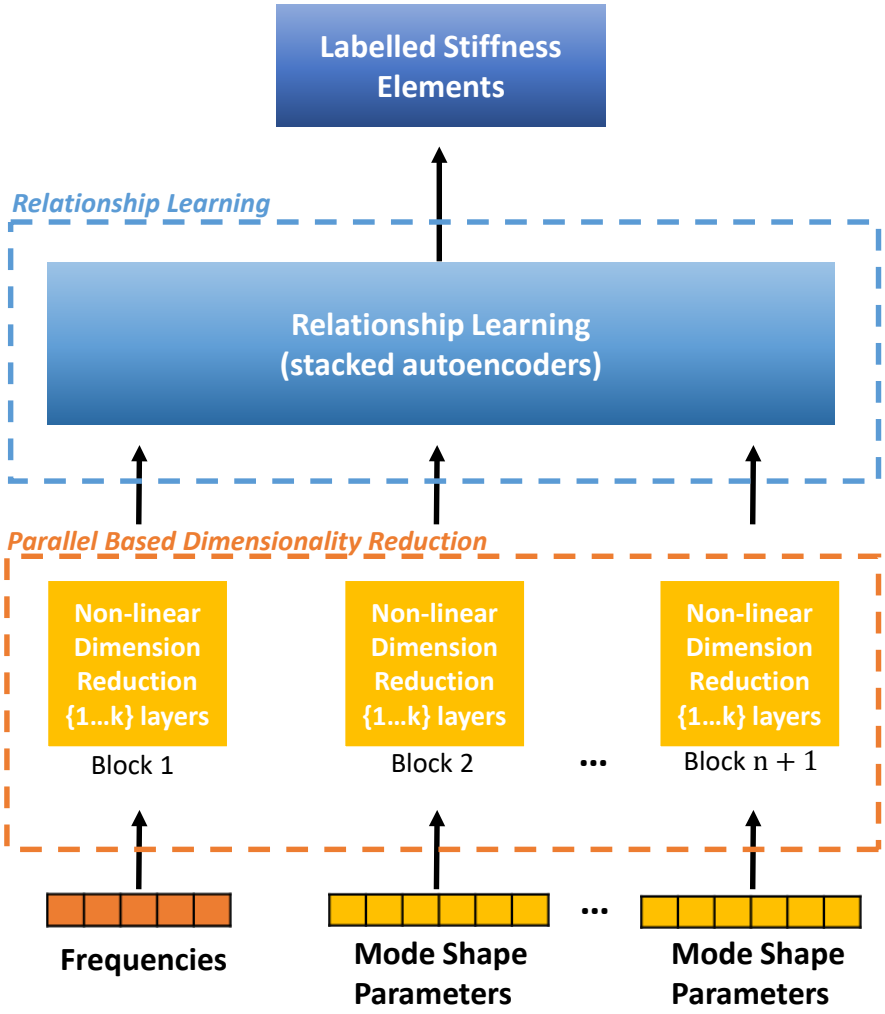

Figure 3-1: The proposed parallel sparse autoencoder framework.

where m is the number of the training samples. The sparse constraint is enforced with

$$\widehat{\rho}_j \approx \rho \tag{3.2}$$

where ρ is a sparsity parameter which is a small positive number and needs to be pre-determined. By this constraint, the average activation of each hidden neuron j will be trained to be close to zero. Roughly speaking, the hidden neurons are mostly "inactive". To achieve this, the following extra penalty term is introduced to the optimization objective

function that penalizes $\widehat{\rho}$ if it deviates significantly from ρ:

$$KL(\rho||\widehat{\rho_j}) = \rho \log(\frac{\rho}{\widehat{\rho_j}}) + (1-\rho)\log(\frac{1-\rho}{1-\widehat{\rho_j}}) \tag{3.3}$$

where $KL(\cdot)$ is the Kullback-Leibler (KL) divergence [52] which measures the similarity between two distributions ρ and $\widehat{\rho_j}$. As $\widehat{\rho_j}$ diverges from ρ, the KL divergence increases monotonically. Hence the sparsity penalty term can be formulated as

$$J_{sparse}(W, \mathbf{b}) = \sum_{j=1}^{r} KL(\rho||\widehat{\rho_j}) \tag{3.4}$$

where r is the number of neurons in a hidden layer and the index j is summing over all the hidden neurons in that hidden layer of the network. The sparse autoencoders are utilized as the generic building blocks of the proposed model.

3.2.2 Parallel Sparse Dimensionality Reduction

The main objective of this component is to perform feature extraction with dimensionality reduction from the multi-scale datasets. The original input is firstly divided into several groups as follows:

$$Natural frequencies : \mathbf{q}^r = [q_1^r, q_2^r, ..., q_n^r]^T \tag{3.5}$$

$$Mode shapes : \mathbf{m}^{q_1^r} = [m_1^{q_1^r}, ..., m_j^{q_1^r}]^T, \mathbf{m}^{q_2^r} = [m_1^{q_2^r}, ..., m_j^{q_2^r}]^T, ..., \mathbf{m}^{q_n^r} = [m_1^{q_n^r}, m_2^{q_n^r}, ..., m_j^{q_n^r}]^T \tag{3.6}$$

We use the vector $\mathbf{q}$ to denote the natural frequencies input subset, where n is the number of frequencies while r is the index of samples. $\mathbf{m}^{q_1^r}, \mathbf{m}^{q_2^r} ..., \mathbf{m}^{q_n^r}$ are the mode shape groups associated with the frequencies $q_1^r, q_2^r ..., q_n^r$. Each mode shape subset has j number of parameters. The original input has been divided into $n+1$ groups. For simplicity, we use $\mathbf{c}_{ori}^r$ to denote each input subset of r^{th} sample in the generic formulation.

As described in Figure 3-2, each subset of input data is fed into a stacked sparse autoencoder model with a deep architecture to perform dimensionality reduction in parallel. Layer-wise pre-training is conducted for the stacked deep autoencoder model of each parallel block. The hidden features from each encoder is used as the input to the next autoencoder. Only the encoding layers of the stacked deep autoencoders are shown in Figure 3-2. The

dimensionaity of each feature subset is compressed layer by layer, forcing the network to learn the most intrinsic representations of the input data. Then the features from each subset are concatenated into a new vector and used as the input to the next relationship learning component.

The overall cost function of each layer of the stacked sparse autoencoder in the parallel dimensionality reduction component is defined as follows:

$$J^p_{cost}(W, \mathbf{b}) = J^p_{MSE}(W, \mathbf{b}) + \lambda J^p_{weight}(W, \mathbf{b}) + \beta J^p_{sparse}(W, \mathbf{b}) \tag{3.7}$$

which includes a reconstruction loss term, a sparsity penalty term and a l_2-weight decay term. W and $\mathbf{b}$ denote the weights and bias of the network, respectively.

Mean Squared Error(MSE) is employed as the metric to measure the difference between the original input and the reconstructed input. Thus, the reconstruction loss function of the p^{th} layer of each stacked autoencoder model can be defined as:

$$J^p_{MSE}(W, \mathbf{b}) = \sum_{n=1}^{N} ||\mathbf{h}^r_{p-1} - g_p(f_p(\mathbf{h}^r_{p-1}))||^2_2 \tag{3.8}$$

where $p = \{1, ...k\}$ with k defined as the last layer of each stacked deep autoencoder. N is the number of training samples while r is the index of each training sample. $g(\cdot)$ and $f(\cdot)$ represent the encoder and decoder of each pre-trained autoencoder model respectively. Inspired by the work of SAF [86], the encoder uses Rectified Linear Units (ReLU), which indirectly controls the sparsity of the model by achieving actual zeros in the hidden representations. While the decoder function is set to be 'linear' to reconstruct the real values of the input. A low dimensional representation, denoted by $\mathbf{h}_{p-1}$, of the r^{th} training sample from $(p-1)^{th}$ layer is used as the input to the next layer of the autoencoder, *i.e.*, the p^{th} layer of the stacked deep autoencoder. Particularly, the input to the first layer is $\mathbf{h}^r_0 = \mathbf{c}^r_{ori}$.

The sparsity penalty term defined in Section 3.2.1, which has a better de-nosing ability, is employed to learn robust representations of the input. In addition, to avoid overfitting, a l_2-weight decay term is enforced during training to shrink the weights of the model, which is defined as follows:

$$J^p_{weight}(W, \mathbf{b}) = \frac{1}{2} \sum_{l=p-1}^{p} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w^l_{ji})^2 \tag{3.9}$$

where s_l represents the number of neurons in the l^{th} layer and w^l_{ji} is an element of W^l.

However, applying too strong a L2-weight decay or sparsity regularization may hurt the inference performance, *i.e.*, cause underfitting, as they reduce the effective capacity of the model. Therefore, hyper-paramaters λ and β are applied in the overall cost function to adjust the weight decay term and sparsity regularization term, respectively. It is worth mentioning that dimensionality reduction is applied on each subset separately in the proposed model for simplicity, with the assumption that all stacked autoencoders have the same number of layers.

In summary, an unsupervised layer-wise pre-training, based on sparse autoencoders, is performed for all the layers of every stacked model in the parallel architecture. The sparse hidden representation of each pre-trained autoencoder is taken out as the input to the next layer. Finally, the features learnt from the k^{th} layer of each model are concatenated into a new input vector $\mathbf{c}^r_{new}$ and fed into a nonlinear relationship learning component as described in the next section.

3.2.3 Relationship Learning

A stacked autoencoder with multiple nonlinear layers followed by a linear output layer is formed as the relationship learning component in Figure 3-2.

$\mathbf{c}^r_{new}$ is learned to be a better representation of the original input and it will be fed into the relationship learning component for learning the nonlinear regression against the output $\mathbf{o}^r$ (labelled stiffness elements).

A supervised greedy layer-wise pre-training scheme is performed at this stage. Each layer is pre-trained as the encoding layer of a simple autoencoder, with the input being the output of the previous layers. The output (decoding) layer is then discarded, and the pre-trained layers are used for initialization. It is expected that the pre-training scheme will yield a better representation which is closer to the optimal solution. The cost function of the sth layer for nonlinear relationship learning is defined as follows.

$$J^s_{cost}(W, \mathbf{b}) = J^s_{MSE}(W, \mathbf{b}) + \lambda J^s_{weight}(W, \mathbf{b}) \tag{3.10}$$

where

$$J^s_{MSE}(W, \mathbf{b}) = \sum_{n=1}^{N} ||\mathbf{o}^r - g_s(f_s(\mathbf{h}^r_{s-1}))||^2_2. \tag{3.11}$$

The same L2-weight decay term defined in Eq.(3.9) is used. Similarly, the reconstruction

loss of each layer is defined by a MSE term as given in Eq.(3.11), where $s = \{1, ..., m\}$ expresses m layers in the relationship learning component. $g(\cdot)$ and $f(\cdot)$ are the encoder and decoder mapping functions respectively as mentioned in previous section. Here, the hyperbolic tangent (tanh) function is selected as the encoder activation to produce nonlinear representations of features. $\mathbf{h}_{s-1}^r$ is the hidden feature vector extracted from $(s-1)^{th}$ relationship learning layer for the rth sample where $\mathbf{h}_0^r = \mathbf{c}_{new}^r$. Eq.(3.10) minimizes the difference between the learnt features and the structural stiffness. The hidden feature vector, which is in a lower dimensional representation of our original input from each previous layer, compressed gradually and trained to regress to the target.

3.2.4 Training and Fine-tuning

When the pre-training of relationship learning is completed, all the encoding layers are stacked together followed by a linear regression output layer forming the stacked deep autoencoder model for fine-tuning. The final objective function is defined as:

$$J_{cost}^F(W, \mathbf{b}) = J_{MSE}^F(W, \mathbf{b}) + \lambda J_{weight}^F(W, \mathbf{b}) \tag{3.12}$$

$$J_{MSE}^F(W, \mathbf{b}) = \sum_{n=1}^{N} ||\mathbf{o}^r - p(\mathbf{c}_{new}^r)||_2^2 \tag{3.13}$$

where $p(\mathbf{c}_{new}^r) = g_s(f_s(f_{s-1}(...(\mathbf{c}_{new}^r))))$ is the predicted output vector through all layers in the relationship learning model as presented in Figure 3-2.

In summary, a parallel sparse autoencoder framework is developed to identify the damage locations and severities of structures. It is capable to deal with input data with multiple magnitude scales or different physical meanings. In this study, the frequencies and mode shapes are trained as input to the proposed framework while the stiffness parameters are the output vector. The relationship between input and output is learned via two stages as mentioned in Sections 3.2.2 and 3.2.3 respectively. For a fair comparison with the state-of-the-art method SAF [86], the same optimization method is used in training the proposed framework. Both the pre-training and fine-tuning steps are conducted using a full batch-scaled conjugate gradient algorithm [75]. To evaluate the performance of the proposed approach, experiments are conducted and presented in the next section.

3.3 Experiments

Experimental studies including the numerical model, data generation, data pre-processing and the evaluation of the proposed framework are conducted in this study. Details of the numerical model are described in Section 2.5.1, and the rest of the work is presented in this section. A seven-storey steel frame structure has been fabricated in the laboratory, as illustrated in Figure 2-8. The finite element model of this structure is shown in Figure 2-9. With the consideration of the uncertainties in the finite element modelling and measurement noise effect in the data, the accuracy and efficiency of the proposed approach are examined through the simulated data generated from the numerical finite element model.

3.3.1 Data Generation

Modal analysis is conducted using the finite element model described in Section 2.5.1 to generate the training dataset for both the input and output. As mentioned before, seven frequencies and their corresponding mode shape parameters at 14 beam-column joints are measured and defined as the input. Seventy elemental stiffness parameters are generated as the output. The output is normalized to be in the range between 0 and 1, where 0 represents the fully damaged state and 1 represents the intact state of structure elements. For example, if the stiffness parameter of an element is equal to 0.9, it indicates that there is 10% stiffness reduction at this element.

Both single damage and multiple damage cases are considered in the 70-elements model. For single damage cases, 30 data instances are generated for each element from the baseline model based on various stiffness parameters of this element from 0.7 to 1 in increments of 0.01 (*i.e.*, $0.7, 0.71, \cdots, 0.99, 1$), leaving the other elements undamaged. In total, 2100 data instances are generated by introducing local damages on each element of the 70 elements. For multiple damage cases, 10,300 data instances are generated where two or more elements have damages with various stiffness parameters from 0.7 to 1 with increment of 0.01. Overall, 12,400 baseline data instances are generated based on the finite element model for training and validation.

Apart from the clean baseline dataset, a noise dataset with measurement noises and modelling uncertainties to further investigate the effectiveness and robustness of the proposed parallel structured model. For the noise dataset, based on the fact that the measurement

of frequencies is usually more accurate than the measurement of mode shapes as reported in [50], 1% and 5% Gaussian noises are added to the seven frequencies and the associated mode shapes respectively. Besides, 1% uncertainty is added in the stiffness parameters to simulate the finite element modelling errors. With the noise data being used, the trained model is expected to be more robust in predicting unknown data with measurement noises and uncertainties. Both the baseline dataset and the noise dataset are used to test the performance of the proposed model with comparison to the state-of-the-art models in our experiments.

3.3.2 Data Pre-Processing

Since each mode shape is associated with a specific natural frequency, and the frequency features and the mode shape features are measured in different scales, we split the feature of the data, which includes seven frequencies and mode shape parameters, into eight groups and separately normalise them to be in the range of $[0, 1]$.The first subset includes all the seven frequencies, the other seven groups are for the seven mode shapes each associated with one of the seven frequencies.

For the output vector, each element lies in the range between 0 and 1, where 0 denotes the fully damaged state and 1 denotes the intact state. The stiffness parameters are normalized to be in the range of $[-1, 1]$.

3.3.3 Performance Evaluation

The performance of the proposed approach is evaluated with both the baseline dataset and the noise dataset described in Section 3.3.1. The performance comparison against the state-of-the-art models, for example, "Sparse Autoencoder Framework (SAF)" [86] , is also conducted to show the advantage of the proposed approach. The pre-processed datasets are randomly split into three subsets, namely, 70% for training, 15% for validation, and another 15% for testing. Two performance metrics, the Mean Squared Error (MSE) value and the Regression value (R-value) are selected to evaluate the quality of the damage predictions of the proposed parallel model with comparison to the state-of-the-art models on the testing dataset. In particular, the MSE measures the distance between the estimated outputs of the proposed model and the ground truth. Besides, a linear regression between estimated output and ground truth is fitted. R-value (between 0 and 1) evaluates the scatter of the data points

around the fitted regression line. The higher R-value represents smaller differences between the estimated outputs and the ground truth, *i.e.*, the more accurate of the proposed method. These two metrics are used throughout this thesis.

Model architecture. As shown in Figure 3-2, the proposed parallel autoencoder framework (Para-AF) consists of two main components: a parallel architecture-based dimensionality reduction component followed by a relationship learning component. In this study, to deal with the eight groups of input features individually, eight parallel blocks are used in the dimensionality reduction component, where each block is a sparse autoencoder with one single hidden layer. As we implement sparse autoencoders in this component, the number of hidden neurons is chosen carefully to allow the model to have more capacity for feature learning. Nine hidden neurons are used in the 1^{st} block for the frequencies while thirteen hidden neurons are used in the rest of seven blocks for the mode shapes.

After learning sparse representations of the eight groups of input features, we concatenate these sparse representations as a vector to be fed as the input to the relationship learning component. This component is basically a stacked autoencoder model that aims to learn the underlying mapping between the input and output. Five hidden layers of having 90, 85, 80, 75 and 70 neurons respectively are used in this model. By using decreasing number of hidden neurons in the hidden layers, the input vector is gradually compressed and regressed to the output. This model is named Para-AF in the evaluation results below. To perform a fair comparison with the state-of-the-art method, *e.g.*, SAF, a simpler proposed parallel model, namely Para-AF-0, is evaluated in this study as well. The same number of hidden layers and neurons (90-80-70) as the SAF is used in the relationship learning component. The performance evaluation results are shown below.

MSE and R-value. As shown in Table 3.1, for the baseline dataset, MSE values obtained from SAF, Para-AF-0 and Para-AF are 2.9×10^{-5}, 1.8×10^{-5}, and 1.3×10^{-5}, respectively. Para-AF-0 has marginally improved the performance over SAF with a smaller MSE value for the baseline dataset. Meanwhile, for the noise dataset, we can observe a 3% increment in the R-value of Para-AF-0 versus SAF. They indicate that the proposed parallel dimensionality reduction has achieved the expected effectiveness by extracting features separately from multi-scale datasets. In addition, a significant increment (around 8%) in R-value is observed in Para-AF-0 against Para-AF, in terms of the effectiveness of using more relationship learning layers and deeper neural networks.For both datasets, the proposed

method constantly outperforms the state-of-the-art method, which shows both effectiveness and robustness for damage identification.

Table 3.1: Evaluation results for SAF and the proposed methods.

Methods	**Baseline Dataset**		**Noise Dataset**	
	MSE	**R-Value**	**MSE**	**R-Value**
SAF	2.9×10^{-5}	0.993	3.2×10^{-4}	0.792
Para-AF-0	1.8×10^{-5}	0.994	3.0×10^{-4}	0.823
Para-AF	1.3×10^{-5}	0.996	1.7×10^{-4}	0.901

Typical damage identification results. As the experiment conducted on the noise dataset is more challenging than the baseline one, the results of the noise dataset will be further explored to show the advantage of the proposed method. To further evaluate the quality of damage identification in terms of both magnitudes and locations, the predictions of a single-element damage case and a multiple-element damage case randomly selected from the testing dataset are shown in Figure 3-3 and Figure 3-4 respectively.

For the single-element damage case, it is observed that SAF can predict the true location of the damage but fail to identify the true magnitude of the damage. The proposed method performs better in damage identification with a very closed identified stiffness reduction value against the true value. Moreover, SAF produces a significant false identification at the $13th$ element of the structure. Therefore, the proposed method has a higher accuracy in damage identification in terms of location and severity.

An example of multiple-element damage identification result is shown in Figure 3-4. It can be observed that the proposed approach performs better consistently, with all the damage locations accurately detected. Besides, the identified stiffness reductions are very close to the true values with very small false identifications, while SAF is not working very well in identifying multiple damages with some significant false identifications appeared at non-damage locations.

Damage identification results from the above results demonstrate clearly the high accuracy and robustness of using the proposed method in structural damage identification, even when the measurement noises and modeling uncertainties are introduced. The improvement of the proposed method is also demonstrated, supported by the comparisons with the latest previous study [86] based on SAF.

3.4 Summary

In this chapter, we presented a parallel sparse autoencoder framework (Para-AF) for structural damage identification, which is feasible to deal with multi-scale datasets. This framework consists of two main components: a parallel architecture-based dimensionality reduction component followed by a relationship learning component. At first, it achieves dimensionality reduction and feature extraction for the frequency data and mode shape data separately. The latent features of frequency data and mode shape data are learned and concatenated into one vector. Later, the concatenated feature vector is utilized as the new input to the stacked autoencoder for relationship learning. Experiments are conducted on both the clean dataset and the noise dataset to evaluate the performance of the proposed method. Compared to the state-of-the-art approach SAF, the proposed Para-AF provides a sound process to normalize and extract features separately from frequencies and mode shapes, which significantly improves the effectiveness and robustness of structural damage identification.

SAF and Para-AF are both two-component frameworks. The two components are trained separately, thus may result in sub-optimal results. An end-to-end learning framework for SHM is presented in the next chapter.

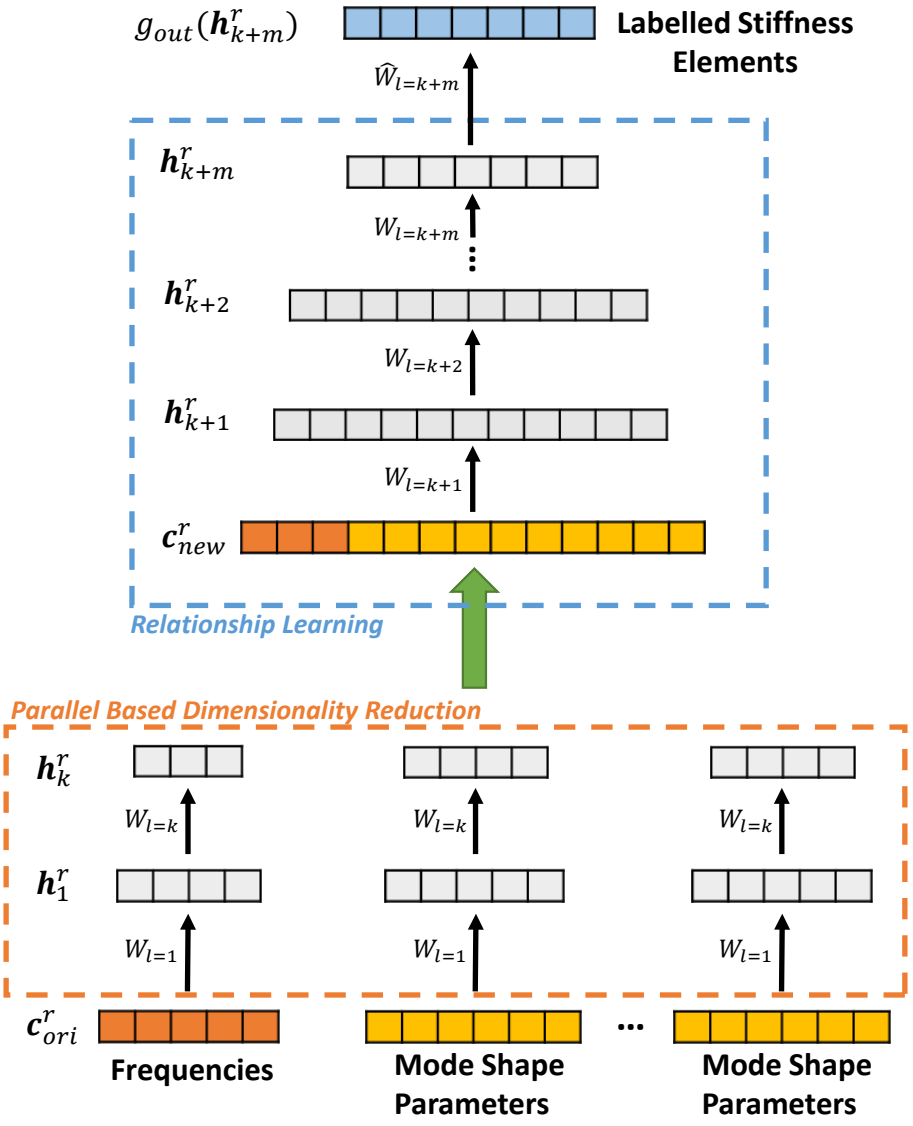

Figure 3-2: The proposed parallel sparse autoencoder framework.

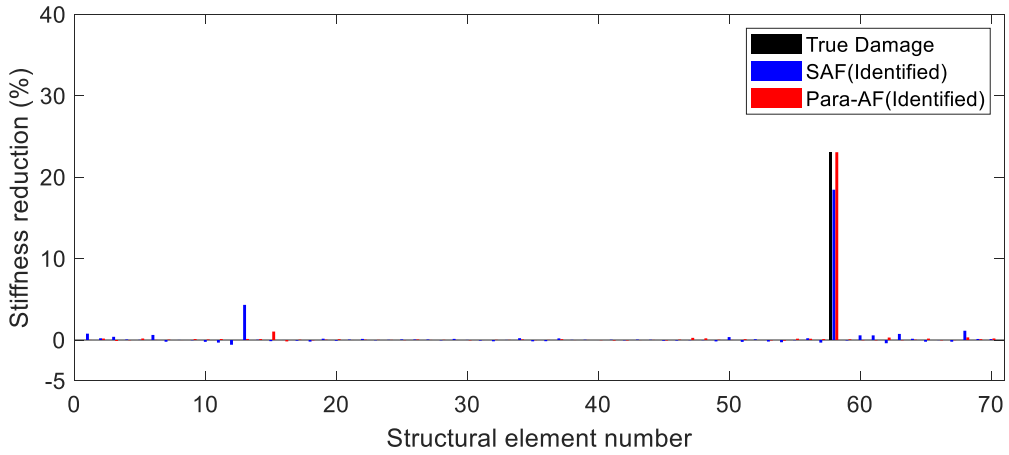

Figure 3-3: Damage identification result for the single-element damage case of the structure.

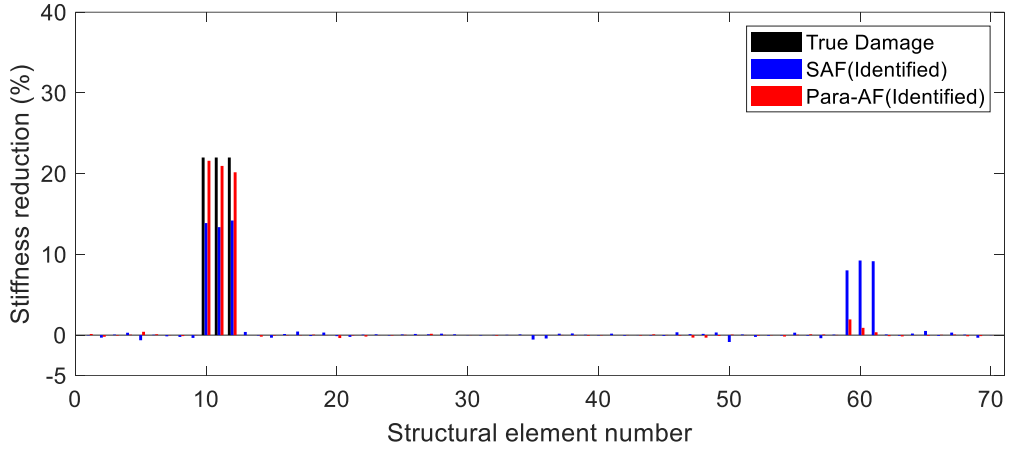

Figure 3-4: Damage identification result for the multiple-element damage case of the structure.

Chapter 4

Deep Residual Network Framework for Structural Health Monitoring

In Chapter 3, a two-component framework based on autoencoders, namely, dimensionality reduction and relationship learning, is introduced for a multivariate regression task. The training process involves greedy layer-wise pre-training of each component and fine-tuning, resulting in high computational cost. Also, the training of the two components are separated and thus results in sub-optimal solutions.

Another important type of neural networks, known as Convolutional Neural Networks (CNNs), is capable of constructing an end-to-end learning method for regression. Two CNN-based frameworks, a deep residual network framework and a densely-connected convolutional network framework, designed for the application of structural health monitoring are introduced in the following two chapters. Both of them utilize the convolutional layers to scale high dimensional data. Besides, they have further improved the representational power with special architectures, comparing to the standard CNNs.

In this chapter, we present the deep residual network framework for structural damage identification.

[0]This chapter is reprinted, with permission, from [Wang, R.,Chencho, An, S., Li, J., Li, L., Hao, H., & Liu, W. Deep residual network framework for structural health monitoring. *Structural Health Monitoring*, p.1475921720918378. © 2020 SAGE. DOI: 10.1177/1475921720918378].

4.1 Introduction

CNNs [34] are specialized neural networks that use convolution in place of general matrix multiplications in the layers. It leverages three key ideas, including sparse interactions, parameter sharing, and equivariant representations, which can be used to effectively extract features from images. Besides, due to the special architecture of CNNs, it is scalable to big datasets and more computationally efficient than traditional Artificial Neural Networks (ANNs). This results in success in optimizing deep networks with gradient based back-propagation algorithms [90]. Deep CNNs have achieved tremendous success on various tasks in the domain of computer vision, such as image classification [59, 95], object detection [32], image segmentation [18], and other visual understanding problems.

Recently, deep CNNs have also been explored in the research domain of structural health monitoring (SHM), particularly for vibration-based damage detection and system identification applications [16, 111]. Lin *et al.* [70] designed a deep CNN model to automatically extract features from low-level sensor data and identify damage locations. Osama *et al.* [2] presented a real-time structural damage detection method using a one-dimensional CNN model. Later, a two-dimensional CNN model was introduced in [54] for structural condition assessment using vibration response data. It is important to design an appropriate network for structural damage quantification which is mainly trained on signal-based data rather than visual images. Besides, the measurement noise effect and uncertainty effect on the measurement increase the problem complexity and should be considered as well in designing suitable networks. Therefore, the architecture of the network should be appropriately designed to perform a robust feature extraction that reflects the true damages of structures.

Since AlexNet [59] was proposed in 2012, the popular trend of architecture design has become to increase the depth of CNNs by repeatedly stacking convolutional building blocks. Typically, the building blocks are composed of convolutional layers, non-linear activations, and pooling layers. As deep CNNs integrate low level features into high level features gradually in its depth, stacking more layers enriches the "levels" of features. VGG [95] and GoogLeNet [97] demonstrated the crucial importance of deep architectures in performance improvement for image classification tasks. However, as the depth of the network increases to a certain level, the accuracy will get saturated due to the gradient vanishing or exploding problems. Gradients vanishing is a well-known issue in optimizing deep neural networks.

When back-propagation is conducted, the gradients of the loss with respect to the weights in each layer are calculated and flows back through the network. Due to the repeated multiplication or convolution with small weights in the network, the gradients tend to be ineffectively small in earlier layers. As a result, the performance degrades when adding too many layers.

He *et al.* [38] introduced Residual Networks (ResNets) to tackle the above problems in training very deep CNNs and achieved best performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification competition in 2015. Skip connections were adopted in the architecture, which allowed an alternative path for the gradient to flow through. As a result, the entire network can still be trained with back-propagation algorithms but in much deeper architectures.

The residual learning concept proposed in [38] has also been adopted recently along with a multi-scale module for damage detection of a simple beam [37]. However, rare work based on purely deep residual network for vibration-based SHM applications, *e.g.*damage identification on a more complex structure, has been conducted. Motivated by the above observations, a novel structural health monitoring framework is proposed based on deep residual networks to detect the damage locations and quantify the damage severities of structures effectively and efficiently. This framework is composed of purely residual blocks which operate as feature extractors, and a fully connected layer as a regressor. It learns the damage related features from the vibration characteristics such as mode shapes, and maps them into the damage index labels, *e.g.*, stiffness reductions of structures.

To evaluate the efficacy and robustness of the proposed framework, comprehensive studies are conducted including both numerical and experimental studies. The remainder of this chapter is organized as follows: Section 4.2 introduces core ideas of the ResNets, including residual learning and various types of residual blocks; Section 4.3 describes the theoretical development and the overall architecture of the proposed framework where the pre-activation residual block is adopted as the essential component. Sections 4.4 and 4.5 evaluate the performances of the proposed framework with numerical studies and experimental validation respectively. Finally, Section 4.6 summarises this chapter.

4.2 Core ideas of Residual Networks

Deep convolutional neural networks have shown their high capacities to improve feature representation progressively with depth, and ultimately learn the underlying mapping between the input and output. However, performance will be degraded due to the gradient vanishing/exploding issues when the depth of network increases to a certain level. To address this problem, a deep residual network framework by incorporating residual learning in the typical deep convolutional network is introduced in [38]. The core idea of the residual network is to explicitly let the layers of the original CNN fit a residual mapping rather than the underlying mapping between input and output, which eases the training of deep networks and makes it possible to efficiently train a very deep network. To achieve this, shortcut connections are introduced in the deep network that skips one or more layers, which forms the building blocks of the residual network, often referred to as residual blocks. We present the intuition of residual learning and the architecture of the residual block in the following sections.

4.2.1 Residual Learning

Considering an input x which goes through a few stacked nonlinear layers of the traditional CNN, as shown in Figure 4-1(a), the desired underlying mapping can be denoted as $H(x)$. By introducing a shortcut connection as shown in Figure 4-1(b), the network can directly use an identity mapping x, whereas the stacked nonlinear layers can fit a so-called residual mapping $F(x) = H(x) - x$ assuming that $H(x)$ and x have the same dimension. The original mapping $H(x)$ can be reformatted as $H(x) = F(x) + x$. In other words, the layers in a traditional CNN are learning the true output $H(x)$, while the layers in a ResNet are learning the residual $F(x)$. It is observed that it is easier to optimize the residual mapping $F(x)$ than the original mapping in practice.

Formally, the residual block presented in Figure 4-1 can be expressed in a general form

$$y_l = X_l + F_l(x_l, W_l) \tag{4.1}$$

$$X_{l+1} = f(y_l) \tag{4.2}$$

where X_{l+1} and X_l are the input and output of the lth block in the network. $F(\cdot)$ is the residual function, and W denotes the parameters to be learned of the block. $f(\cdot)$ is a

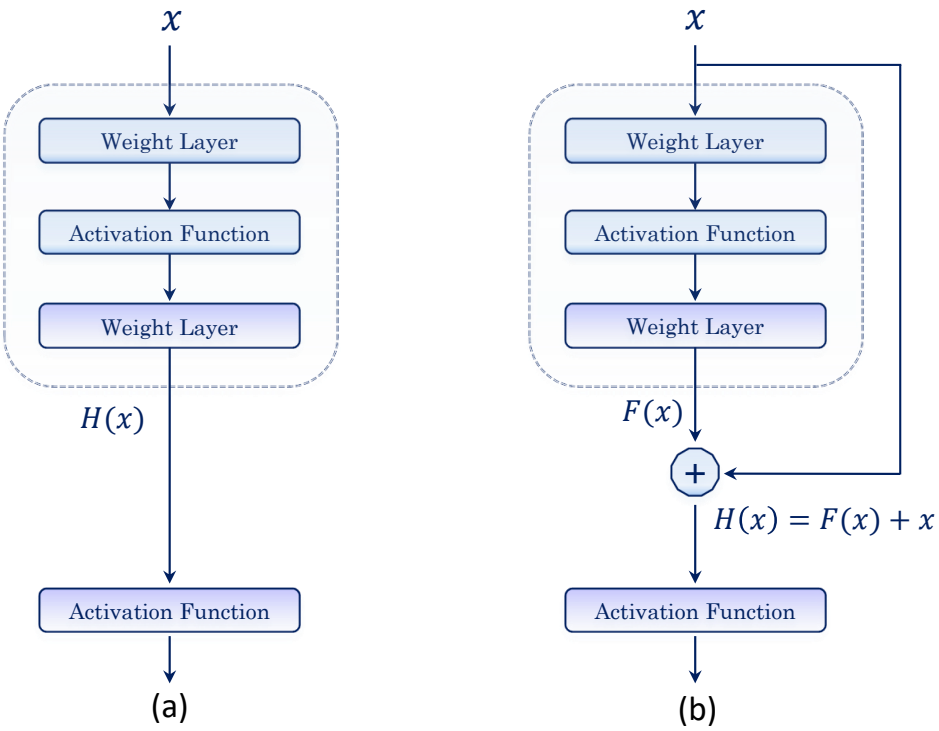

Figure 4-1: Comparison between CNN and ResNet. (a) A regular block used in CNN; and (b) A residual block used in ResNet.

nonlinear activation (*e.g.*, the ReLU function). By sequentially stacking residual blocks, one can construct a deep residual network. The detailed architecture of the residual block and its variants are described in the next paragraph.

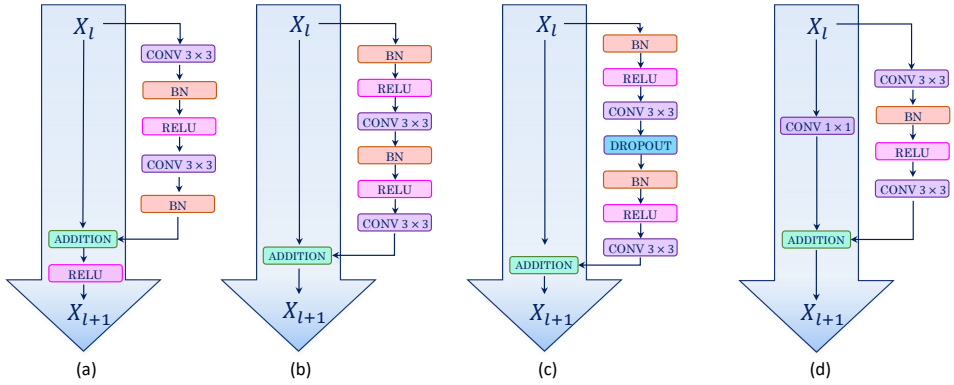

Figure 4-2: Variants of residual blocks.

4.2.2 Variants of residual blocks

A typical residual block is shown in Figure 4-2(a). It was proposed in [38] and follows the design of VGG's full 3×3 convolutional layer. It consists of two consecutive 3×3 convolutional layers, with each layer followed by batch normalization and a ReLU activation. The input goes through both the residual path and the identity mapping path, and the outputs of these paths are added together before the next ReLU activation. This is the original residual block that forms the first deep residual network. In [39], the authors refined the original residual block and revealed that if $f(y_l)$ in Eq.4.2 is also an identity mapping, that is $f(y_l) = y_l$ in the network, the signal could directly propagate from one block to any other block, in both forward and backward passes. In other words, the signal could be propagated directly through the entire network. This is realized by constructing a new variant of the residual block, known as the pre-activation residual block. Figure 4-2(b) shows the pre-activation residual block where batch normalization and ReLU precede each convolution. As experimentally shown in [39], the pre-activation residual block makes training easier and improves generalization. It has subsequently become the most popular one used in the applications of ResNets [110]. In this chapter, the pre-activation residual block is used in the proposed framework and is referred to as the 'basic' residual block as shown in Figure 4-2(b). Another variant of residual blocks introduced in [110] is also

investigated in this chapter for structural damage identification, namely 'basic-dropout' shown in Figure 4-2(c). In this block, a dropout layer is inserted in-between the two weight layers to regularize training and prevent overfitting when a large number of filters are used. Figure 4-2(d) shows a special variant of residual blocks called 'expand' block [110], which inserts 1×1 convolutional layer in the shortcut to widen the network architecture by adding more filters.

4.3 The Proposed Approach

This section describes the theoretical development and overall architecture of the proposed deep residual network framework for structural damage identification. It is a known fact that the changes in structural vibration characteristics (*e.g.*, modal information) are related to the physical properties of structures such as element stiffness. In this study, the changes in the modal information are exploited to identify the damages of structures. Since the state-of-the-art structural damage identification results are reported in [86] using a sparse autoencoder framework ('SAF'), a comprehensive comparison is conducted on the same experiment scenarios to demonstrate the superiority of the proposed approach. Moreover, the proposed deep ResNet is also compared with its plain model by removing all the shortcut connections but still having the same number of parameters. The plain model is actually a deep convolutional network as it is composed of convolutional building blocks. It is referred to as a 'Plain CNN' model in this chapter. A shallower version of convolutional networks is also investigated for comparison in the numerical studies to illustrate the importance of increasing the depth of networks in this study. It is referred as a 'Plain CNN-S' model. The main advantages of the proposed ResNet framework can be summarized as follows: 1) It integrates the feature extraction and damage identification processes into an end-to-end learning system. This is more efficient comparing to 'SAF' because no pre-training and fine-tuning are required in training; 2) The relationship learning component of 'SAF' uses conventional fully-connected layers, which requires high computational costs for training with a large amount of training data. On the other hand, the ResNet is composed of convolutional layers, which employs sparsely-connectivity between adjacent layers and makes the training much easier and generalization much better; 3) The proposed framework built upon convolutional layers is feasible to deal with any input signal of any dimension; and

4) It is hard for 'SAF' and CNN to go very deep due to the gradient vanishing problem, whereas the proposed deep ResNet framework can easily enjoy the gains from the increased depth.

In the numerical tests, experiments with the same scenarios as those in [86] considering the uncertainties in the structural finite element model and measurement noise in the data are conducted to identify damages of a seven-storey steel frame structure which is described in Section 2.5.1. The modal information such as mode shapes is used in this study. Experimental verification is also conducted on a laboratory T-section beam (which is described in 2.5.2) of a reinforced concrete bridge model to further demonstrate the accuracy of the proposed approach for damage identification. In this experiment, mode shape information is used as well. The overall architecture of the proposed framework is presented in the next section.

4.3.1 Architecture of the proposed framework

In this study, a deep ResNet framework is proposed based on 'basic' blocks and 'expand' blocks. For some experiment scenarios, a larger number of filters (*i.e.*, wider architecture) and a deeper architecture are selected in designing the deep residual network to deal with more complicated and difficult tasks, for instance, with noise introduced in the data. Therefore, we replace 'basic' blocks with 'basic-dropout' in those more complex models. A generic framework is proposed for all the experiment scenarios, but the depths and widths are selected separately. For simplicity, two factors are defined when we describe the architecture of the proposed networks, that is, the depth factor N and width factor K.

The overall architecture of the proposed framework is a stacked network of residual blocks, on top of the global average layer and final objective layer, namely the regression layer. Details of the proposed framework based on mode shape data is shown in Table 4.1. There are in total five convolution groups and the network is essentially built based on convolutions. These groups are stacked sequentially and perform feature extraction progressively along with the depth. In the first group, an initial convolutional block followed by an 'expand' residual block is defined. The initial convolutional block is a typical building block composed of convolution layers, BN and ReLU activations. The first 'expand' residual block expands the network architecture by K times without reducing the feature dimension. In the 2^{nd} to 4^{th} groups, a 'basic' residual block and an 'expand' residual block are introduced.

The network dimension expands along with these groups and the feature dimension reduces at the same time. The last convolution group performs the last stage feature extraction with a 'basic' block of the greatest number of filters. Finally, a global averaging pooling (GAP) layer sums out each feature map, and enforces it to be the confidence maps for damage predictions. It should be noted that there is no fully connected layers and convolutional layers with large number of parameters used before the final objective layer.

Table 4.1: Architecture of the proposed framework for numerical study.

Group name	**Block type**	**[kernel size, # of filters]**	**Output size**
CONV 1	init conv	$\begin{bmatrix} 2\times 2, & 16 \end{bmatrix}$	(7, 14)
CONV 2	expand	$\begin{bmatrix} 2\times 2, & 16\times K \\ 2\times 2, & 16\times K \end{bmatrix}$	
	basic	$\begin{bmatrix} 2\times 2, & 16\times K \\ 2\times 2, & 16\times K \end{bmatrix}\times N$	(4, 7)
CONV 3	expand	$\begin{bmatrix} 2\times 2, & 32\times K \\ 2\times 2, & 32\times K \end{bmatrix}$	
	basic	$\begin{bmatrix} 2\times 2, & 32\times K \\ 2\times 2, & 32\times K \end{bmatrix}\times N$	(2, 4)
CONV 4	expand	$\begin{bmatrix} 2\times 2, & 64\times K \\ 2\times 2, & 64\times K \end{bmatrix}$	
	basic	$\begin{bmatrix} 2\times 2, & 64\times K \\ 2\times 2, & 64\times K \end{bmatrix}\times N$	(1, 2)
CONV 5	expand	$\begin{bmatrix} 2\times 2, & 128\times K \\ 2\times 2, & 128\times K \end{bmatrix}$	
	basic	$\begin{bmatrix} 2\times 2, & 128\times K \\ 2\times 2, & 128\times K \end{bmatrix}\times N$	(1, 2)
GAP			(1, 1)

In the numerical study, seven mode shapes of the frame structure are included and each mode shape consists of 14 values, thus the input dimension is 7×14. A small kernel size 2×2 is selected due to the small dimension of input. The output size in Table 4.1 describes the dimension changes of the input features along with the depth.

In the experimental study, three mode shapes of the T-section beam structure are included and each mode shape consists of 7 values, thus the input shape is 3×7. A small kernel size 2×2 is selected due to the small dimension of input. Because the input dimension of the T-section structure is smaller than that of the frame structure, a shallower network with four convolution groups, as shown in Table 4.2, is used. Similarly, the output size describes

the dimension reductions of the input features along with the depth.

Table 4.2: Architecture of the proposed framework for experimental study.

Group name	**Block type**	**[kernel size, # of filters]**	**Output size**
CONV 1	init conv	$\begin{bmatrix} 2\times 2, & 16 \end{bmatrix}$	(3, 7)
	expand	$\begin{bmatrix} 2\times 2, & 16\times K \\ 2\times 2, & 16\times K \end{bmatrix}$	
CONV 2	basic	$\begin{bmatrix} 2\times 2, & 16\times K \\ 2\times 2, & 16\times K \end{bmatrix} \times N$	(3, 7)
	expand	$\begin{bmatrix} 2\times 2, & 32\times K \\ 2\times 2, & 32\times K \end{bmatrix}$	
CONV 3	basic	$\begin{bmatrix} 2\times 2, & 32\times K \\ 2\times 2, & 32\times K \end{bmatrix} \times N$	(2, 4)
	expand	$\begin{bmatrix} 2\times 2, & 64\times K \\ 2\times 2, & 64\times K \end{bmatrix}$	
CONV 4	basic	$\begin{bmatrix} 2\times 2, & 64\times K \\ 2\times 2, & 64\times K \end{bmatrix} \times N$	(1, 2)
GAP			(1, 1)

Tables 4.1 and 4.2 summarize the architectures of the proposed frameworks. The convolution groups aim at discovering the damage-related patterns from the input data, *i.e.*, extracting the robust feature maps. Then the GAP layer averages all the feature maps, outputs the damage confidence maps and feeds them into the final objective layer.

4.3.2 Objective layer

The objective layer is also called the output layer of the neural networks. It is generally a fully-connected layer with a specific activation function that produces the final results to serve a particular task. Moreover, the number of neurons and activation of the output layer should be selected appropriately. For the damage identification task in this study, we have 70 elements (categories) in the output where each of them indicates the real damage level (but not the probabilities) at a small fraction of the seven-storey frame structure. Therefore, this is a regression problem. It is common to set a nonlinear activation function such as sigmoid or tangent hyperbolic (tanh) for the output layer to perform regression. Here, the tanh function $\sigma(x) = \frac{e^{2x}-1}{e^{2x}+1}$ is used in this study. Moreover, the mean squared error (MSE) loss is chosen as the cost function that minimizes the difference between the damage predictions and the true labels. The cost function can be described as $Cost(y, \bar{y}) = \frac{1}{m}\sum_{i=1}^{m} ||y_i - \bar{y}_i||_2^2$

, where m is the dimension of the output label y. To optimize the network parameters, the most popular optimizer called Adam is selected [56] in our experiments. It is known as an adaptive learning rate method, which is commonly used as the default optimization method in training of deep neural networks.

In summary, a generic deep ResNet framework is developed to detect and identify the damages of structures. It should be noted that our framework is feasible to deal with various datasets with different complexities and dimensionalities. To evaluate the performance of the proposed approach, both numerical studies and experimental studies are conducted. Numerical studies will be presented in the following section.

4.4 Numerical Studies

Numerical studies, including data generation, data pre-processing and the evaluation of the proposed framework, are described in this section, and the numerical model is introduced in Section 2.5.1. With the consideration of both the uncertainties in the finite element modelling and the measurement noise effect in the data, the accuracy and efficiency of the proposed approach are examined through simulation data generated from the numerical finite element model.

4.4.1 Data Generation

Modal analysis is conducted using the finite element model to generate the training datasets for both input and output. As mentioned above, 7 frequencies and their corresponding mode shape parameters at 14 beam-column joints are calculated. Only mode shapes are used in this study. Seventy elemental stiffness parameters are generated as the ground truth. The output is normalized to be in the range between 0 and 1, where 0 represents the totally damaged state whereas 1 represents the intact state of the structure element. For an instance, if a particular stiffness parameter is equal to 0.9, it indicates that there is 10% stiffness reduction at this element.

Both single damage and multiple damage cases are investigated in this model. For single damage cases, there are totally 2100 data instances generated from the baseline model based on various stiffness parameters of each element from 0.7 to 1 with increment of 0.01 (*i.e.*, 0.7, 0.71, ..., 0.99, 1), leaving the other elements undamaged. Hence, for single element damage

cases, 30 data instances are generated for each element by introducing local damages. For multiple damage cases, 10300 data instances are generated where two or more elements may have damages. Overall, 12400 baseline data instances are generated based on the finite element model.

All these data instances are clean, that is, the noise effect on the measured mode shape and modelling uncertainty effect are not considered. To further investigate the effectiveness and robustness of the proposed deep ResNet framework, the measurement noise on mode shapes and modelling uncertainty in finite element modelling are considered as well. In total, the following four scenarios are considered in the numerical studies by using mode shape data only:

- *Scenario 1*: No measurement noise effect and modelling uncertainty is considered.
- *Scenario 2*: Measurement noise is considered. 5% Gaussian noise is added to the mode shapes of baseline dataset.
- *Scenario 3*: Modelling uncertainty is considered. 1% uncertainty is included in the stiffness parameters to simulate the finite element modelling errors.
- *Scenario 4*: Both measurement noise effect and modelling uncertainty effect are considered. 5% Gaussian noises is added to the mode shapes and 1% uncertainty is included in the stiffness parameters.

15931, 13280, and 26563 data instances for Scenarios 2, 3, and 4 are generated, respectively, by randomly adding noises to the baseline dataset. A comprehensive study of these four scenarios has been conducted. Since CNN is currently considered as the state-of-the-art method in the research of the structural health monitoring community, the performance of the proposed deep ResNet is compared with CNN. Besides, 'SAF' reported the state-of-the-art results on these datasets, the comparison between 'SAF' and the proposed approach is reported as well.

4.4.2 Data Pre-Processing

For the mode shape data, the same pre-processing proposed in [86] is used for fair comparison. Data whitening is performed on the input followed by a feature scaling process. The features are scaled into the range of $[0, 1]$. For the output vector, each element is between 0

and 1, where 0 denotes the fully damaged state while 1 denotes the intact state. Since only a few elements of each sample are in damage situation and the stiffness parameter does not vary a lot in damage cases, the stiffness parameters are normalized to the range of [-1, 1] to suit the range of 'tanh' activation function of the output. These pre-processed data will be used to evaluate the proposed approach and the performances are reported in the following sections.

4.4.3 Performance Evaluation

Performance evaluation of the proposed approach with comparisons to deep sparse autoencoder network ('SAF') and deep convolutional neural network ('Plain CNN' and 'Plain CNN-S'), is conducted in four different scenarios, as described in Section 4.4.1. It should be noted that the 'SAF' [86] achieved the state-of-the-art results for the damage identification of the frame structure. As discussed in previous sections, the main difference of the architectures between CNN and ResNet is that the latter one adopts shortcut connections in the network, making it feasible to train very deep architectures. Therefore, the 'Plain CNN' with the same number of parameters (*i.e.*, the same depth and width) as the proposed ResNet is compared to show the advantages of the proposed approach.

The pre-processed dataset is randomly split into training, validation and testing subsets in the percentages of 70%, 15% and 15% respectively. The hyper-parameters such as the learning rate, depth and width of the architecture, are selected based on the validation datasets. In order to evaluate the performance of the trained models, MSE and regression value (R-value) are employed to demonstrate the quality of the damage predictions. The MSE measures the distance between the estimated outputs and the ground truth of the damages. R-value measures the correlation between the estimated outputs and the ground truth. Hence a higher R-value or a smaller MSE indicates a better accuracy of the predictions. The details of the model architectures and the performances for each scenario are presented as follows.

Scenario 1: **No measurement noise effect and modelling uncertainty is considered**. In this scenario, the clean datasets are used without considering any noise effects. The performances of 'SAF', 'Plain CNN-S', 'Plain CNN' and the proposed approach are evaluated with MSE and R-value on the testing subsets. Table 4.3 shows the experimental results.

Table 4.3: Performance evaluation results for *Scenario 1* in the numerical study.

Methods	**MSE**	**R-value**
SAF	2.9×10^{-5}	0.993
Plain CNN-S	1.7×10^{-5}	0.996
Plain CNN	1.9×10^{-5}	0.994
The proposed framework	1.0×10^{-5}	0.999

The architecture of the proposed framework for *Scenario 1* is designed by setting the width factor $K = 1$, and the depth factor $N = 2$, which includes in total 23 convolutional layers. As no noise effect is considered in this scenario, it is the easiest case in the numerical studies. Thus fewer parameters with $K = 1$ are used in the training of the network. The 'Plain CNN' has the same number of layers as the proposed approach without the shortcut connections. The 'Plain CNN-S' has a shallower architecture compared to the 'Plain CNN' with a depth factor $N = 1$, which includes in total 17 convolutional layers. As observed in Table 4.3, the proposed method performs the best with the smallest MSE (1.0×10^{-5}) and the highest R-value (0.999). Although the R-values obtained from the CNN models are close to 1, the proposed ResNet still improves the performance slightly. Also, all CNN-based methods outperform the autoencoder based method as observed from the results, which indicates that CNN and its variations are more accurate for structural damage identification. Moreover, the results obtained from 'Plain CNN-S' and 'Plain CNN' models demonstrate that the performance degrades as the depth increases. However, the proposed ResNet still gains performance with the same depth as the 'Plain CNN'. Figure 4-3 presents the testing errors (MSE) of CNN models and the proposed ResNet versus training epochs. It further demonstrates that the proposed approach obtains the best testing results.

More challenging tasks with noisy data are presented below to further demonstrate the superior performances of the proposed approach for damage identification.

Scenario 2: **Measurement noise is considered**. In this scenario, 5% Gaussian noise is added to the mode shape of the clean dataset to simulate the measurement noise effect on the input. As the noise case is more challenging in estimating true damages, a more complex architecture of the proposed framework is constructed. The width factor K is set to be 2, which doubles the number of parameters of the network. The 'basic-dropout' residual blocks are used in the architecture to avoid overfitting. The depth factor N is set to be 2, which

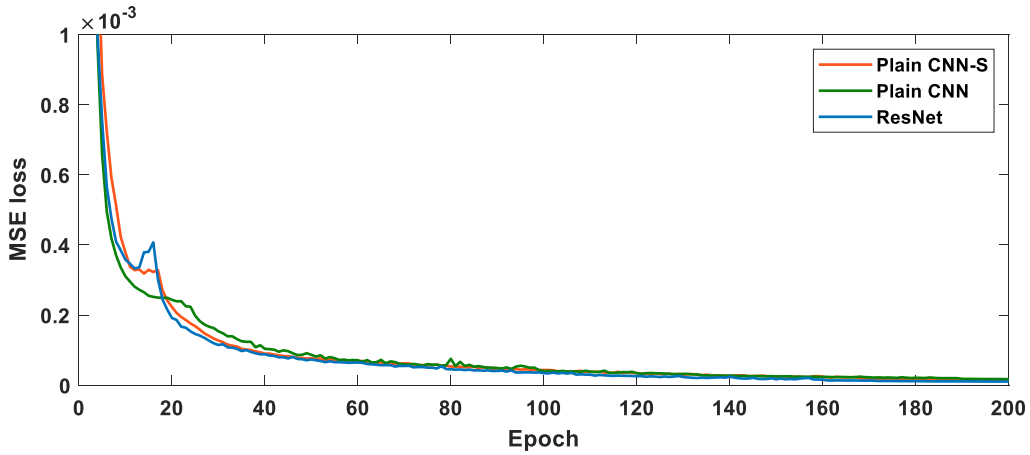

Figure 4-3: Testing curve of the CNN models and the proposed approach for *Scenario 1*.

includes in total 23 convolutional layers. The 'Plain CNN' is constructed with the same number of layers as the proposed framework. Similarly, a shallower model 'Plain CNN-S' is built with the depth factor $N = 1$, which includes 17 convolutional layers. As shown in Table 4.4, consistently, the proposed framework outperforms the other three methods with the smallest MSE, that is, 2.1×10^{-4}. It is noted that the R-values obtained from those four methods are quite close. When the depth or width of the proposed network are increased, the MSE does not decrease much. Thus the performance of the proposed approach for this scenario is considered as the best with the highest R-value. The testing curves reported in Figure 4-4 consistently show that the proposed approach outperforms the CNN models when the testing error stabilizes at around 150 epochs.

Table 4.4: Performance evaluation results for *Scenario 2* in the numerical study.

Methods	**MSE**	**R-value**
SAF	2.3×10^{-4}	0.886
Plain CNN-S	2.2×10^{-4}	0.886
Plain CNN	2.2×10^{-4}	0.886
The proposed framework	2.1×10^{-4}	0.888

To further evaluate the quality of damage identification in terms of both location and severity of the proposed framework, we randomly select two multiple-damage identification cases and present their results in Figures 4-5 and 4-6, respectively. The magnitude of stiffness reduction indicates the severity of the damages. As shown in Figure 4-5, both 'Plain CNN' and the proposed approach can detect the location of the two damages in

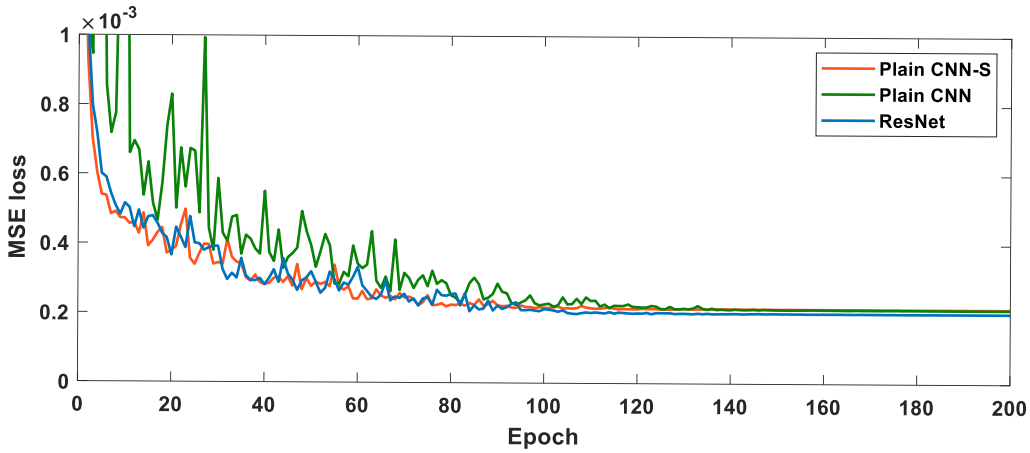

Figure 4-4: Testing curve of the CNN models and the proposed approach for *Scenario 2*.

the structure. However, the 'Plain CNN' fails to identify the true magnitude of the second damage, whereas the proposed approach produces a more accurate prediction in terms of the severity. Another multiple damage identification case is shown in Figure 4-6, in which both the 'Plain CNN' and the proposed approach can successfully identify the multiple damages in the structure. The identified stiffness reductions of the proposed framework are closer to the true labels, which indicates that the proposed approach is more robust. It is important to note that in [86], although the 'SAF' produces some good multiple damage identifications in *Scenario 2* as well, it results in many minor false identifications. On the contrary, false predictions are rare in the proposed approach, as shown in both Figures 4-5 and 4-6, which are also reflected in the MSE values.

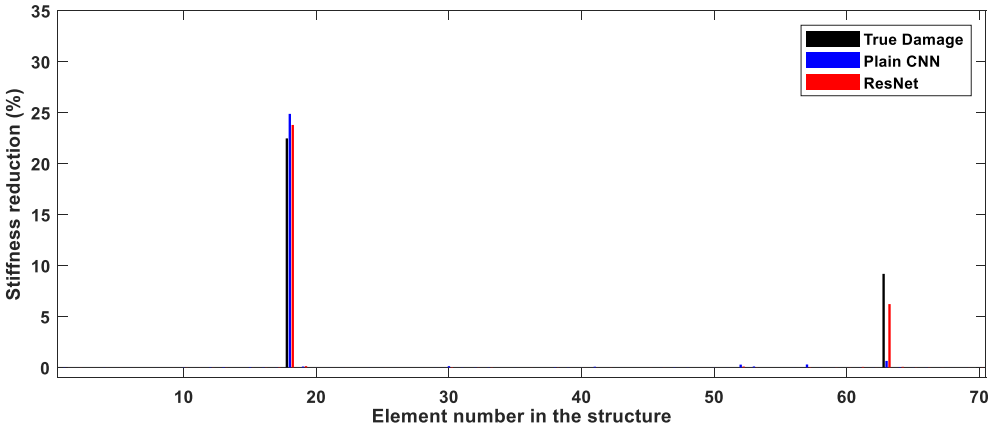

Figure 4-5: Damage identification results of a multiple damage case from the 'Plain CNN' and the proposed approach for *Scenario 2*.

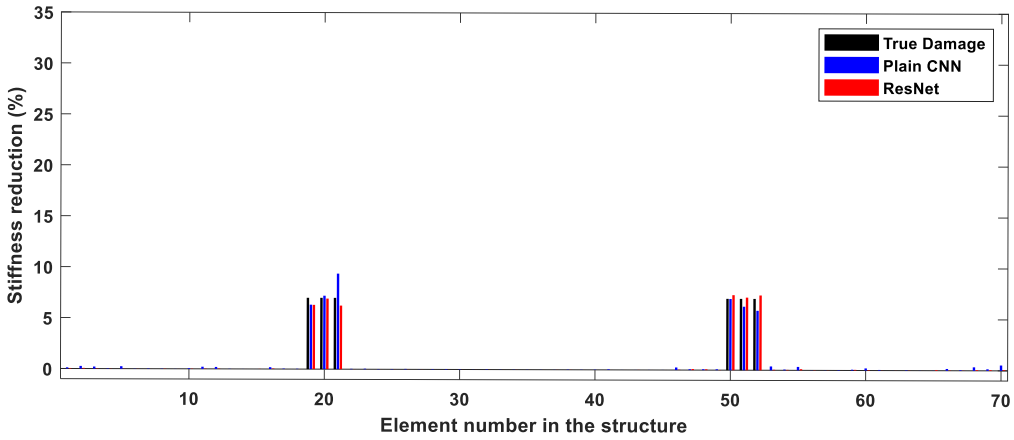

Figure 4-6: Damage identification results of another multiple damage case from the 'Plain CNN' and the proposed approach for *Scenario 2*.

Scenario 3: Modelling uncertainty is considered. Generally, an accurate finite element model is required for data generation in this study. The modelling inaccuracies will significantly affect the performance of damage identification. To investigate the robustness of the proposed approach, 1% uncertainty is included in the stiffness parameters to simulate the finite element modelling errors. The same architecture of the proposed framework used in *Scenario 2* is chosen for this scenario as well, that is, $K = 2$ and $N = 2$. Consistently, the 'Plain CNN' has the same number of layers as the proposed framework, while the 'Plain CNN-S' has a shallower architecture with half the number of layers of the 'Plain CNN'. The evaluation results of the state-of-the-art methods and the proposed approach are shown in Table 4.5. Even though 'SAF' achieves a good performance for this scenario, the proposed framework still further improves the results as reflected in both MSE and R-value. Figure 4-7 shows the testing curves of the CNN models and the proposed approach. Similarly, the training becomes stable at around 200 epochs and the proposed ResNet consistently shows the smallest testing error. Thus, the proposed approach consistently shows the best evaluation results among these four methods.

To further evaluate the robustness in modelling inaccuracies of the proposed framework, two damage identification results selected from the testing datasets are presented in Figures 4-8 and 4-9 to demonstrate the accuracies of damage identification in terms of damage locations and severity. The introduced uncertainties to the structural system are also shown in those figures as a set of random stiffness distributions. The predictions from both the 'Plain CNN' and the proposed approach of a multiple damage case are shown in Figure 4-8.

Table 4.5: Performance evaluation results for *Scenario 3* in the numerical study.

Methods	**MSE**	**R-value**
SAF	2.9×10^{-5}	0.986
Plain CNN-S	2.4×10^{-5}	0.988
Plain CNN	2.7×10^{-5}	0.986
The proposed framework	1.6×10^{-5}	0.992

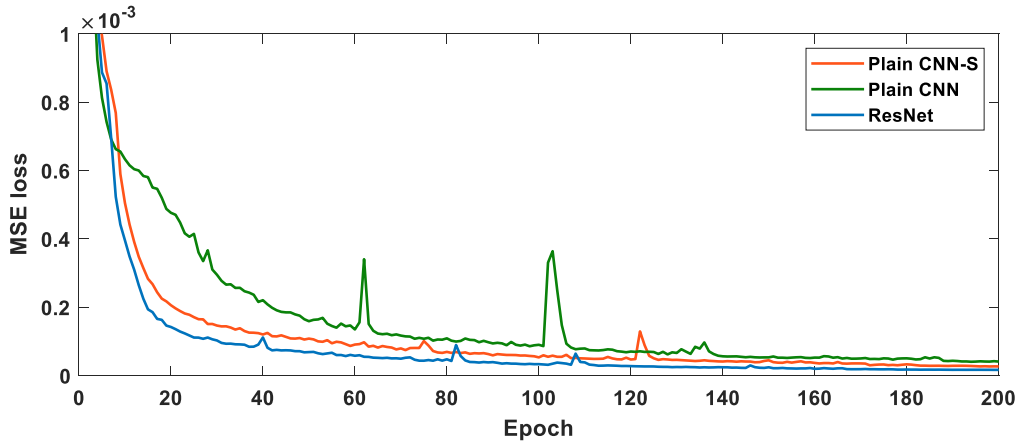

Figure 4-7: Testing curve of the CNN models and the proposed approach for *Scenario 3*.

It is observed that the stiffness reduction obtained from the proposed framework is closer to the true labels comparing to the 'Plain CNN', when the modelling uncertainties are presented in the input data. Besides, another multiple damage case is shown in Figure 4-9. There are two small damages in the structure. One of them has a 7% stiffness reduction at the 4^{th} element, while the other has a 2.2% stiffness reduction at the 55^{th} element. Although these two minor damages are more difficult to detect especially when uncertainties are introduced, the proposed framework can successfully detect the locations of the structural damages and predict relatively accurate damage magnitudes. However, the 'Plain CNN' fails to detect the second damage as shown in Figure 4-9. These experimental results demonstrate that the proposed approach is more accurate and more robust than 'Plain CNN'.

Scenario 4: Both the measurement noise and modelling uncertainty are considered. In this scenario, these two effects are combined in the datasets, making it the most challenging task in the numerical study. Here, 5% Gaussian noises is added to the mode shape and 1% uncertainty is included in the stiffness parameters.

The same architecture of the proposed framework used in both Scenarios 2 and 3 is

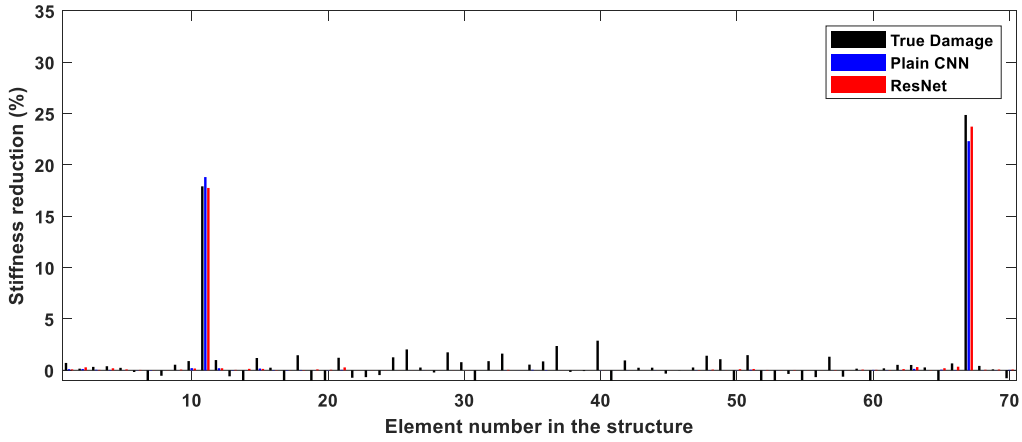

Figure 4-8: Damage identification results of a multiple damage case from 'Plain CNN' and the proposed approach for *Scenario 3*.

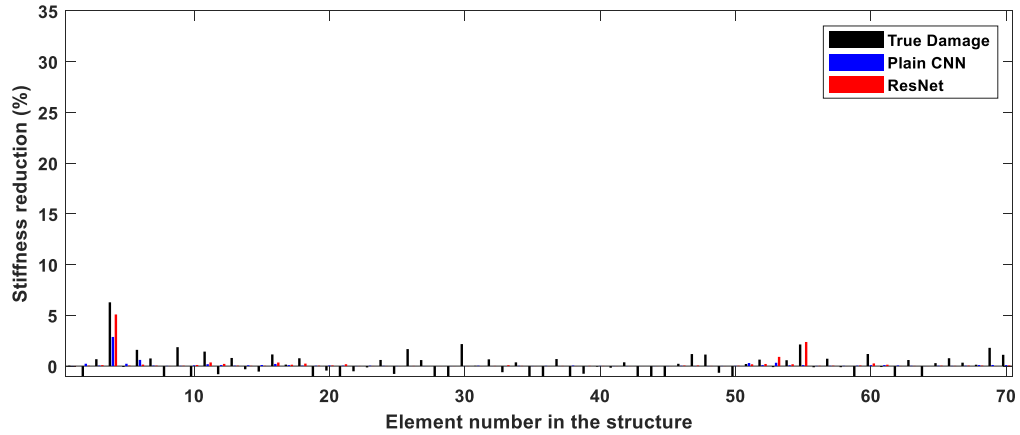

Figure 4-9: Damage identification results of another multiple damage case from 'Plain CNN' and the proposed approach for *Scenario 3*.

again chosen for this scenario. Both the width factor and depth factor are set to be 2, and 'basic-dropout' residual blocks are used in place of the 'basic' blocks as shown in Table 4.1 to regularize the training of the model. The CNN models also keep the same architecture as used in both Scenarios 2 and 3. Table 4.6 shows the testing results for *Scenario 4*, and a significant improvement is observed in the proposed framework in terms of the overall MSE and R-value on the testing dataset. The 'Plain CNN' improves the R-value from 0.792 to 0.903, while the proposed approach shows a further 3% improvement. The testing curves presented in Figure 4-10 show that the proposed approach is superior to other CNN methods. Since this scenario is considered as the most difficult task in this study, three damage identification results are shown in Figures 4-11, 4-12 and 4-13 respectively to demonstrate

the robustness and effectiveness of the proposed approach. A single damage case is shown in Figure 4-11. Both methods can detect the locations of the damage in the structure, while the estimated stiffness reduction from the proposed method is more accurate. Figure 4-12 shows a multiple damage case with a small level damage (*i.e.*, a 5% stiffness reduction) and a relatively big level damage (*i.e.*, a 22% stiffness reduction) in the structure. It is observed that both approaches identify the big level damage with accurate damage level values. However, the small level damage is only precisely detected by the proposed approach. A similar result is shown in Figure 4-13, where a median level damage and a small level damage are introduced. Both damages are well identified by the proposed approach with good stiffness reduction values. The 'Plain CNN' once again fails to correctly identify the small level damage.

Table 4.6: Performance evaluation results for *Scenario 4* in the numerical study.

Methods	**MSE**	**R-value**
SAF	3.2×10^{-4}	0.792
Plain CNN-S	1.9×10^{-4}	0.890
Plain CNN	1.7×10^{-4}	0.902
The proposed framework	1.1×10^{-4}	0.934

It is evidently demonstrated from the results of the four scenarios that the proposed framework produces more accurate and robust damage identifications, comparing to the state-of-the-art models, such as 'SAF' and 'Plain CNN'. The damage identification results presented in [86] show that the 'SAF' still produces some minor false predictions, while from the performance of the proposed approach, false predictions are rarely observed. Besides, the proposed framework is more robust to various level damages, as shown in the identification results.

4.5 Experimental validation

This section will describe experimental verification on a reinforced concrete bridge model (as described in Section 2.5.2) using the proposed damage identification approach.

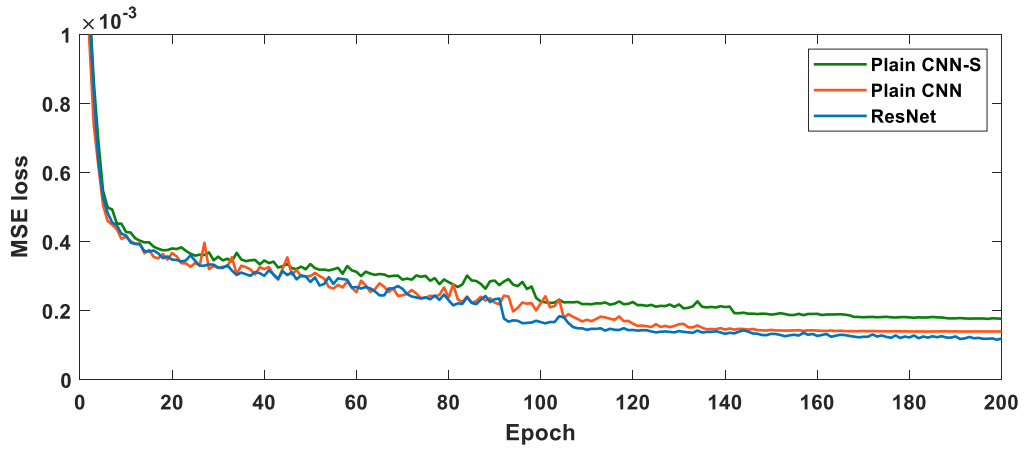

Figure 4-10: Testing curve of the CNN models and the proposed approach for *Scenario 4*.

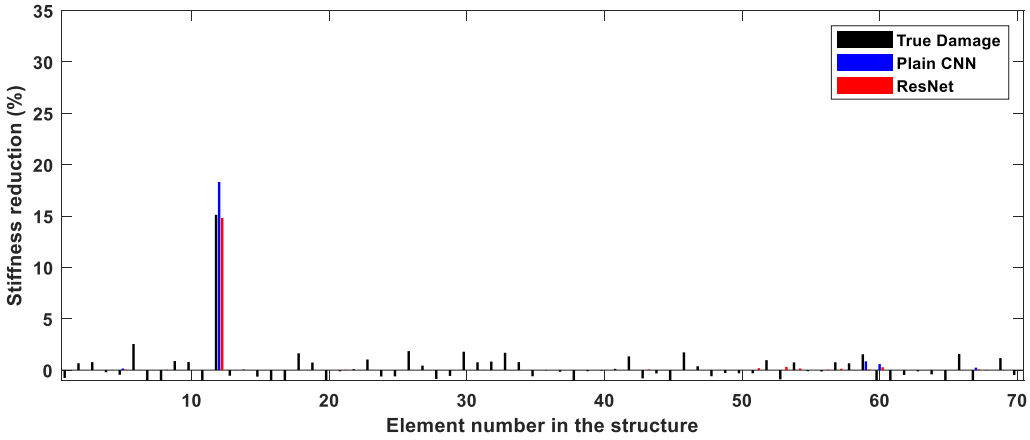

Figure 4-11: Damage identification results of a single damage case from 'Plain CNN' and the proposed approach for *Scenario 4*.

4.5.1 Data Generation

In the mid-span of the bridge structure, two-point static loads are applied to create the cracks in the structure. The static load is continuously increased to 180 kilonewtons and the final crack pattern is observed and shown in Figure 2-13. A large number of cracks are observed in the mid-span elements under damaged state. 24 major cracks are found in the structure from the 4^{th} to the 15^{th} elements. The dynamic test is carried out again to identify the first three natural frequencies and the corresponding mode shapes of the damaged structure to obtain the testing data set to investigate the accuracy of the proposed ResNet framework. For training the model, the training dataset is generated using the baseline finite element model with simulated different damage scenarios and including only the web elements from

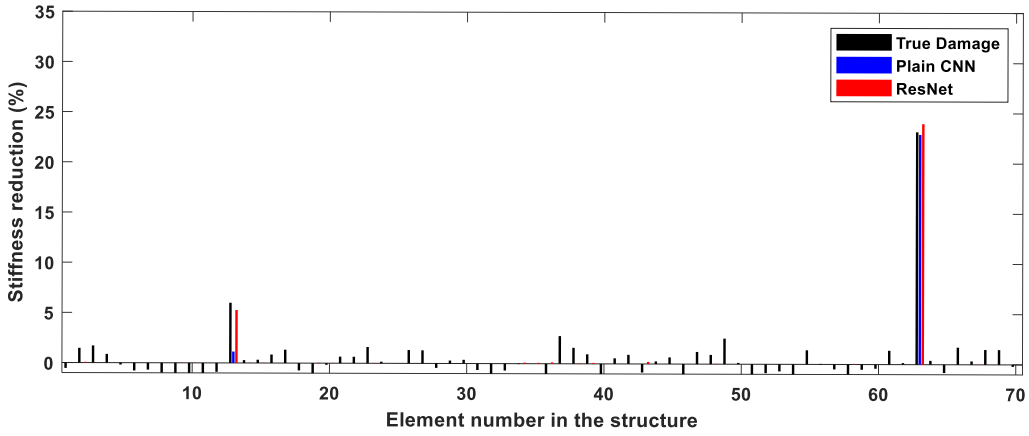

Figure 4-12: Damage identification results of a multiple damage case from 'Plain CNN' and the proposed approach for *Scenario 4*.

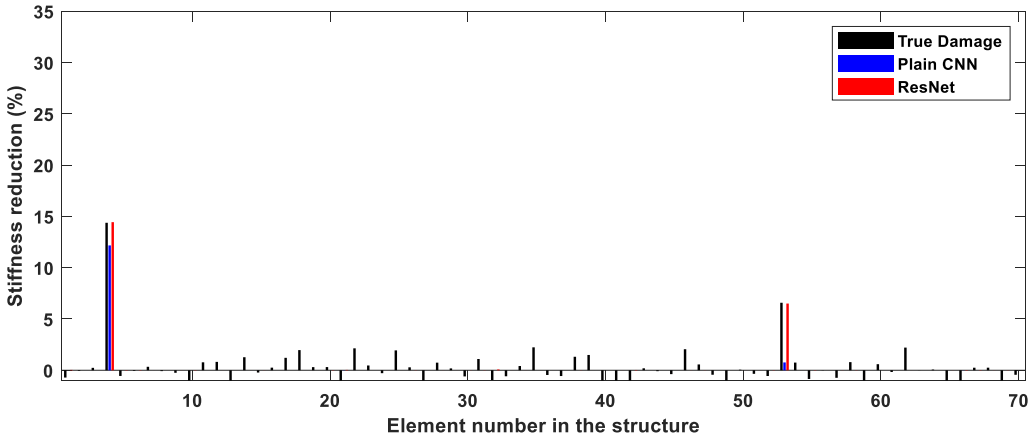

Figure 4-13: Damage identification results of another multiple damage case from 'Plain CNN' and the proposed approach for *Scenario 4*.

the 2nd to the 17th elements. This is because the damages are primarily observed in those elements during testing, as shown in Figure 2-13. In all 16 elements, uniformly distributed random damages are simulated with severity of the damage distributed between 0% and 15%. The damage level is taken in the step of 0.01%. Training samples with a size of 20000 are generated for this study. The modal information obtained is taken as the input to the model and simulated damage levels are taken as output. The training datasets are generated carefully, considering minor damages (less than 15%) in structures since they are usually the primary concerns in SHM. A large dataset with damages in this range is therefore generated to train the model, and a very well trained model is achieved as demonstrated in the damage identification analysis. If less data are generated and used to train the network, *e.g.*, a less

refined step with damage increment of 0.1% is used, the trained model is believed to still be able to identify the damage but the resolution of damage severity would be at best 0.1%, rather than 0.01%. It is difficult to draw a line on the size of the dataset needed to train a model. It varies from case to case and depends on the desired accuracy of the model in identifying the structural conditions. In general, the larger the dataset is, the more accurate the trained model would be.

4.5.2 The deep ResNet structure

The details of the network structure is presented in Table 4.2. The width factor ($K = 1$) and depth factor ($N = 1$) are selected according to the complexity of the damage identification problem of this structure model. In total 13 convolutional layers are used in this network. The 'Plain CNN' with the same number of layers is also conducted for a fair comparison. The input size of the training data is 3×7, as 3 mode shapes of the T-section beam structure are used and each mode shape consists of 7 values, while the output includes 16 stiffness reduction parameters. MSE and R-value are employed to evaluate the quality of damage identification using the state-of-the-art method and the proposed framework.

4.5.3 Training performance and damage identification results

As the measurement data obtained from the laboratory inevitably include noise effect, the noise effect is considered in the training data. Thus, 5% white noise is added in the mode shapes of the training data to simulate the measurement noise. The training data is spilt into training, validation and testing subsets with the percentages of 70%, 15% and 15%, respectively. These datasets are pre-processed with data whitening as discussed in [86], and used for training these deep neural network models. The performance of these models are evaluated on the test data subset and shown in Table 4.7. It is observed that the proposed framework outperforms the other two methods evidenced by both MSE and R-value.

Table 4.7: Performance evaluation results in the experimental study.

Methods	**MSE**	**R-value**
SAF	4.6×10^{-6}	0.998
Plain CNN	1.5×10^{-6}	0.999
The proposed framework	5.1×10^{-7}	0.999

To further investigate the efficacy of the models, a real testing dataset is obtained from the laboratory of the structure. The modal information is fed to the trained network model as an input. Figure 2-7 shows the damage pattern obtained with the proposed approach. The identified damage pattern is compared with the crack pattern observed on the structure. It is observed that the proposed approach can identify the true damage pattern on the structure. Also, the 'Plain CNN' performs similarly as shown in Figure 4-14. The damage patterns obtained are consistent with the results obtained from [23], 'SAF' [86], and the reported cracks in the test structure [66].

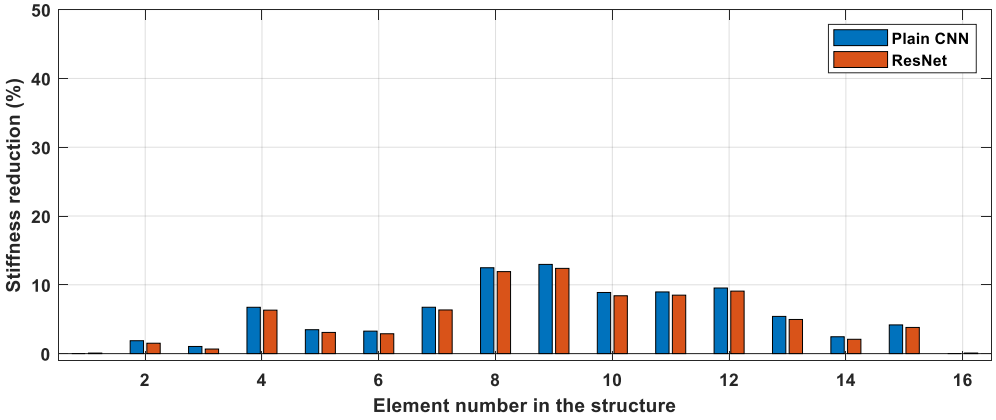

Figure 4-14: Damage identification results from 'Plain CNN' and the proposed framework.

4.6 Summary

This chapter presents a deep ResNet framework for structural damage identification and investigates the effectiveness and accuracy through numerical and experimental studies. The proposed method performs feature extraction and damage identification in the end-to-end learning body without pre-training, improving the efficiency in comparison with the autoencoder-based method described in Chapter 3. It also breaks the limitation of the network architecture on the depth due to the gradient vanishing issue. Evidently, this framework can be used to effectively detect the locations and identify the severity of the damages of structures as presented in the numerical and experimental studies. A seven-storey frame structure is investigated in the numerical study for structural damage identification. The proposed approach shows significant improvement over the state-of-the-art models, such as the 'SAF' and the well-known CNN models, when the datasets contain both uncertainty and

noise, which is expected in practical applications. Moreover, the results demonstrate the robustness of using the proposed approach for the complex structural damage identification problem as the frame structure has a large number of elements. Experimental studies on the T-beam bridge model demonstrate the feasibility and performance of the proposed approach, which achieves accurate crack pattern identification results. The T-beam structure is relatively simple thus the 'plain CNN' works generally well in the experimental studies. Future studies will investigate more challenging cases such as large-scale structures in the wild.

Chapter 5

Densely Connected Convolutional Network Framework for Structural Damage Identification

Chapter 4 presents a deep ResNet framework for structural damage identification of civil engineering structures. It breaks the optimization limitation of the two-component framework presented in Chapter 3. It also avoids performance degradation that happens in the deep CNN models due to vanishing gradients. In this chapter, we present a densely connected convolutional network framework that further improves the performance of damage identification, comparing to the deep residual network framework.

5.1 Introduction

Recent studies [3, 4, 43, 83] used modal information, such as natural frequencies and mode shapes, to identify damages of structures. However, to obtain modal information, such as mode shapes, a sufficient number of sensor measurements and a large number of channels in data acquisition systems accordingly are usually required for covering the whole structure, which may not be practical in the rough environment of large scale structures. However, only a certain number of sensors for measuring acceleration responses are required for the

[0]This chapter is reprinted, with permission, from [Wang, R., Li, J., An, S., Hao, H., Liu, W., & Li, L. Densely connected convolutional networks for vibration based structural damage identification. *Engineering Structures*, 245, 112871. © 2021 Elsevier. DOI: 10.1016/j.engstruct.2021.112871].

time domain methods for structural damage identification. It is noted that these sensor measurements do not necessarily have to cover the whole structure. Fewer sensors are required since time domain responses could provide more information for formulating the identification equation. Therefore, time-domain vibration responses, *e.g.*, acceleration responses are exploited to identify the structural damages. Recently, raw acceleration data have been directly used to identify structural states in [111]. It is straightforward to use several time-domain responses for identification. Besides, sufficient data points can be sampled using limited amount of time-domain sensors. Therefore, it has great potentials to employ time-domain responses for damage identification of structures.

The main challenge of using time-domain responses is that the sensor data always contain significant level of noise, it is not easy to interpret vibration response information, and vibration response is also loading condition dependent. Hence, it is essential to find an efficient and effective approach to learn the unknown relationship between sensor measurements and damage patterns. As discussed in Chapter 4, deep leaning methods have been extensively applied to structural health monitoring. Some advanced deep learning models, includes deep autoencoders [85, 86, 102], deep CNNs [7, 10], and deep ResNets [101], have achieved great results in vibration-based structural damage identification.

Inspired by ResNets, Huang [45] developed a new architecture known as densely connected convolutional networks (DenseNets) and achieved state-of-the-art results in the same competition (ILSVRC) in 2017. Instead of increasing the representational power from simply increasing the depth of architecture, DenseNets exploit the potential of the network via feature reuse. It sets up skip connections to connect all layers directly to each other, which is referred to as dense connectivity. These designs further strengthen the feature propagation and encourage feature reuses in the network, making the network easy to train. The dense connectivity pattern makes the network highly parameter-efficient, since there is no need to relearn redundant feature maps.

In this chapter, a novel approach for structural damage identification, namely, SDI-DenseNet is presented. It is a specially designed framework based on the DenseNet, which fits well for the SHM study using time-domain responses, *e.g.*, acceleration. The remainder of this chapter is arranged as follows. Section 5.2 introduces the theoretical background and development of DenseNets; Section 5.3 describes the architecture of the proposed SDI-DenseNet; Sections 5.4 and 5.5 evaluates the performance of the proposed SDI-DenseNet

framework, presenting both numerical studies and experimental verifications results; Finally, summary is drawn in Section 5.6.

5.2 DenseNets

For a standard convolutional neural network with L layers, all layers are usually connected sequentially in the architecture. When increasing the depth of architecture, the gradients may vanish or explode during back propagation. Thus, it is hard to train the network when L is substantially large. Moreover, only high-level features are used for the last objective layer in a standard convolutional neural network.

Although ResNets solves the gradient vanishing/exploding problems by attaching skip connections in the architecture, it requires a large number of parameters compared to DenseNets. DenseNets is developed with a novel connectivity pattern introduced in the architecture. The L layers are directly connected, hence has $L(L+1)/2$ connections in the network, instead of L connections. Concatenation operation is used instead of summation to collect features from the previous layers for each layer. Details of the key components of DenseNets are described as follows.

5.2.1 Dense Block

Figure 5-1 illustrates the connectivity patterns in a dense block, in which the input of each layer is a concatenation of all feature maps from all the preceding layers. Each layer receives the concatenation of feature maps from all preceding layers. Therefore, the layer transition in a dense block can be formulated as $x_l = f_l([x_0, x_1, \ldots, x_{(l-1)}])$, where $[x_0, x_1, \ldots, x_{(l-1)}]$ is the concatenation of the feature maps of all layers preceded to the lth layer. $f_l(*)$ is a composite function, which is applied to perform a sequence of the successive transformations: batch normalization (BN), followed by a rectified linear unit (ReLU) and a convolution (Conv). Generally, each function $f_l(*)$ produces k feature maps, hence k indicates the growth rate of the network. Intuitively, these k feature maps generated by each dense block can be viewed as its local state that contributes to the global state of the network by concatenation.

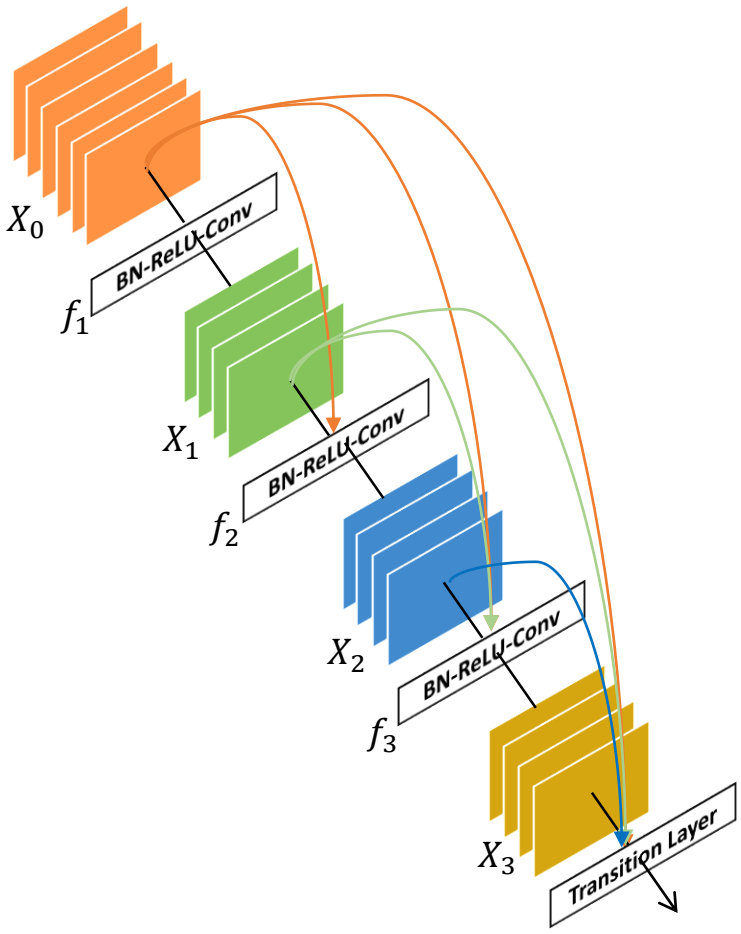

Figure 5-1: A schematic 3-layer dense block with the growth rate k=4.

5.2.2 Transition Layers

Only feature maps with the same size can be concatenated, thus no feature size down-sampling is performed in dense blocks. However, down-sampling is an important part of CNNs. To achieve this, transition layers with pooling are inserted in-between dense blocks to form a DenseNets. The basic architecture of the transition layer is BN-ReLU-Conv (1×1) with Average Pooling (2×2).

5.2.3 Model Compression

Bottleneck layers and compression in transition layers are introduced in DenseNets to improve computational efficiency and model compactness. The typical architecture of the

bottleneck layer is from BN-ReLU-Conv(1×1) to BN-ReLU-Conv(3×3), in which 1×1 Conv reduces the feature-maps. In the transition layer, feature-maps are further compressed by the compression factor $\theta(0 < \theta \leq 1)$. $\theta = 0.5$ is generally used, which means that the number of input feature maps is reduced to a half.

5.3 The Proposed SDI-DenseNet

This section describes the proposed SDI-DenseNet for structural damage identification. The overall architecture of the proposed model is shown in Figure 5-2. The measured acceleration responses, which can reflect changes in structural vibration characteristics, are related to the physical properties of structures such as structural element stiffness. In this study, the time-domain acceleration responses are exploited to identify the damages (*e.g.*, stiffness reductions) of structures. Since acceleration responses collected from multiple sensors are used as the input, the input shape is the number of data points by the number of sensors. The output of SDI-DenseNet is a vector of elemental stiffness reductions of the structure.

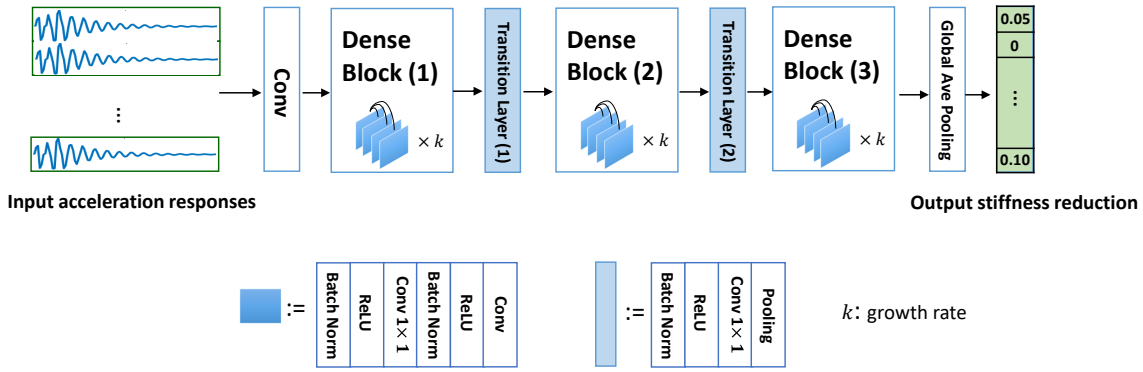

Figure 5-2: The proposed SDI-DenseNet.

5.3.1 Architecture and objective function

The initial convolutional layer in the proposed SDI-DenseNet is to extract the preliminary patterns from the input of each sensor, with a relatively large filter size. Three dense blocks and two transition layers are used in the network. Bottleneck layers are adopted in the dense block to improve the computational efficiency, while the compression factor $\theta = 0.5$ is introduced to further compress the model. A global average pooling (GAP) layer is placed after the last dense block, followed by the final objective layer. In this study, the

objective of the developed network is to map the damage-related patterns to the structural stiffness parameters, including damage locations and severities. Hence, the objective layer is viewed as a regression layer, implemented by a fully-connected layer with a 'tanh' activation function $\sigma(x) = \frac{e^{2x}-1}{e^{2x}+1}$, which produces the estimated stiffness reductions, *i.e.*, damage levels in structural elements. Furthermore, the objective function, *i.e.*, cost function, is set as the mean squared error (MSE) loss that minimizes the mean squared difference between the predicted stiffness reductions $\bar{y}$ and the true labels y. To avoid the overfitting of the network, an L2 regularization term is added to the cost function which penalizes the weight parameters W. The penalty parameter λ is set to 0.0005 in this study. The cost function can be denoted as $Cost(y, \bar{y}) = \frac{1}{m}\sum_{i=1}^{m} ||y_i - \bar{y}_i||_2^2 + \frac{\lambda}{2m}\sum_{l=1}^{L} ||W^{[L]}||_2^2$ where m is the number of training samples while L is the number of layers.

5.3.2 Advantages of the proposed SDI-DenseNet

The proposed SDI-DenseNet employs the dense connectivity pattern in the network architecture which makes full use of the advantages of DenseNets. Firstly, the feature maps generated by different layers are concatenated continuously, which strengthens the information flow in forward computation and gradients propagation in backward computation. Meanwhile, the features from all previous layers are reused, which reduces the redundancy and increases the computational efficiency during the training process. In addition, diverse levels of features are preserved through the dense connectivity and fed to the objective layer, which tends to generalize better convergence and achieve better performance. As time-domain acceleration responses are used as input in this study, it is beneficial to preserve diverse levels of features. Intuitively, low-level features are more related to the changes of acceleration in a short period, while high-level features cover the whole measurement of the input. Thus, diversified variations of the acceleration responses are captured and used for the final regression, making the performance more generalized. Moreover, bottleneck layers and model compression are introduced in the proposed approach which further improves the computational efficiency. Thus, it is capable of training a very deep model with the proposed approach for even more complicated structural damage identification problems. In summary, a generic SDI-DenseNet is developed to identify the damage locations and severities of structures. To evaluate the performance of the proposed approach, both numerical studies and experimental verification are conducted and presented in the following sections.

5.4 Numerical Studies

Numerical studies are conducted in this section to validate the accuracy of using the proposed approach for structural damage identification with time-domain acceleration responses. A beam structure is used for this numerical study which is described in Section 2.5.1.

5.4.1 Data Generation

Figure 2-7 shows the finite element model of the beam structure. Acceleration responses from selected nodes No. 2, 5, 8, and 9 are measured when an impact force is applied. The sampling rate is taken as 100 Hz, and time-domain acceleration response data are generated for undamaged and damaged cases. For the damage case, up to three element damages are considered. The stiffness parameters of the elements are reduced to introduce stiffness reductions in the elements. The maximum stiffness reduction of 30% is used for some specific elements. The stiffness reductions are simulated from 0% to 30% with an interval of 0.5%. The datasets have ten labelled target variables, and each represents the stiffness reduction in a corresponding element. The elements and stiffness reduction parameters are randomly selected for multiple damage cases. To consider the loading effect, multiple samples are generated for every damage case using hammer impact forces with different amplitudes. The simulation studies consider the following four cases:

- Case 1: No measurement noise is considered in acceleration responses and no uncertainties are considered in structural finite element modelling.
- Case 2: No uncertainties in structural finite element modelling but measurement noise is considered. Different levels of white noise with signal-to-noise ratios (SNR) of 30 dB and 20 dB are added to the acceleration responses, respectively.
- Case 3: Uncertainties are considered in structural finite element modelling but no measurement noise is included in acceleration responses. Uncertainties with a Covariance of Variation of 1% - 3% are included randomly in the stiffness parameters of structural elements.
- Case 4: Both the uncertainties in the structural finite element modelling and measurement noise are considered. White noise is added to the acceleration responses considering uncertainty in the finite element modelling in structural analysis.

27,182 samples for the first two cases, and 59,700 samples for Cases 3 and 4 are generated respectively. For monitoring the structure, the force should be applied at the same location in data generation. It should be noted that each sample is generated with a randomly selected force from a force vector matrix with a varying magnitude following a normal distribution, and the applied impact force is used only for data generation and not for testing. Only the generated acceleration responses are used for training, validating, and testing the proposed approach.

5.4.2 Data Pre-processing and Model Hyper-parameters

The acceleration responses collected from four nodes are formulated as the input of dimension 100×4. Each stiffness parameter in the output vector is pre-processed to the range of [-1, 1] to serve the operating range of 'tanh' activation function. A generic architecture is proposed for all the experiments, but the hyper-parameters, such as the number of layers, filter sizes, and strides are selected accordingly based on the datasets. In numerical studies, the input data of the simply supported beam structure is collected from 4 sensors each providing 100 data points, and thus the input size is 100×4. Since this beam structure is relatively simple with only 10 stiffness elements, a model with $L = 40$ layers and growth rate $k = 12$ is selected. Table 5.1 summarises the architecture of the proposed network for numerical studies.

Table 5.1: The architecture of the proposed network for numerical studies.

Layers	**Settings**	**Output size**
Convolution	10×1, strides = (5,1)	$20 \times 4 \times 24$
Dense Block(1)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 2 \text{ conv} \end{bmatrix} \times 6$	$20 \times 4 \times 96$
Transition Layer(1)	1×1 conv	$20 \times 4 \times 48$
	2×2 ave pooling, strides = (2,2)	$10 \times 2 \times 48$
Dense Block(2)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 2 \text{ conv} \end{bmatrix} \times 6$	$10 \times 2 \times 120$
Transition Layer(2)	1×1 conv	$10 \times 2 \times 60$
	2×2 ave pooling, strides = (2,2)	$5 \times 1 \times 60$
Dense Block(3)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 1 \text{ conv} \end{bmatrix} \times 6$	$5 \times 1 \times 132$
Regression Layer	5×1 global ave pooling	$1 \times 1 \times 132$
	10×1 fully-connected, tanh	10

5.4.3 Performance Evaluation

The performance of the proposed approach is evaluated for the four cases described in Section 5.4.1. The pre-processed datasets are randomly split into three subsets, namely, 70% for training, 15% for validation, and another 15% for testing. All hyper-parameters are selected according to the validation loss and optimized by using random search. 100 epochs are executed with the Adam optimizer. The learning rate is set as 0.0008 and decayed by a factor of 0.5 once the validation loss reaches a plateau. As presented above, MSE loss is chosen as the metric for measuring the distance between labels (true damages) and predicted outputs. Another metric, that is, regression value $R(0 < R \leq 1)$ is also used to assess the quality of the trained models. In more detail, R is the correlation coefficient, which quantifies the linear correlation between true labels and estimated output values. To summarize, the smaller the MSE loss or the higher the R value, the more accurate the trained models. The details of performance evaluation for all cases are presented as follows.

MSE and R-value

The proposed SDI-DenseNet is first assessed on Case 1 and Case 2, and the results are shown in Table 5.2. In Case 1, clean datasets are used without considering any noise effect. Both MSE (4.76×10^{-6}) and R-value (0.999) indicate an outstanding performance of the proposed approach. In Case 2, R-values obtained from the noise datasets are 0.998 when SNR=30dB and 0.981 when SNR=20. These results indicate that the proposed approach is robust to measurement noise.

Table 5.2: Performance evaluation results for Case 1: Clean Dataset and Case 2: Measurement Noise.

Metrics	**Baseline**	**Measurement Noise**	
		SNR=30dB	SNR=20dB
MSE	4.76×10^{-6}	3.38×10^{-5}	2.49×10^{-4}
R-value	0.999	0.998	0.981

The test results for Cases 3 and 4 are presented in Table 5.3. 1% to 3% modelling uncertainty is considered in Case 3, while Case 4 includes both modelling uncertainty and measurement noise effects. The MSE and R-value for Case 3 are 1.59×10^{-5} and 0.999, respectively, which indicates that the proposed approach is also robust to modelling uncer-

tainty effect. The performance degrades when measurement noise is included. However, the lowest R-value is 0.96 even when a high-level noise (SNR=20dB) is added, which is the most challenging case in all these experiments. It is observed in Tables 5.2 and 5.3 that in general, the proposed approach can provide accurate structural damage identifications.

Table 5.3: Performance evaluation results for Case 3: Uncertainty and Case 4: Uncertainty and Measurement Noise.

Metrics	**Uncertainty**	**Uncertainty+Measurement Noise**	
		SNR=30dB	SNR=20dB
MSE	1.59×10^{-5}	8.13×10^{-5}	3.97×10^{-4}
R-value	0.999	0.992	0.960

Examples of damage identification results

To further demonstrate the performance of using the proposed approach for damage identification, several typical damage examples are randomly selected from the datasets and the damage identification results are visualized. Cases 1 and 2 are first illustrated and discussed. Two single-element damage cases and one multiple-element damage case are randomly selected from the testing datasets to demonstrate the accuracy of the trained model.

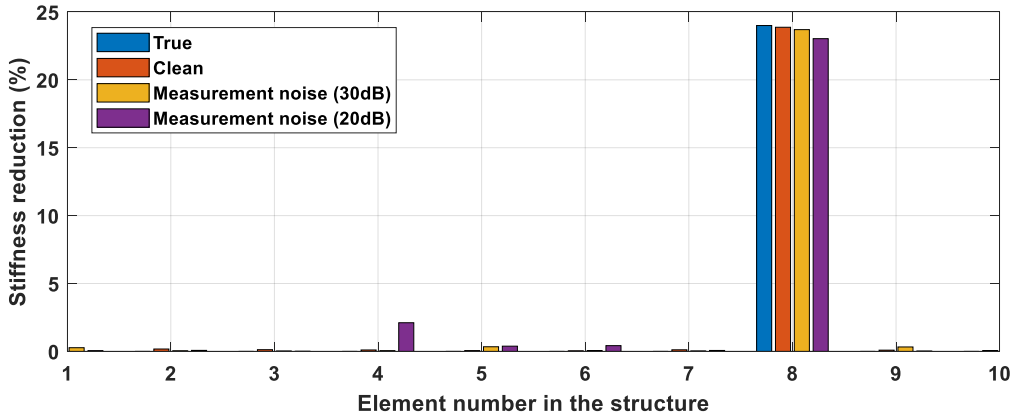

Figure 5-3: Single-element damage: major stiffness reduction sample with and without noise measurement.

Figure 5-3 shows a single-damage case with a major stiffness reduction (24%) at element No.8. It is observed that the damage can be located correctly for all cases. The identified stiffness reductions for the clean dataset, a noisy dataset (SNR= 30dB) and another noisy

dataset (SNR= 20dB) are 23.9%, 23.7%, and 23%, respectively. The results indicate that the severities of damage can be estimated accurately. A single-damage case with a minor stiffness reduction (3.5%) at element No.5 is shown in Figure 5-4. Although detecting minor damage is a more difficult task especially when measurement noise effect is included, the proposed approach can still accurately identify the damage location for all the cases. Consistently, the estimated stiffness reductions (*i.e.*, 3.6%, 2.9%, and 2.7%) are also very close to the true damage level.

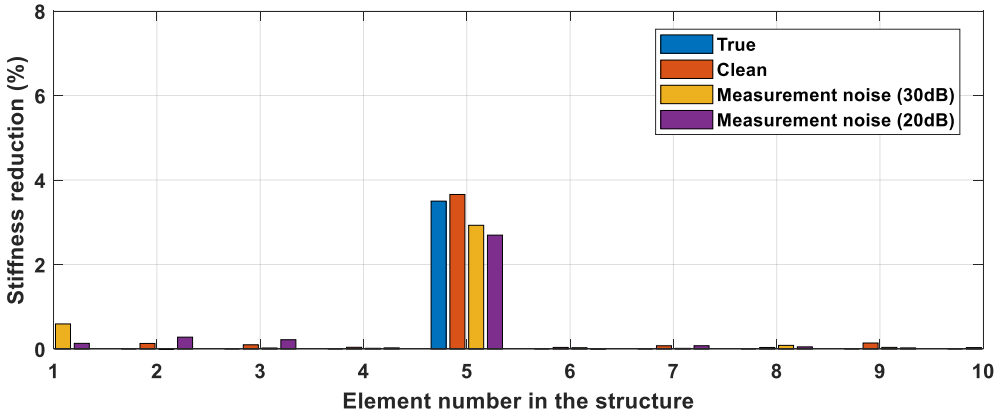

Figure 5-4: Single-element damage: minor stiffness reduction sample with and without noise measurement.

Figure 5-5 demonstrates the damage identification result of a multiple-element damage case. There are two minor stiffness reductions and one major stiffness reduction in the beam structure for the scenario. Clearly, the identified stiffness reduction values agree well with the true damage intensities, with 2 different levels of measurement noises included.

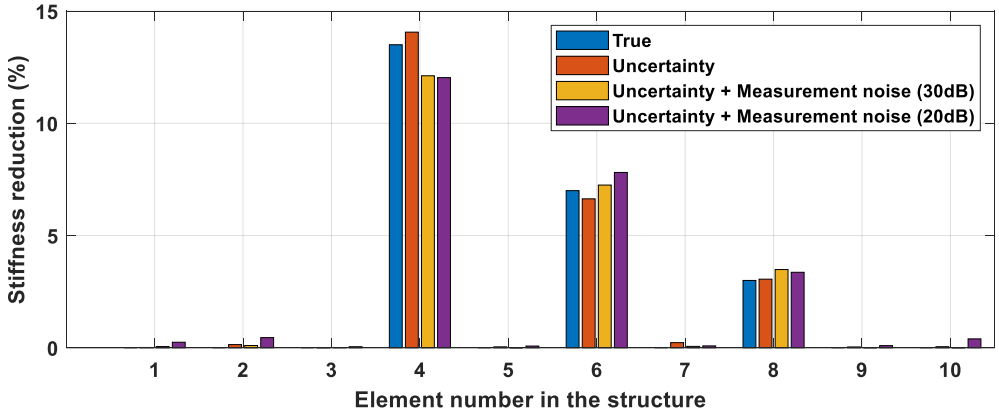

Figure 5-5: A multiple-element damage case with and without noise measurement.

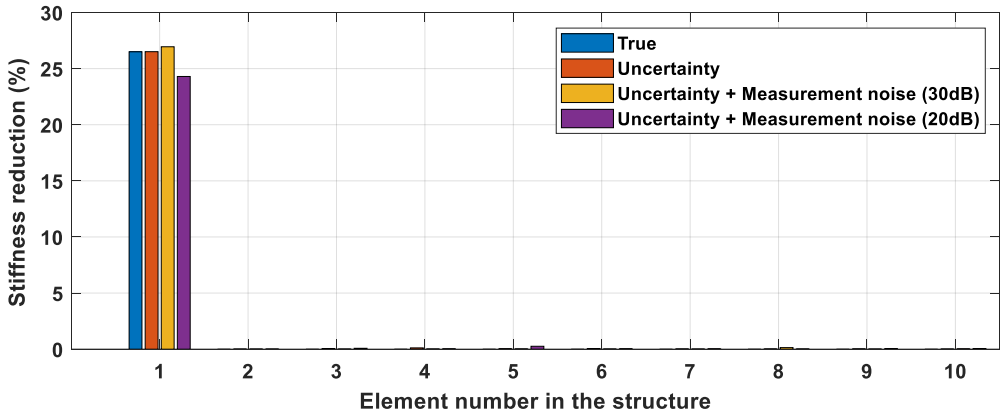

Figure 5-6: Single-element damage: major stiffness reduction case with noise measurement and modelling uncertainty.

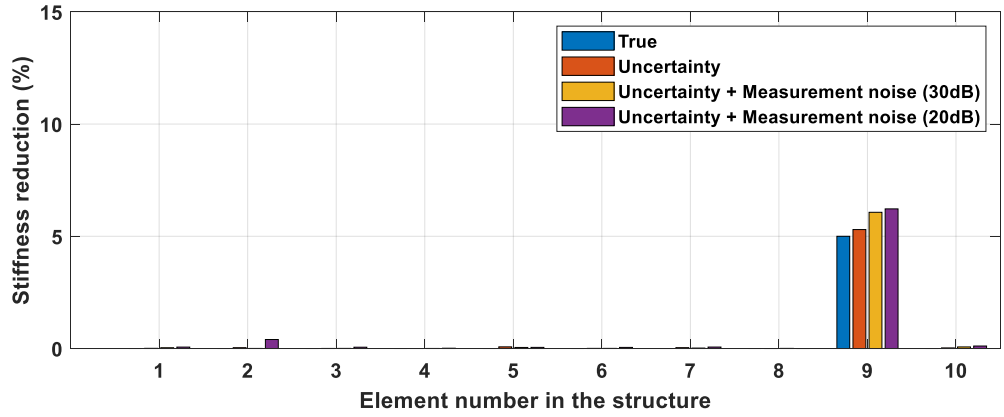

Figure 5-7: Single-element damage: minor stiffness reduction case with noise measurement and modelling uncertainty.

Two single-element damage cases are randomly selected from the testing datasets for Cases 3 and 4. Figure 5-6 shows the damage identification results of a major stiffness reduction (26.5%) at the 1st element. The identified damage value is 26.5% for the case with uncertainty, which is exactly the same as the true damage level. For Case 4, the identification accuracy degrades slightly with the increment of noise level for the cases with SNR of 30dB and 20dB. The identified stiffness reductions are 26.9% and 24.3% respectively. The identification results agree well with the true damage severities. Another single-element damage case with a minor stiffness reduction (5%) at element No.9 is shown in Figure 5-7. Similar to the major damage case, the identified result of the uncertainty case (5.2%) matches well with the true damage. The performance degrades marginally in the most challenging Case 4. The identified stiffness reduction is 6.2% when both the modelling uncertainty and

20 dB noise are included.

Figure 5-8 presents the multiple-element damage identification results for Case 3 and Case 4, respectively. It is evident that all damage locations are correctly identified by the proposed approach. Similar to the single-element damage cases, the identified stiffness reductions are very close to the true damage levels when only uncertainty is included. With the measurement noises included, the performance degrades slightly in the identified damage results. Generally, the proposed approach can correctly identify damage locations and approximately identify the damage severities for both Case 3 and Case 4. Overall, all the simulation cases conducted in the numerical studies demonstrate that the proposed SDI-DenseNet works effectively and accurately for structural damage identification using acceleration responses.

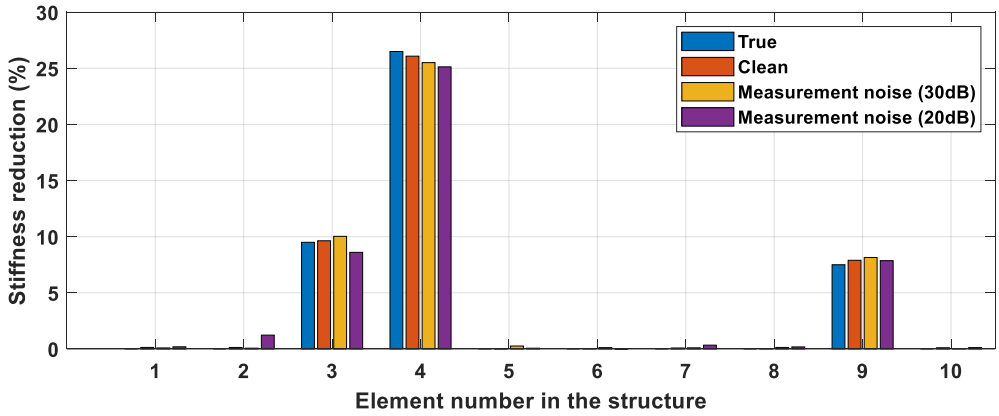

Figure 5-8: A multiple-element damage case with both measurement noise and modelling uncertainty.

5.5 Experimental validation

In this section, experimental studies are conducted to further validate the effectiveness of the proposed approach. The performance comparison against CNN and ResNet models is also conducted to verify the improvement of the proposed approach. A seven-storey steel frame structure, described in Section 2.5.2, is used. In this study, experimental data are used to investigate the accuracy and performance of the proposed SDI-DenseNet for structural damage identification.

As discussed in Section 2.5.2, numerical data generated from the updated finite element

model is used for training, validation, and testing of the developed network. The experimental data collected from laboratory testing of damaged structures are used for testing the accuracy of the proposed approach.

Two damage cases, *i.e.*, a 12.5% stiffness reduction at the 6^{th} element and a two-element damage case with a 12.5% stiffness reduction at both the 6^{th} and 12^{th} elements, are introduced in the frame structure in laboratory testings as shown in Figure 2-15. The data generation process of the numerical data is presented in the next section.

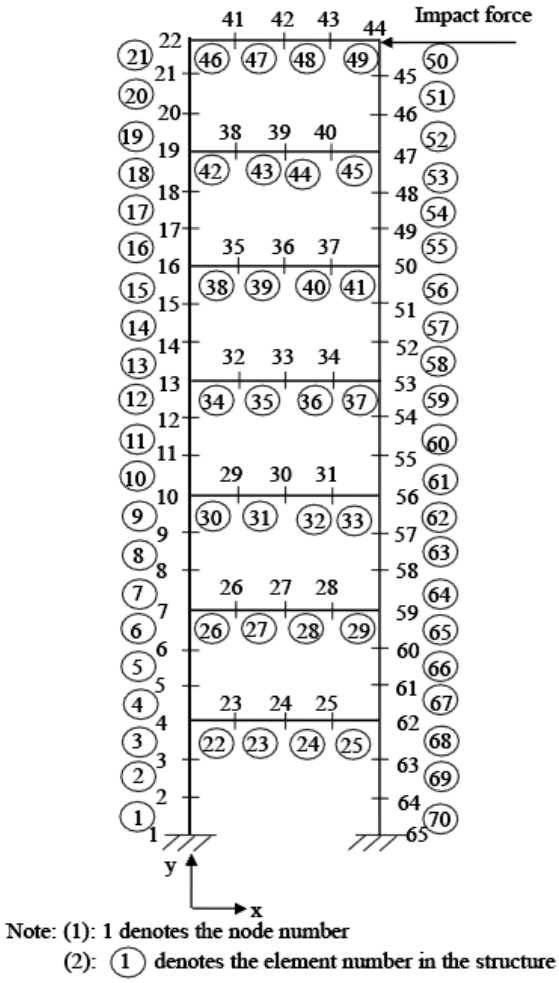

Figure 5-9: The finite element model of the experimental testing frame structure.

5.5.1 Data Generation

The training datasets are generated based on the updated finite element model (described in Section 2.5.1) of this frame structure with undamaged, single-element, and two-element

damage cases by applying an impact force at node 44 in the x-direction as shown in Figure 5-9. Random impact force with 1-2% variance of the measured force in the tests is applied to simulate the variations in the applied impact force. Acceleration responses are measured from nine nodes at the sampling rate of 1000 Hz. For the single-element damage case, $4x$, $7x$, $9x$, $11x$, $17x$, $47x$, $50x$, $53x$, and $56x$ are selected as sensor measurement locations, where x denotes the acceleration response measured in the x-direction. $4x$, $5x$, $11x$, $14x$, $19x$, $50x$, $53x$, $56x$, and $59x$ are selected for the two-element damage case.

Since different sensor placements are employed in the experimental dynamic tests for single and multiple damage cases, two networks are trained accordingly in this study. If a large number of training examples including single and multiple damage scenarios with different sensor placement configurations are available, training a network for identifying both single and multiple damage scenarios is feasible. For the single-element damage case, training data samples are generated by taking stiffness reduction in each element from 0% to 30% with an increment of 0.05%. Each sample is a concatenation of acceleration responses from the selected sensor locations. 16,520 samples of damage cases and four samples of undamaged cases are generated. For the two-element damage case, the stiffness is reduced in randomly selected two elements with a maximum stiffness reduction of 30%. In total, 28,008 samples are generated.

The data pre-processing technique remains the same as the one used in the numerical study described in Section 5.4.2. Acceleration responses measured for a damage case from the selected sensor locations are formulated as a two-dimensional matrix of dimension (m, n), where m is the number of data points and n is the number of sensors. Seventy elemental stiffness parameters of the steel frame structure are defined as the output. For the single-element damage case, 0.5 seconds of acceleration responses are acquired from each of nine sensors at a 1000 Hz sampling rate. Therefore the input dimension is 500×9. For the two-element damage case, the input dimension is 450×9, as 0.45 seconds of acceleration responses are used. Although both cases use the same number of sensors, the locations of sensors are different. Hence, two experiments with different networks are conducted separately using different datasets. Similarly, for all experiments, the data generated from the updated finite element model are split into three subsets for training (70%), validation (15%), and testing (15%), respectively. Additionally, experimental acceleration response data collected from laboratory testing are used as testing datasets for experimental verification to investigate if

the proposed approach can identify the introduced damage in these two damage scenarios of the laboratory frame structure.

5.5.2 Model Hyper-parameters

The hyper-parameters for experimental studies are selected accordingly based on the datasets. A model with the layer of $L = 40$ and model compression factor of $\theta = 0$ (*i.e.*, feature maps are not compressed) is used for the single-damage element case. While a deeper model with the layer of $L = 100$ and compression factor of $\theta = 0.5$ is used for the more challenging case, *i.e.*, two-damage element case. For both cases, the growth rate is set as $k = 12$. Details of these two models are presented in Tables 5.4 and 5.5, respectively.

Table 5.4: The architecture of the proposed network for experimental verification on single-element damage case.

Layers	**Settings**	**Output size**
Convolution	20×1, strides = (5,1)	$100 \times 9 \times 24$
Dense Block(1)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 2 \text{ conv} \end{bmatrix} \times 6$	$100 \times 9 \times 96$
Transition Layer(1)	1×1 conv	$100 \times 9 \times 96$
	2×2 ave pooling, strides = (2,2)	$50 \times 4 \times 96$
Dense Block(2)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 2 \text{ conv} \end{bmatrix} \times 6$	$50 \times 4 \times 168$
Transition Layer(2)	1×1 conv	$50 \times 4 \times 168$
	2×2 ave pooling, strides = (2,2)	$25 \times 2 \times 168$
Dense Block(3)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 2 \text{ conv} \end{bmatrix} \times 6$	$25 \times 2 \times 240$
Regression Layer	25×2 global ave pooling	$1 \times 1 \times 240$
	70×1 fully-connected, tanh	70

5.5.3 Performance Evaluation

Identification results using experimental data

The performance of the proposed SDI-DenseNet is evaluated by examining the damage identification results of both the single-element and two-element damage cases. Two networks are trained and tested individually for each case, and the network architectures are presented in Tables 5.4 and 5.5, respectively. As aforementioned, 40 layers and 100 layers are

Table 5.5: The architecture of the proposed network for experimental verification on two-element damage case.

Layers	**Settings**	**Output size**
Convolution	20×1, strides = (5,1)	$90 \times 9 \times 24$
Dense Block(1)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 2 \text{ conv} \end{bmatrix} \times 16$	$90 \times 9 \times 216$
Transition Layer(1)	1×1 conv	$90 \times 9 \times 108$
	2×2 ave pooling, strides = (2,2)	$45 \times 4 \times 108$
Dense Block(2)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 2 \text{ conv} \end{bmatrix} \times 16$	$45 \times 4 \times 300$
Transition Layer(2)	1×1 conv	$45 \times 4 \times 150$
	2×2 ave pooling, strides = (2,2)	$22 \times 2 \times 150$
Dense Block(3)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 5 \times 2 \text{ conv} \end{bmatrix} \times 16$	$22 \times 2 \times 342$
Regression Layer	22×2 global ave pooling	$1 \times 1 \times 342$
	70×1 fully-connected, tanh	70

used in the networks respectively for the single and multiple damage cases according to the complexity of the damage identification problem of this structural model. CNN and ResNet are also applied for comparison in both damage cases to demonstrate the advantages of the proposed SDI-DenseNet. The number of layers, *i.e.*, the depth of each model and their corresponding number of parameters are shown in Table 5.6. It is demonstrated that with the same structure, the number of training parameters is significantly reduced by using the proposed approach. This makes the training more efficient, and is able to provide more accurate results. For each case (single- and multiple-damage), the same number of layers are used in all three networks to perform a fair comparison.

Table 5.6: Network parameters of each model with different depths.

Methods	**Depth**	**Parameters**	**Depth**	**Parameters**
CNN	40	0.95 Million	100	2.4 Million
ResNet	40	0.95 Million	100	2.4 Million
SDI-DenseNet	40	0.28 Million	100	0.84 Million

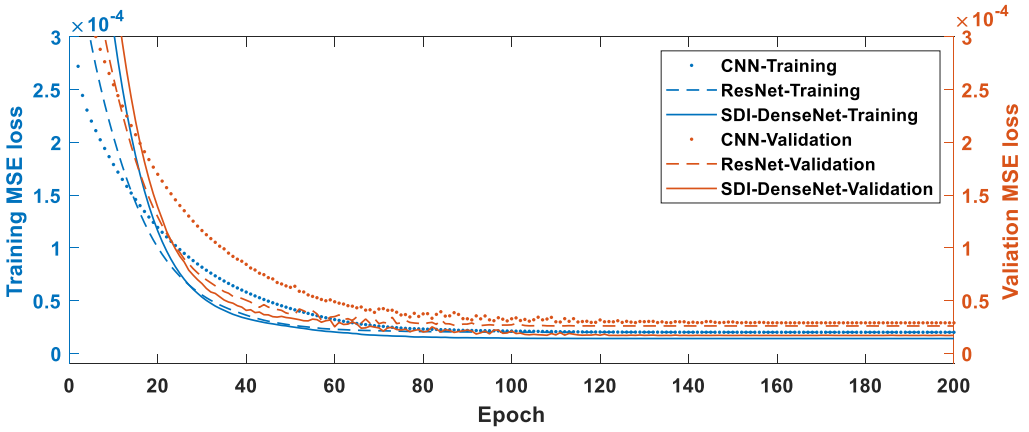

Figure 5-10: Training and validation curves of MSE loss for the single-element damage case.

Single-element Damage Case

Figure 5-10 shows the training and validation curves of using CNN, ResNet, and SDI-DenseNet for the single-element damage case. All networks are trained for 200 epochs starting with an initial learning rate of 0.0001. According to the plots, the training becomes stabilized and the validation loss of each network converges after 150 epochs. It is observed that the training and validation losses of the proposed SDI-DenseNet consistently outperforms the other two methods. Besides, SDI-DenseNet shows the smallest gap between the training and validation curves after stabilized, which indicates a better generalization on the validation sets than the CNN and ResNet. The MSE and R-values of testing on the updated finite element model are presented in Table 5.7. SDI-DenseNet shows the best performance with the lowest MSE value of 1.9×10^{-5} and the highest R-value of 0.980, further indicating that it can generalize better on the unseen datasets than the CNN and ResNet. The testing results on the experimental test case are shown in Figure 5-11. It is clearly shown that SDI-DenseNet can perfectly detect the location and quantify the severity of the damage, *i.e.*, a 12.5% stiffness reduction at the 12^{th} element of the frame structure. The CNN and ResNet can also be used to detect the location of damage at the 12^{th} element, but with less accurate damage quantification results of 7.5% and 9.6%, respectively. SDI-DenseNet also gives much more accurate predictions at other locations while the other two methods produce several false predictions, as observed in Figure 5-11.

Table 5.7: Performance evaluation results on the numerical testing datasets for the single-element damage case.

Metrics	CNN	ResNet	SDI-DenseNet
MSE	3.3×10^{-5}	2.9×10^{-5}	1.9×10^{-5}
R-value	0.963	0.967	0.980

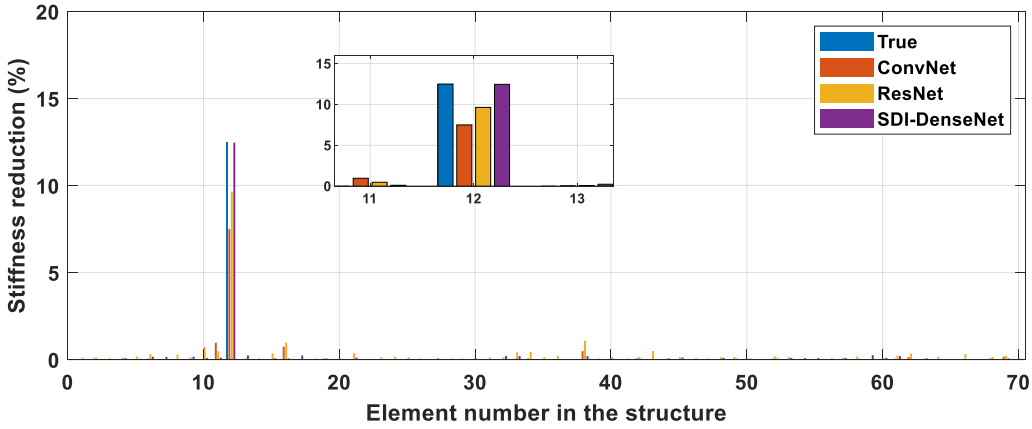

Figure 5-11: Damage identification result for the single-element damage case of the frame structure.

Two-element Damage Case

As aforementioned, 100 layers are used in training the three models, which is too deep for the CNN to converge due to the gradient vanishing issue. Thus, no result is obtained and presented from the CNN. Figure 5-12 shows the training and validation curves from the ResNet and the proposed SDI-DenseNet for the two-element damage case. The training process executes 200 epochs, and the initial learning rate is again set as 0.0001. Similarly, the curves indicate that the training of both models converges at around 160 epochs, while SDI-DenseNet consistently outperforms the ResNet with much lower training and validation loss. For the numerical testing data, the performance evaluation is shown in Table 5.8. A lower MSE value (8.7×10^{-5}) and a higher R-value (0.949) is obtained from SDI-DenseNet. The results further demonstrate the effectiveness of SDI-DenseNet. Next, a real experimental data set obtained in the laboratory is used to detect the damage. The same level (12.5%) of stiffness reductions is introduced into the 6^{th} and 12^{th} elements of the structure. Figure 5-13 shows the damage identification results of the multiple damage case. It is observed that both damages are well located. For SDI-DenseNet, the stiffness reduction of the first

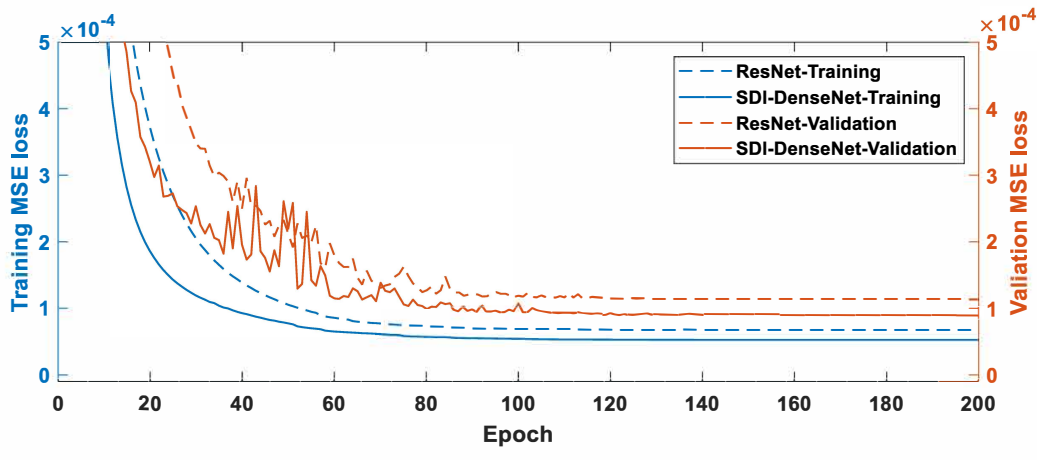

Figure 5-12: Training and validation curves of MSE loss for the two-element damage case.

damage in the 6$^{\text{th}}$ element is identified as 9.2%, whereas the second one is identified as 13.3%. The stiffness reductions obtained from the ResNet at both locations are 7.9% and 14.9%, respectively. It shows that SDI-DenseNet produces a more accurate identification result on the overall damage quantification. Besides, more false positive predictions (more than 1%) are obtained using the ResNet at the undamaged elements.

Table 5.8: Performance evaluation results on the numerical testing datasets for the two-element damage case.

Metrics	**CNN**	**ResNet**	**SDI-DenseNet**
MSE	-	1.2×10^{-4}	8.7×10^{-5}
R-value	-	0.932	0.949

5.6 Summary

This chapter presents a novel structural damage identification approach termed as SDI-DenseNet, and investigates its effectiveness and accuracy by numerical and experimental studies. Dense connectivity and feature concatenation are exploited in the architecture of the proposed SDI-DenseNet. Diverse levels of features are preserved and reused to optimize the network, which is especially beneficial to the task using time-domain responses. Besides, dense connectivity mitigates gradient vanishing and strengthens feature propagation and information flow. Time-domain acceleration responses are used to identify damages of structures. The proposed SDI-DenseNet composes of dense blocks and transition layers,

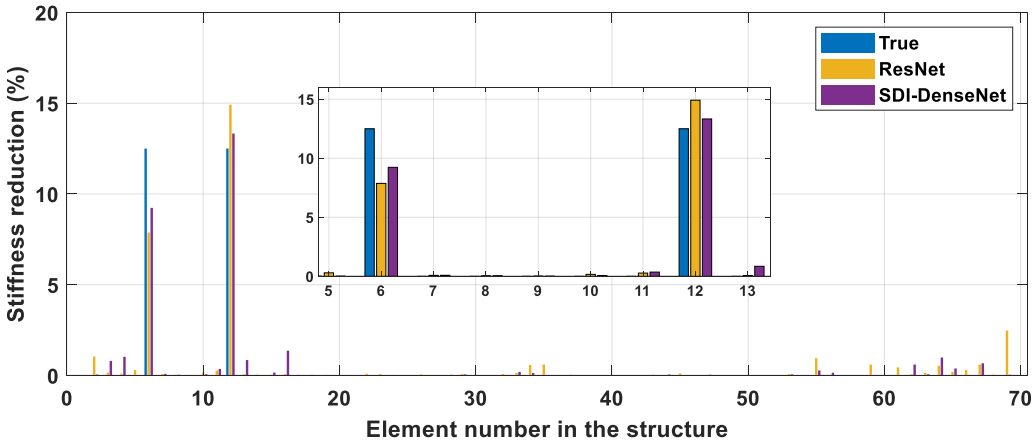

Figure 5-13: Damage identification results for the two-element damage case of the structure.

which correlates structural damages with the measured acceleration responses. A simply supported beam structure is investigated in the numerical study for structural damage identification considering both modelling uncertainties and measurement noise. A seven-storey steel frame structure in the laboratory is employed to evaluate the performance of the proposed SDI-DenseNet, in comparison with the CNN and ResNet. Both the numerical and experimental studies demonstrate the superior performance of SDI-DenseNet for identifying structural damage locations and quantifying damage severities. Overall, SDI-DenseNet fulfills good accuracy, robustness, and computational efficiency in structural damage identification applications.

Chapter 6

On the Invertibility of Residual Neural Networks

As described in Chapters 3, 4, and 5, several advanced deep learning based methods are used to develop comprehensive frameworks for structural health monitoring (SHM) applications, achieving good accuracy and efficiency in the identification of structural damages. The progress achieved in this thesis indicates the potential of implementing these deep learning-based SHM frameworks in real-world systems or structures. If these frameworks are to be implemented in real-world SHM projects, rigorous security and robust evaluation are critical since even the slightest flaw in deep learning models may have devastating effects.

Szegedy et al. [98] first showed that deep neural network models are vulnerable to adversarial attacks in 2013. For instance, image classifiers can be fooled by adding an imperceptible perturbation to the original input image. Since this revelation, there has been extensive research [51,68,114] on adversarial attacks or the generation of adversarial examples in order to develop robust neural networks against adversarial attacks. Recent work by Champneys et al. [15] shows that adversarial attacks can also be a real threat to SHM frameworks built on data-driven deep learning models. It is believed that the investigation of adversarial attacks can help improve the robustness and security of data-driven SHM models, even though limited progress has been made on this issue to the best of our knowledge.

A nice property of invertible neural networks is that it defines a bijective mapping from

[0]Part of this chapter is reprinted, with permission, from [Wang, R., An, S., Liu, W., & Li, L. Fixed-point algorithms for inverse of residual rectifier neural networks. *Mathematical Foundations of Computing*, 4(1), p.31. © 2021 AIMS. DOI: 10.3934/mfc.2020024].

the input space to the output space and one can invert the latent representation back to its corresponding input signal. With this property, people can have a better understanding on transformations of hidden layers of deep neural networks. A brief introduction on invertible neural networks and their applications are presented in Section 2.3.3. Specifically, some recent works explore adversarial attacks via invertible neural networks. Dolatabadi et al. [26] introduces AdvFlow: a novel adversarial attack model on image classifiers that exploits the capacity of invertible neural networks to represent the density of adversarial examples around a given target image. It was observed that AdvFlow improved the attack success rate over other state-of-the-art approaches [36,48,67] on adversarially trained classifiers. Bai et al. [8] propose a novel method to generate adversarial examples via i-ResNets [12] which are invertible neural network models with contractive Lipschitz constants. The invertibility property of neural networks can be used to investigate the difference between legitimate and adversarial examples. In other words, a better understanding of the distribution of perturbations may help to increase the robustness of classifiers against adversarial examples.

Motivated by the above observations, we investigate the invertible conditions of neural networks. The invertible property of networks preserves all the information of the original input that may help us to get a better understanding of the latent representations of the invertible models and can be potentially used to improve the robustness and security of the SHM frameworks. In this chapter, we first propose a novel fixed-point algorithm that requires a weak constraint on the weights to invert a residual rectifier network in Section 6.1. Next, we further reduce the constraints under which a residual block is invertible, and a tight condition is proposed in Section 6.2. Experimental results on structural damage identification are also presented in Section 6.2, followed by a discussion on adversarial attacks and how invertible neural networks may help to improve the robustness of SHM frameworks. Lastly, we conclude this chapter in Section 6.3.

6.1 Fixed-Point Algorithms for Inverse of Residual Rectifier Neural Networks

One of the fundamental difficulties in understanding the behaviour of deep neural networks is the loss of information due to the rectifier activation wherein the information of the negative components is discarded. As a consequence, the transformations of hidden layers may not be

invertible, and the input signals cannot be recovered from their hidden layers. Many works have investigated the invertibility of visual representations [27, 72, 94] to open the black box of deep neural networks and understand deep representations, but have observed significant information loss of the input signal with increasing depth.

In this section, we investigate the conditions under which the hidden layers of rectifier neural networks are invertible. Under these conditions there is no information loss in hidden layer transformations, and the inputs are recoverable from hidden layers. A recent work (i-ResNets [12]) presents a contractive condition on the Lipschitz-constant of the convolution path under which the residual unit is invertible. In this section, we present some weaker conditions and propose new inverse algorithms for the invertible residual networks.

The rest of this section is organised as follows. Section 6.1.1 introduces the concept of nonsingularity for rectifier linear transform and present the conditions under which the rectifier linear transform is nonsingular and thus invertible. Section 6.1.2 investigates the inverse of the residual units. Section 6.1.3 exhibits the architecture of the proposed simple invertible network whereas Section 6.1.4 provides experimental results.

6.1.1 Inverse of Rectifier Linear Transform

In this section, we consider the rectifier linear equation

$$\max(0, \mathbf{x}) = A\mathbf{x} + \mathbf{b} \tag{6.1}$$

and investigate the conditions on A under which Eq.(6.1) has a unique solution for any given $\mathbf{b}$. Similar to linear transforms, these conditions are closely related to the nonsingularity of the rectifier linear transform $\{\max(0, \mathbf{x}) - A\mathbf{x}\}$. A transform from $\mathbb{R}^n$ to $\mathbb{R}^m$, namely $f(\mathbf{x})$, is called nonsingular if $f(\mathbf{x}_1) \neq f(\mathbf{x}_2)$ holds for any $\mathbf{x}_1 \neq \mathbf{x}_2$. Similarly, it is called singular if there exists $\mathbf{x}_1 \neq \mathbf{x}_2$ such that $f(\mathbf{x}_1) = f(\mathbf{x}_2)$. Note that a linear transform $f(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$ is nonsingular if and only if A is a nonsingular matrix. Apparently, if $\{\max(0, \mathbf{x}) - A\mathbf{x}\}$ is nonsingular, the solution to (6.1) is unique for any $\mathbf{b}$. On the other hand, if the solution to (6.1) is unique for any $\mathbf{b}$, $\{\max(0, \mathbf{x}) - A\mathbf{x}\}$ must be a nonsingular transform.

Next we present a sufficient condition on A such that Eq. (6.1) has a unique solution for any given $\mathbf{b}$.

Theorem 1 *If*

$$\frac{1}{2}(A + A^T) < AA^T \tag{6.2}$$

then the solution of $\mathbf{x}$ *to Eq. (6.1) is unique for any given* $\mathbf{b}$.

Theorem 1 shows that when A is large enough, the solution of (6.1) is unique. For instance, let A be a nonsingular matrix, if $(A + A^T)$ is negative definite, then (6.2) holds and the solution to Eq. (6.1) is unique. Otherwise, let $\sigma_1 > 0$ be the largest eigenvalue of $(A+A^T)$ and $\sigma_2 > 0$ be the smallest eigenvalue of $A^T A$. Then for $\alpha > \sigma_1/(2\sigma_2)$, αA satisfies (6.2) and thus $(\max(0, \mathbf{x}) - \alpha A\mathbf{x})$ is nonsingular. This shows that for any non-singular A, if α is sufficiently large, $(\max(0, \mathbf{x}) - \alpha A\mathbf{x})$ is then nonsingular.

Proof: Note that $\max(0, \mathbf{x}) = \max(0, -\mathbf{x}) + \mathbf{x}$. Equation (6.1) implies that

$$\max(0, -\mathbf{x}) = (A - I)\mathbf{x} + \mathbf{b}. \tag{6.3}$$

Suppose Equation (6.1) has two different solutions, namely $\mathbf{x}_1$ and $\mathbf{x}_2$. Then we have

$$\begin{aligned} &\begin{bmatrix} \max(0, \mathbf{x}_1) - \max(0, \mathbf{x}_2) \\ \max(0, -\mathbf{x}_1) - \max(0, -\mathbf{x}_2) \end{bmatrix} \\ &= \begin{bmatrix} A \\ A - I \end{bmatrix} (\mathbf{x}_1 - \mathbf{x}_2). \end{aligned} \tag{6.4}$$

From *Proposition 8* of [5], it follows that

$$\left\| \begin{bmatrix} \max(0, \mathbf{x}_1) - \max(0, \mathbf{x}_2) \\ \max(0, -\mathbf{x}_1) - \max(0, -\mathbf{x}_2) \end{bmatrix} \right\| \leq \|\mathbf{x}_1 - \mathbf{x}_2\| \tag{6.5}$$

which, from Eq. (6.4), implies

$$\left\| \begin{bmatrix} A \\ A - I \end{bmatrix} (\mathbf{x}_1 - \mathbf{x}_2) \right\| \leq \|\mathbf{x}_1 - \mathbf{x}_2\| . \tag{6.6}$$

Since $A^TA \geq \frac{1}{2}(A + A^T)$, we have

$$\begin{bmatrix} A \\ A-I \end{bmatrix}^T \begin{bmatrix} A \\ A-I \end{bmatrix} = 2A^TA - A - A^T + I \geq I \tag{6.7}$$

and therefore

$$\left\| \begin{bmatrix} A \\ A-I \end{bmatrix} (\mathbf{x}_1 - \mathbf{x}_2) \right\| > \|\mathbf{x}_1 - \mathbf{x}_2\|, \forall \mathbf{x}_1 \neq \mathbf{x}_2. \tag{6.8}$$

Hence (6.6) holds only when $\mathbf{x}_1 = \mathbf{x}_2$, which implies that the solution to Eq (6.1) is unique.

■

Next we present an algorithm to solve the rectifier linear equation (6.1), that is, compute $\mathbf{x}$ with given A and $\mathbf{b}$.

From Eq.(6.1) and note that $\max(0, -\mathbf{x}) = \max(0, \mathbf{x}) - \mathbf{x}$, we have

$$\begin{bmatrix} \max(0, \mathbf{x}) \\ \max(0, -\mathbf{x}) \end{bmatrix} = \begin{bmatrix} A \\ A-I \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{b} \\ \mathbf{b} \end{bmatrix} \tag{6.9}$$

and therefore

$$\begin{aligned} \mathbf{x} = & \begin{bmatrix} A \\ A-I \end{bmatrix}^\dagger \begin{bmatrix} \max(0, \mathbf{x}) \\ \max(0, -\mathbf{x}) \end{bmatrix} \\ & - \begin{bmatrix} A \\ A-I \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{b} \\ \mathbf{b} \end{bmatrix}. \end{aligned} \tag{6.10}$$

The proposed fixed-point inverse algorithm is described as below:

1) Let $\mathbf{x}_0 = \mathbf{b}$*;*

2) for $k = 1, 2, \cdots,$ *do*

$$\begin{aligned} \mathbf{x}_k = & \begin{bmatrix} A \\ A-I \end{bmatrix}^\dagger \begin{bmatrix} \max(0, \mathbf{x}_{k-1}) \\ \max(0, -\mathbf{x}_{k-1}) \end{bmatrix} \\ & - \begin{bmatrix} A \\ A-I \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{b} \\ \mathbf{b} \end{bmatrix}. \end{aligned} \tag{6.11}$$

until $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon$ *where* ϵ *is a threshold for convergence.*

Note that the maximum singular value of $\begin{bmatrix} A \\ A-I \end{bmatrix}^{\dagger}$ is less than 1 under the condition in Theorem 1, and

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \begin{bmatrix} A \\ A-I \end{bmatrix}^{\dagger} \begin{bmatrix} \max(0, \mathbf{x}_k) - \max(0, \mathbf{x}_{k-1}) \\ \max(0, -\mathbf{x}_k) - \max(0, -\mathbf{x}_{k-1}) \end{bmatrix} \tag{6.12}$$

we have

$$\begin{aligned} \|\mathbf{x}_{k+1} - \mathbf{x}_k\| &< \left\| \begin{bmatrix} \max(0, \mathbf{x}_k) - \max(0, \mathbf{x}_{k-1}) \\ \max(0, -\mathbf{x}_k) - \max(0, -\mathbf{x}_{k-1}) \end{bmatrix} \right\| \\ &< \|\mathbf{x}_k - \mathbf{x}_{k-1}\|. \end{aligned} \tag{6.13}$$

Hence, this algorithm converges if the condition (6.2) in Theorem 1 is satisfied. From Theorem 1, the solution is unique and therefore the algorithm converges to its unique solution.

6.1.2 Inverse of Residual Units

For residual networks with rectifier as the nonlinear activation function, we have

$$\mathbf{x}_{k+1} = \max(0, W_k \mathbf{x}_k + \mathbf{b}_k) + \mathbf{x}_k \tag{6.14}$$

where W_k is the linear transformation matrix which may include convolution and batch normalization since both of them are essentially linear transforms.

Next, we consider the conditions under which $\mathbf{x}_k$ is recoverable from $\mathbf{x}_{k+1}$. Let $\mathbf{z}_k = W_k \mathbf{x}_k + \mathbf{b}_k$ and assume that W_k is nonsingular, we have

$$\mathbf{x}_k = W_k^{-1} \mathbf{z}_k + \hat{b}_k \tag{6.15}$$

where $\hat{\mathbf{b}}_k = -W_k^{-1} \mathbf{b}_k$. Rewrite (6.14) in terms of $\mathbf{z}_k$, we have

$$\mathbf{x}_{k+1} = \max(0, \mathbf{z}_k) + W_k^{-1} \mathbf{z}_k + \hat{b}_k \tag{6.16}$$

that is,

$$\max(0, \mathbf{z}_k) = -W_k^{-1} \mathbf{z}_k + \mathbf{x}_{k+1} - \hat{b}_k. \tag{6.17}$$

From **Theorem 1**, if

$$W_k^{-T} W_k^{-1} \geq \frac{1}{2}(-W_k^{-1} - W_k^{-T}), \tag{6.18}$$

i.e.,

$$I \geq \frac{1}{2}(-W_k - W_k^T), \tag{6.19}$$

then $\mathbf{z}_k$ is recoverable from $\mathbf{x}_{k+1}$. Since $\mathbf{x}_k$ is recoverable from $\mathbf{z}_k$, $\mathbf{x}_k$ is recoverable from $\mathbf{x}_{k+1}$ as well.

Hence, we have

Corollary 2 *If W_k is nonsingular and*

$$-(W_k + W_k^T) \leq 2I, \tag{6.20}$$

then $\mathbf{x}_k$ is recoverable from $\mathbf{x}_{k+1}$ using Eq.(6.14). That is, the transform from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ in a residual hidden layer is nonsingular.

Next, we present a fixed-point inverse algorithm for the inverse of a residual unit. For the inverse of a residual unit, one needs to solve the following rectifier equation:

$$\max(0, W\mathbf{x} + \mathbf{b}) = -\mathbf{x} + \mathbf{c} \tag{6.21}$$

which is equivalent to

$$\max(0, -W\mathbf{x} - \mathbf{b}) = -(I + W)\mathbf{x} + \mathbf{c} - \mathbf{b}. \tag{6.22}$$

By combining these two equations, we have

$$\begin{bmatrix} I \\ I + W \end{bmatrix} \mathbf{x} = -\begin{bmatrix} \max(0, W\mathbf{x} + \mathbf{b}) \\ \max(0, -W\mathbf{x} - \mathbf{b} \end{bmatrix} + \begin{bmatrix} \mathbf{c} \\ \mathbf{c} - \mathbf{b} \end{bmatrix} \tag{6.23}$$

and therefore

$$\begin{aligned} \mathbf{x} \;=\; & -\begin{bmatrix} I \\ I + W \end{bmatrix}^{\dagger} \begin{bmatrix} \max(0, W\mathbf{x} + \mathbf{b}) \\ \max(0, -W\mathbf{x} - \mathbf{b} \end{bmatrix} \\ & + \begin{bmatrix} I \\ I + W \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{c} \\ \mathbf{c} - \mathbf{b} \end{bmatrix} \end{aligned} \tag{6.24}$$

The proposed inverse algorithm is as below

1) Let $\mathbf{x}_0 = \begin{bmatrix} I \\ I+W \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{c} \\ \mathbf{c}-\mathbf{b} \end{bmatrix}$;

2) for $k = 1, 2, \cdots,$ *do*

$$\begin{aligned} \mathbf{x}_k &= -\begin{bmatrix} I \\ I+W \end{bmatrix}^{\dagger} \begin{bmatrix} \max(0, W\mathbf{x}_{k-1}+\mathbf{b}) \\ \max(0, -W\mathbf{x}_{k-1}-\mathbf{b}) \end{bmatrix} \\ &\quad + \begin{bmatrix} I \\ I+W \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{c} \\ \mathbf{c}-\mathbf{b} \end{bmatrix} \end{aligned} \tag{6.25}$$

until $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon$ *where* ϵ *is a threshold for convergence.*

For convolution layers, we need to carefully consider the implementation of the pseudo-inverse of $\begin{bmatrix} I \\ I+W \end{bmatrix}$ since the size of W is extremely large. We need to find an efficient way to compute the pseudo-inverse through convolutions/deconvolutions. Note that the pseudo-inverse of $\begin{bmatrix} I \\ I+W \end{bmatrix}$ equals to:

$$\begin{aligned} &\left(\begin{bmatrix} I \\ I+W \end{bmatrix}^T \begin{bmatrix} I \\ I+W \end{bmatrix} \right)^{-1} \begin{bmatrix} I & I+W^T \end{bmatrix} \\ &= \left(2I + W + W^T + W^T W\right)^{-1} \begin{bmatrix} I & I+W^T \end{bmatrix}. \end{aligned} \tag{6.26}$$

Then Eq. (6.25) can be rewritten as

$$\begin{aligned} \mathbf{x}_k &= \left(2I + W + W^T + W^T W\right)^{-1} \begin{bmatrix} I & I+W^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \\ &= \left(2I + W + W^T + W^T W\right)^{-1} \left(\mathbf{y}_1 + \left(I + W^T\right)\mathbf{y}_2\right) \\ &= \left(I + (I+W)^T (I+W)\right)^{-1} \left(\mathbf{y}_1 + \left(I + W^T\right)\mathbf{y}_2\right) \end{aligned} \tag{6.27}$$

where $\mathbf{y}_1 = \mathbf{c} - \max(0, W\mathbf{x}_{k-1} + \mathbf{b})$ and $\mathbf{y}_2 = \mathbf{c} - \mathbf{b} - \max(0, -W\mathbf{x}_{k-1} - \mathbf{b})$.

Let F be the filter for the convolution associated with W. W^T can be implemented using convolution with filter F^T. To obtain $\left(I + W^T\right)\mathbf{y}_2$, one can implement it as a convolution $F_1 * \mathbf{y}_2$. Here, F_1 is the associated filter, that is $F_1 = (C + F^T)$, while C is a matrix whose

central element is 1 and other elements are zero.

Then Eq. (6.27) can be simplified as

$$\left(I + (I+W)^T (I+W)\right) \mathbf{x}_k \;=\; \mathbf{y} \tag{6.28}$$

where $\mathbf{y}$ equals to $\left(\mathbf{y}_1 + \left(I + W^T\right)\mathbf{y}_2\right)$. It can be implemented as a convolution as well, that is $F_2 * \mathbf{x}_k = \mathbf{y}$. F_2 is the associated convolution filter of $\left(I + (I+W)^T (I+W)\right)$, which can be easily obtained via $F_2 = C_2 + (C_1 + F)^T * (C_1 + F)$. Note that C_1 and C_2 are matrices whose central element is 1 and other elements are zero. Finally, we can compute $\mathbf{x}_k$ through deconvolution [91] once F_2 and $\mathbf{y}$ are obtained.

Comparison to the existing fixed-point algorithm

In [12], another fixed-point algorithm is presented to inverse a residual network. When applied to Eq. (6.21), this algorithm can be described as below.

1) Let $\mathbf{x}_0 = \mathbf{c}$*;*

2) for $k = 1, 2, \cdots,$ *do*

$$\mathbf{x}_k \;=\; \mathbf{c} - \max(0, W\mathbf{x}_{k-1} + \mathbf{b}) \tag{6.29}$$

until $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon$ *where* ϵ *is a threshold for convergence.*

For the convergence of this algorithm, the maximum singular value of W should be less than 1, that is

$$-1 < \frac{\mathbf{x}^T W \mathbf{x}}{\mathbf{x}^T \mathbf{x}} < 1, \forall\, \mathbf{x} \tag{6.30}$$

and therefore

$$-2I < W + W^T < 2I \tag{6.31}$$

which is much stronger than the condition in Eq.(6.20), *i.e.*, $(W_k + W_k^T) \geq -2I$, for the convergence of the proposed algorithm.

6.1.3 Invertible Network Architecture

To test the proposed fixed-point inverse algorithm, we construct a simple invertible network, which is illustrated in Figure 6-1. The components BN and pointwise convolutions are

invertible [57] since they are nonsingular linear transformations. A deconvolution technique called Inverse Filtering [91] is used to reverse the effects of general convolution operation on the input signal. Next, the architecture of the proposed invertible network is presented in detail, followed by the inverting operation of the bidirectional ReLU layer.

Network Architecture

As presented in Eq.(6.14), W_k in a residual unit is the linear transformation matrix which may be a convolution layer, or a batch normalization layer, or a combination of them. However, batch normalization and the 3×3 convolution cannot be inverted together in the residual unit. Therefore, we move the batch normalization out of the residual unit when designing the network architecture. In our experiments, only one residual unit is used in the network. The architecture of the specially designed residual network is illustrated in Figure 6-1, in which the layers in green are invertible.

The network starts with a regular $3 \times 3 \times 5$ convolution layer, which expands the input to a higher dimension. Then, the feature dimension is doubled by a bidirectional ReLU activation. For a regular ReLU activation function, it is defined as $f(\mathbf{x}) = \mathbf{x}^+ = \max(0, \mathbf{x})$, where $\mathbf{x}$ is the input to a neuron. Only the positive part of $\mathbf{x}$ is preserved, hence it is impractical to reverse this activation operation in a normal layer. To address this issue, we use a bidirectional ReLU that preserves all the information of $\mathbf{x}$. It can be defined as

$$f(\mathbf{x}) = \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix} = \begin{bmatrix} \max(0, \mathbf{x}) \\ \max(0, -\mathbf{x}) \end{bmatrix}, \tag{6.32}$$

where the square brackets denote a matrix concatenation in channel dimension. That is why the feature dimension is doubled after a bidirectional ReLU activation. An invertible residual unit is added afterwards. Here we use a $3 \times 3 \times 10$ filter in depthwise conv layer in the residual unit after which a $1 \times 1 \times 10$ in pointwise convolution layer and a batch normalization layer are added. Finally one dropout layer and two fully-connected layers are introduced for a classification problem. The proposed residual network achieves a comparable performance for classification in our experiments. Experiment results are presented in Section 6.1.4.

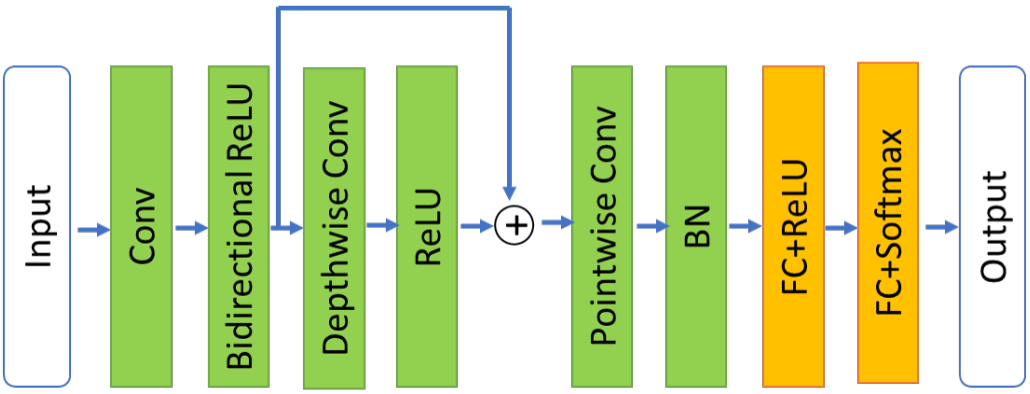

Figure 6-1: The proposed simple residual network architecture.

Invertibility of Bidirectional ReLU

The bidirectional ReLU used after the first conv layer in the proposed network preserves all the information of $\mathbf{x}$ (Eq.(6.32)). Thus it has the property of recovering the original input data, which can be achieved by

$$\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^- = \max(0, \mathbf{x}) - \max(0, -\mathbf{x}). \tag{6.33}$$

6.1.4 Experimental Results

Experiments are conducted to validate the performance of the proposed fixed-point inverse algorithms for the rectifier linear transform in the residual units in Eq.(6.25). We compared our inverse algorithms with the existing inverse algorithm presented in [12], which also inverts a residual layer via a fixed-point method. The experimental results demonstrate that the proposed inverse algorithms are more widely applicable than the method we compared as claimed theoretically in Section 6.1.2. In other words, the proposed fixed-point algorithm works on more cases under weaker conditions. The experiment details are described below.

A special residual network using only one residual unit is used and its architecture is described in Section 6.1.3. As shown in Figure 6-1, the layers in green are all invertible. Based on Theorem 1, if the weights in a residual layer are small enough, the residual unit is guaranteed invertible. Thus, a max norm constraint(*e.g.*, max norm is 0.5) is enforced on the kernels of the depthwise convolution layer in our implementation. The value is selected based on the validation dataset.

To validate the invertibility of the proposed method and visualize the recovered input

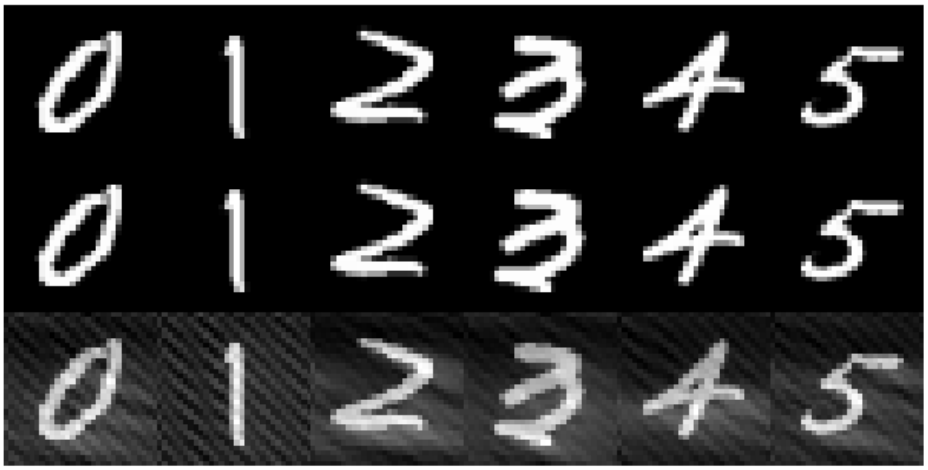

Figure 6-2: Comparison of recovered images to original digit images. The 1st row illustrates the original images, whereas the 2nd and 3rd rows show the recovered images from the proposed fixed-point method and the existing fixed-point method, respectively.

signal in a more intuitive manner, an image dataset, MNIST [62] data set, is used in the experiments. The experimental results indicate that the proposed residual network shows good invertibility under this condition while achieves a good performance for classification. The MNIST database consists of 70,000 handwritten digits examples. The whole database is split into three subsets: 50,000 for training, 10,000 for validation and another 10,000 for testing. Each digit is a 28×28 grey scale image. 60 epochs are executed while Adam optimizer with learning rate 0.001 is used during the training stage. The test classification error of the proposed residual network is 0.88% , which is comparable to the state-of-art performance on MNIST without data augmentation [87], despite using a small network with weight constraints enforced.

The invertibility of the proposed architecture is then validated based on the trained classification model. The layers shown in green in Figure 6-1 are all invertible as discussed above. We use the inverse algorithm presented in Eq.(6.25) to recover the original images from the residual unit. Again, the proposed inverse algorithm for residual units is compared with the existing fixed-point method. Six samples are randomly selected from the training set for recovering (Figure 6-2). It can be seen that each sample is successfully reconstructed from the invertible model using the proposed method. Figure 6-2 also demonstrates that the recovered images of the proposed approach are much better than those recovered using the existing fixed-point method [12].

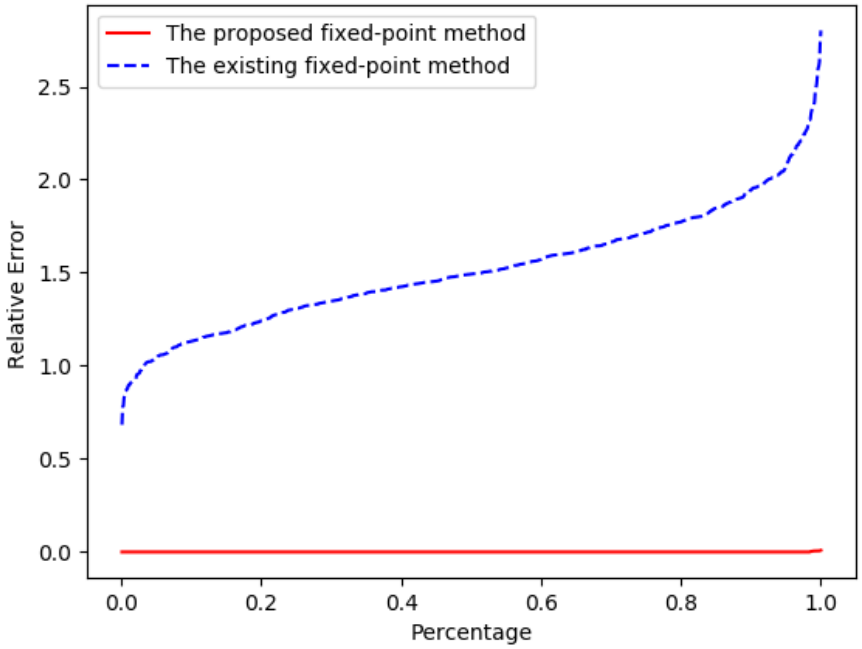

Figure 6-3: Relative error rates (%) of the recovered images. One hundred samples per each class, in total 1000 samples, were chosen and recovered.

To further study the applicability of our model, we conducted this recovering on 1000 MNIST images and the relative error is shown in Figure 6-3. 100 samples per class are randomly chosen and recovered. We calculate the relative errors between the original images $\mathbf{x}$ and the recovered images $\widehat{\mathbf{x}}$ based on $E_r = \frac{\|\widehat{\mathbf{x}}-\mathbf{x}\|}{\|\mathbf{x}\|} \times 100\%$. The relative errors for all recovered images using the proposed method are zeros. The same validation is conducted on the existing fixed-point method, and the relative error for all cases are above 68%. Thus it can be concluded that the existing method fails to invert the trained residual network model.

In summary, the proposed simple invertible network performs well in classification and its invertibility is guaranteed at the same time. Besides, the proposed fixed-point inverse algorithm is more widely applicable, in the sense that it works on more cases under a weaker condition comparing to the existing fixed-point method.

6.2 General Invertible Residual Blocks in Deep Learning Networks

Section 6.1 considers the same problem of invertible residual networks as [12] but presents weaker conditions under which a residual block is invertible. This section further reduces the constraints and investigate necessary and sufficient conditions under which a residual block with one ReLu layer is invertible. For residual blocks with general densely or sparsely connected layers, we will show that the inverse problem of residual blocks is closely related to the well known Linear Complementarity Problem (LCP) [21]. Based on this connection, for residual blocks with one ReLU layer inside the residual path, we provide a necessary and sufficient condition for a residual block to be invertible under a mild assumption on the linear transform in the residual path. For standard residual blocks with Batch Normalisation (BN), convolution (CONV), and ReLU layers, we show that, if the convolution is implemented with zero padding on the top and the left sides, their invertibility depends only on the scaling factors of the BN layers and one element of each CONV filter. The provided conditions are necessary and sufficient, and also are easy to check. The experimental results demonstrate that this condition can be easily imposed by initializing the elements of the convolution filters.

The remainder of this section is organized as follows. Sections 6.2.1 and 6.2.2 present the invertibility of residual blocks for vectors and convolutions, respectively. Next, Section 6.2.3 reports the experimental results. Finally, Section 6.2.4 discusses the promising direction of adversarial attacks using the proposed invertible residual network.

6.2.1 Invertibility of Residual Blocks for Vectors

Consider a general residual block in Figure 6-4, where the relationship between the input $\mathbf{x}$ and the output $\mathbf{y}$ can be described as:

$$\begin{aligned} \mathbf{z}_1 &= W_1\mathbf{x} + \mathbf{b}_1 \\ \mathbf{y} &= \mathbf{x} + W_2 \max(0, \mathbf{z}_1) + \mathbf{b}_2 \end{aligned} \tag{6.34}$$

where $\mathbf{z}_1$ is the output of the first linear transform and also the input of the rectifier layer (*i.e.*, ReLU).

The linear transformation can be a convolution layer, a fully connected layer, a batch

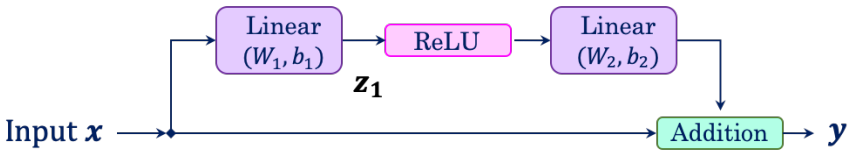

Figure 6-4: General Residual Blocks with One Layer of ReLU.

normalization layer, or a combination of them such as a convolution layer followed by a batch normalization layer.

In this section, we are interested in finding the conditions under which the input $\mathbf{x}$ can be recovered from the output $\mathbf{y}$. An invertible residual block can change the shape of the data distribution but will never make different inputs identical in the output.

Interestingly, we will show that the inverse problem of residual networks is connected to the well-known linear complementarity problem (LCP) [21] where a special type of matrices called P-matrix [73] is involved. Next, we will introduce P-matrix, LCP and then present the conditions under which a residual block is invertible.

P-Matrix

A matrix is called a P-matrix if all its principal minors are positive. A principal minor of a matrix is the determinant of a principal submatrix while a principal submatrix is a submatrix containing columns and rows from the same index set [92]. P-matrices are square matrices in which the principal minors are all positive. For example, a 2×2 matrix Q is a P-matrix if and only if $Q_{11} > 0, Q_{22} > 0$ and $|Q| = Q_{11}Q_{22} - Q_{12}Q_{21} > 0$ where Q_{ij} denote the element of Q in the i^{th} row and j^{th} column.

The Linear Complementarity Problem (LCP)

Given a real matrix $M \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{q} \in \mathbb{R}^n$, the linear complementarity problem $LCP(M, \mathbf{q})$ seeks vectors $\mathbf{z} \in \mathbb{R}^n$ and $\mathbf{w} \in \mathbb{R}^n$ which satisfy the following constraints:

$$\begin{aligned} \mathbf{w} &= \mathbf{q} + M\mathbf{z} \\ \mathbf{w} &\succeq 0 \\ \mathbf{z} &\succeq 0 \\ \mathbf{z}^T\mathbf{w} &= 0. \end{aligned} \tag{6.35}$$

where $\mathbf{z} \succeq 0$ represents that every element of $\mathbf{z}$ is greater than or equal to 0.

Proposition 3 *[21]. The Linear Complementarity Problem $LCP(M, \mathbf{q})$ in (6.35) has a unique solution for any $\mathbf{q} \in \mathbb{R}^n$ if and only if M is a P-matrix.*

Invertibility Conditions

Theorem 4 *Assume that W_1 is nonsingular. Equation (6.34) has a unique solution of $\mathbf{x}$ for every $\mathbf{y}$ if and only if $I + W_1W_2$ is a P matrix.*

Proof: From Eq. (6.34), we have

$$\mathbf{y} = \mathbf{x} + W_2 \max(0, W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2. \tag{6.36}$$

Multiplying W_1 and adding $\mathbf{b}_1$ on both sides of Eq. (6.36), with $\mathbf{z}_1 = W_1\mathbf{x} + \mathbf{b}_1$, we have

$$\begin{aligned} W_1\mathbf{y} + \mathbf{b}_1 &= W_1\mathbf{x} + \mathbf{b}_1 + W_1W_2 \max(0, W_1\mathbf{x} + \mathbf{b}_1) + W_1\mathbf{b}_2 \\ &= W_1W_2 \max(0, \mathbf{z}_1) + \mathbf{z}_1 + W_1\mathbf{b}_2 \\ &= (I + W_1W_2) \max(0, \mathbf{z}_1) - \max(0, -\mathbf{z}_1) + W_1\mathbf{b}_2 \end{aligned} \tag{6.37}$$

which is equivalent to the following equation

$$\max(0, -\mathbf{z}_1) = (I + W_1W_2) \max(0, \mathbf{z}_1) + W_1(\mathbf{b}_2 - \mathbf{y}) - \mathbf{b}_1. \tag{6.38}$$

Now let $M = I + W_1W_2, \mathbf{w} = \max(0, -\mathbf{z}_1)$, $\mathbf{z} = \max(0, \mathbf{z}_1)$ and $\mathbf{q} = W_1(\mathbf{y} - \mathbf{b}_2) - \mathbf{b}_1$, then we have $\mathbf{w} \succeq 0, \mathbf{z} \succeq 0, \mathbf{z}^T\mathbf{w} = 0$ and $\mathbf{w} = \mathbf{q} + M\mathbf{z}$. Therefore, solving Equation (6.38) for $\mathbf{z}_1$ is equivalent to solving the $LCP(M, \mathbf{q})$. From Proposition 3, the solution of $\mathbf{z}_1$ to Eq. (6.38) is unique for any $\mathbf{y}$ if and only if $I + W_1W_2$ is a P-matrix. Note that $\mathbf{z}_1 = W_1\mathbf{x} + \mathbf{b}_1$ and W_1 is nonsingular, the solution of $\mathbf{x}$ to Eq. (6.36) for any $\mathbf{y}$ is unique if and only if $I + W_1W_2$ is a P-matrix, and this completes the proof.

□

Theorem 4 states that, if W_1 is nonsingular and $I + W_2W_1$ is a P-matrix, the function

$$f(\mathbf{x}) = \mathbf{x} + W_2 \max(0, W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \tag{6.39}$$

is a bijective mapping. That is, for any two different inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\mathbb{R}^n$, we have $f(\mathbf{x}_1) \neq f(\mathbf{x}_2)$; and for any $\mathbf{y} \in \mathbb{R}^n$, there is one and only one $\mathbf{x}$ such that $f(\mathbf{x}) = \mathbf{y}$. Each element in the input domain $\mathbb{R}^n$ is paired with exactly one element in the range $\mathbb{R}^n$ of the mapping.

We conjecture that Theorem 4 still holds when the requirement on the nonsingularity of W_1 is dropped. Next we show that this conjecture is true for one dimensional inputs and outputs. The following Corollary is also particularly useful in investigating the invertibility of residual blocks with convolutions, which will be presented in Section 6.2.2.

Corollary 5 *Consider the equation*

$$f(x) = y \tag{6.40}$$

where $f(x) = x + w_2 \max(0, w_1 x + b_1) + b_2$ *is a function of a single variable* x*, and* w_1, w_2, b_1, b_2 *are given real numbers. Then* $f(x) = y$ *has a unique solution of* x *for any* $y \in \mathbb{R}$ *if and only if* $1 + w_1 w_2 > 0$*. Furthermore, when* $1 + w_2 w_1 > 0$*, the unique solution is*

$$x = \begin{cases} y - b_2, & \text{if } b_1 + w_1 y \leq 0 \text{ or } w_2 = 0; \\ \frac{y - b_2 - w_2 b_1}{1 + w_2 w_1}, & \text{otherwise.} \end{cases} \tag{6.41}$$

Proof: Theorem 4 implies that Corollary 5 holds when $w_1 \neq 0$. When $w_1 = 0$, the solution is $x = y - w_2 \max(0, b_1) - b_2$ which is obviously unique.

For the correctness of (6.41), it is easy to check that, if $w_2 = 0$ or $b_1 + w_1 y \leq 0$, $x = y$ is a solution to Eq. (6.40). If $b_1 + w_1 y > 0$ and $w_2 \neq 0$, then $x = y$ is not a solution and therefore the solution must satisfy $w_1 x + b_1 > 0$. Hence, Equation 6.40 is then equivalent to $x + w_2(w_1 x + b_1) = y - b_2$ which implies that $x = \frac{y - b_2 - w_2 b_1}{1 + w_1 w_2}$.

□

Next we extend this scalar case to high dimensional cases with a special type of transformation where the transformation matrices are lower triangular matrices. In Section 6.2.2, we will show that, when we rewrite the images as vectors, the associated transformation matrices of convolutions are lower triangular if zero padding is applied on the left and the top sides of the images. Moreover, the invertibility of the residual units with lower triangular transforms only depends on the diagonals of the triangular matrices.

Theorem 6 *Assume that W_1 and W_2 are lower triangular matrices. Then Equation (6.34) has a unique solution* $\mathbf{x}$ *for every* $\mathbf{y}$ *if and only if $I + W_2W_1$ is a P matrix or equivalently $1 + W_2[k,k]W_1[k,k] > 0$ for any $k = 1, 2, \cdots, n$ where $W_1[k,k]$ and $W_2[k,k]$ denote the k^{th} diagonals of W_1 and W_2 respectively.*

Proof: Let $x_k, b_{1,k}, b_{2,k}$ be the k^{th} elements of $\mathbf{x}$, $\mathbf{b}_1$ and $\mathbf{b}_2$ respectively, and

$$W_i = \begin{bmatrix} W_i[11] & 0 & \cdots & 0 \\ W_i[21] & W_i[22] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ W_i[n1] & W_i[22] & \cdots & W_i[nn] \end{bmatrix}, i = 1, 2. \tag{6.42}$$

The proof is proceeded by mathematics induction. We will first prove that Theorem 6 holds when $n = 1$. From (6.34), we have

$$x_1 + W_2[11] \max(0, W_2[11]x_1 + b_{1,1}) + b_{2,1} \quad = \quad y_1 \tag{6.43}$$

which, by Corollary 5, has a unique solution for any y_1 if only if $W_2[1,1]W_1[11] > -1$.

Now assume that Theorem 6 holds for $n = k - 1$, that is, the solutions of $x_1, \cdots, x_{k-1}$ to Eq. (6.34) for any $y_1, \cdots, y_{k-1}$ are unique if only if $W_2[ii]W_1[ii] > -1$ holds for any $1 \leq i \leq k - 1$. Next we prove that Theorem 6 holds when $n = k$. From (6.34), we have

$$x_k + \begin{bmatrix} W_2[k1] & \cdots & W_2[kk] \end{bmatrix} \begin{bmatrix} \max(0, W_1[11]x_1 + b_{1,1}) \\ \vdots \\ \max\left(0, \sum_{i=1}^{k} W_2[ki]x_i + b_{1,k}\right) \end{bmatrix} + b_{2,k} = y_k \tag{6.44}$$

which is equivalent to

$$x_k + W_2[kk] \max(0, W_1[kk]x_k + \hat{b}_{1,k}) + \hat{b}_{2,k} = y_k \tag{6.45}$$

where

$$\begin{array}{rcl} \hat{b}_{2,k} & \triangleq & b_{2,k} + \sum_{i=1}^{k-1} W_2[ki] \max\left(0, \sum_{j=1}^{k-1} W_1[kj]x_j + b_{1,i}\right) \\ \hat{b}_{1,k} & \triangleq & b_{1,k} + \sum_{i=1}^{k-1} W_1[ki]x_i. \end{array} \tag{6.46}$$

Hence, when $x_1, \cdots, x_{k-1}$ are given, by Corollary 5, the solution of x_k to Eq. (6.45) is unique for any y_k if and only if $1 + W_2[kk]W_1[kk] > 0$. Since the solutions of $x_1, \cdots, x_{k-1}$

to Eq. (6.34) for any $y_1, \cdots, y_{k-1}$ are unique if only if $W_2[ii]W_1[ii] > -1$ holds for any $1 \leq i \leq k-1$, Theorem 6 holds for $n = k$. By mathematical induction, Theorem 6 holds for any positive integer n.

□

The Inverse Algorithm

Based on Eq. (8) in Corollary 3 and Eqs. (10, 12, 13), we present the pseudo code of the proposed inverse algorithm below.

Algorithm 1 Inverse algorithm for a residual unit described in Eq. (1).

Require: $\mathbf{y} = \left[y_1, y_2, \cdots, y_n\right]^\top$ and $W_1, W_2, \mathbf{b}_1, \mathbf{b}_2$.

Compute the first element: x_1, by solving Eq. (10).
if $W_2[11] = 0$ or $W_1[11]y_1 + b_{1,1} \leq 0$ **then**
  return $x_1 = y_1 - b_{2,1}$
else
  return $x_1 = \frac{y_1 - b_{2,1} - W_2[11]b_{1,1}}{1 + W_2[11]W_1[11]}$.
end if
Compute the rest elements:
for $k = 2, 3, \cdots, n$ **do**
  Compute $\hat{b}_{1,k}$ and $\hat{b}_{2,k}$ using Eq. (13)
  Compute x_k, by solving Eq. (12).
  if $W_2[kk] = 0$ or $W_1[kk]y_1 + \hat{b}_{1,k} \leq 0$ **then**
    return $x_k = y_k - b_{2,k}$
  else
    return $x_k = \frac{y_k - b_{2,k} - W_2[kk]b_{1,k}}{1 + W_2[kk]W_1[kk]}$
  end if
end for

Ensure: The recovered $\hat{\mathbf{x}} = \left[x_1, x_2, \cdots, x_n\right]^\top$.

6.2.2 Invertible Residual Blocks for Convolutions

Consider the convolution

$$Y = X * F \tag{6.47}$$

where

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}. \tag{6.48}$$

In order to make the size of the output Y equal to the size of the input X, zero padding is required. There are different ways of zero padding such as zero padding on 1) the top and the left sides; 2) the top and the right sides; 3) the bottom and the left sides; 4) the bottom and the right sides; and 5) all the four sides surrounding the images. In this chapter, we will consider the first zero padding method. For zero padding methods 2), 3) and 4), similar results can be obtained following the same procedure to transform the convolution of matrices into a triangular linear transform of vectors. However, for zero padding method 5), the convolution of matrices cannot be transformed into triangular linear transforms of vectors.

Now let

$$\hat{X} = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & x_{11} & x_{12} & \cdots & x_{1n} \\ 0 & 0 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} \tag{6.49}$$

be the zero-padded matrix of X, and let $\mathbf{y}_k^T, \mathbf{x}_k^T$ denote the k^{th} row of Y and X respectively (*i.e.*, $\mathbf{y}_k^T = [Y_{k1}, \cdots, Y_{kn}]$, $\mathbf{x}_k^T = [X_{k1}, \cdots, X_{kn}]$), we have

$$\begin{aligned} \mathbf{y}_1 &= W_1\mathbf{x}_1 \\ \mathbf{y}_2 &= W_2\mathbf{x}_1 + W_1\mathbf{x}_2 \\ \mathbf{y}_k &= W_3\mathbf{x}_{k-2} + W_2\mathbf{x}_{k-1} + W_1\mathbf{x}_k,\ k > 2 \end{aligned} \tag{6.50}$$

where

$$W_i \triangleq \begin{bmatrix} f_{i1} & & & & \\ f_{i2} & f_{i1} & & & \\ f_{i3} & f_{i2} & f_{i1} & & \\ & \ddots & \ddots & \ddots & \\ & & f_{i3} & f_{i2} & f_{i1} \end{bmatrix}, i = 1, 2, 3. \tag{6.51}$$

Therefore the convolution (6.47) of a matrix X is equivalent to the following linear transform of a vector $\mathbf{x}$:

$$\mathbf{y} = W\mathbf{x} \tag{6.52}$$

$$\mathbf{x} \triangleq \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \vdots \\ \mathbf{y}_n \end{bmatrix} \quad W \triangleq \begin{bmatrix} W_1 & & & & \\ W_2 & W_1 & & & \\ W_3 & W_2 & W_1 & & \\ & \ddots & \ddots & \ddots & \\ & & W_3 & W_2 & W_1 \end{bmatrix} \quad \mathbf{y} \triangleq \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}. \tag{6.53}$$

Note that W_i, defined in (6.51), are lower triangular matrices and all the diagonals of W_1 are f_{11}. Thus, the equivalent linear transform (6.52) of the convolution (6.47) is a triangular linear transform and all the diagonals of W are f_{11}.

Batch normalization (BN) is another essential layer in residual blocks. BN can be described as

$$Y = sX + b \tag{6.54}$$

where X is the input, Y is the output, s is the scaling factor and b is the bias. If we transform the matrices Y and X into vectors $\mathbf{y}$ and $\mathbf{x}$ similarly as that in the discussion of convolution, we have $\mathbf{y} = s\mathbf{x} + b$ which is a special type of lower triangular transform with a diagonal transformation matrix, namely sI, where I is an identity matrix.

To summarise the discussions above, we have

Lemma 7 *Let $\mathbf{y}_k^T, \mathbf{x}_k^T$ denote the k^{th} row of Y and X respectively, and let $\mathbf{y}, \mathbf{x}, W$ be defined as in (6.53). Then*

i) The BN transform in (6.54) is equivalent to $\mathbf{y} = s\mathbf{x} + b$.

ii) The convolution in (6.47) is equivalent to $\mathbf{y} = W\mathbf{x}$ where W is a lower triangular matrix whose diagonals are all f_{11}.

Now consider the residual block in Figure 6-5 (b) where the BN layer and the CONV layer are described as in (6.54) and (6.47). From Lemma 7, the transformation of the residual block in Figure 6-5 (b) is equivalent to the residual block in Figure 6-4 with $W_1 = sI, \mathbf{b}_1 = \mathbf{b}$, $W_2 = W$ (where W is defined in 6.53), $\mathbf{b}_2 = \mathbf{0}$. From Theorem 8, it follows

Theorem 8 *Consider the residual block in Figure 6-5 (b), and let s and b be the scaling factor and bias of BN layer and f_{11} be the element in the top left corner of the convolution filter. If we use zero padding on the top and left sides to implement the convolution, the residual block in Figure 6-5 (b) is invertible for any output Y if and only if $sf_{11} > -1$.*

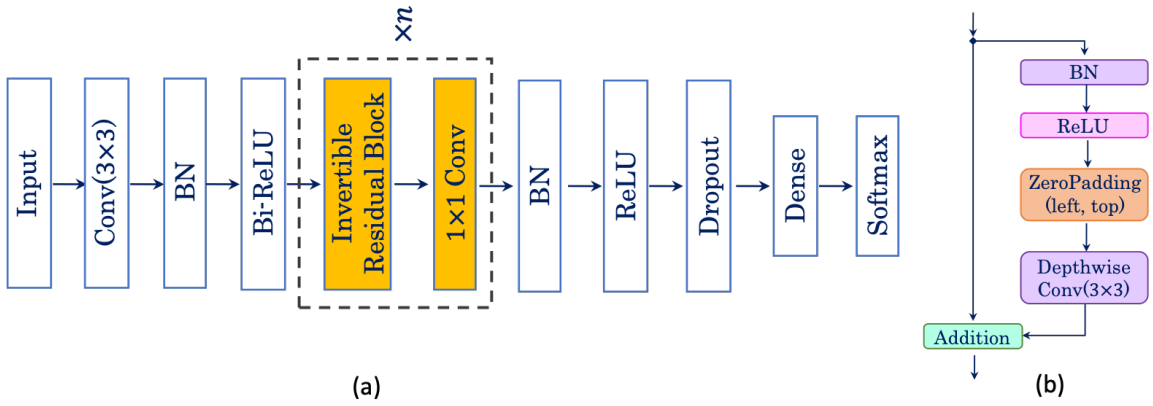

Figure 6-5: (a) A residual network; (b) A residual block with one ReLU layer.

6.2.3 Experimental Results

In this section, we first design a residual network for digit recognition and conduct experiments on MNIST [62] dataset to check the invertibility condition and verify the effectiveness of the inverse algorithm. We will also show how to initialize the filters to obtain invertible residual blocks. Next, we conduct experiments on a SHM dataset of a beam structure (presented in Section 2.5.1) to further demonstrate the discriminative ability of the proposed network for structural damage identification.

The Network Architecture: We construct an invertible residual network using the invertible residual blocks described in Section 6.2.2. The architecture of the proposed invertible residual network is illustrated in Figure 6-5. The first layer of the proposed network is a general convolution (CONV) layer, followed by a bidirectional ReLU (Bi-ReLU) layer, a BN layer, and a 1×1 convolution layer. The BN layer can be inverted directly by subtracting the bias and then multiplying the inverse of the scaling factor. Bi-ReLU and 1×1 Convolution are also invertible. The Bi-ReLU activation placed after the first convolution aims to preserve all the information of the input $\mathbf{x}$ by concatenating both the positive and the negative parts of $\mathbf{x}$. Thus, we can easily restore $\mathbf{x}$ from the output of the Bi-ReLU layer through $\mathbf{x} = \max(0, \mathbf{x}) - \max(0, -\mathbf{x})$. The 1×1 convolution is a linear transform which can be formulated as $\mathbf{y} = \mathbf{W}\mathbf{x}$ whose inverse function is simply $\mathbf{x} = \mathbf{W}^{-1}\mathbf{y}$ when W is nonsingular.

Following the Bi-ReLU activation layer, an invertible residual unit and a 1×1 convolution are repeatedly stacked n times where n is the required number of residual blocks. BN and

ReLU layers are placed after the last 1×1 convolution in the last residual block, followed by a dropout layer to avoid over-fitting. Finally, a dense layer with a lower dimension (*e.g.*, 500) is used before the output layer with softmax.

Figure 6-5(b) shows the residual block with a number of BN+ReLU+CONV in the residual path. In the CONV layer, depthwise convolution is applied where the spatial convolution is conducted channel-wise. That is, each channel has a separate filter. Depthwise convolution has been widely used in deep learning networks (*e.g.*, in MobileNet [44] and Xception [20]) to reduce the number of parameters. A 1×1 convolution is usually used after the depthwise convolution layer to mix the channels. In the proposed residual network architecture, we adopted this idea in a novel way. Instead of placing both depthwise convolution and 1×1 convolution in the residual unit, we place the depthwise convolution in the residual unit but the 1×1 convolution is placed out of the unit so that each channel of the residual block can be inverted separately. It should be noted that the network preserves all the information of input until the last BN layer if the invertibility conditions are satisfied for each residual block.

Experimental results on MNIST: The following experiments use the same settings of database partition as in Section 6.1.4. Three experiments are conducted with two residual blocks in the proposed residual network in Fig 6-5 where each block has one ReLU layer in the residual block. In the first experiment, we check whether the trained models with the Glorot uniform [33] initialization, a default initialization implemented in TensorFlow [1], are invertible.

In the second experiment, we check how to initialize the filters so that the trained models are invertible. As the invertibility requires $sf_{11} > -1$, the 3×3 filters are initialized with $f_{11} = 1$ and $f_{12} = f_{13} = 0$ so that the invertibility conditions are satisfied at the beginning. In the third experiment, the filters are initialized with $f_{11} = -2$ and $f_{12} = f_{13} = 0$ so that the invertibility conditions are not satisfied at the beginning. This experiment aims to check whether this initialization leads to a trained model which is not invertible and how such models performs for classification accuracy.

For each experiment, we train 30 classification models each trained with random initialization on the weights other than the convolution filters. For all the experiments, 60 epochs are conducted with the Adam optimizer. The learning rate is set to 0.001 and decayed by a factor of 0.5 once learning stagnates. The condition $sf_{11} > -1$ is checked for each residual

block in the classification models. For the first experiment, 10 out of the 30 trained models are invertible, which shows that the condition of invertibility is not very strict as nothing is enforced to constrain the filters. Moreover, the selected model with the best performance on the validation data is invertible. For the second experiment, all the trained models are invertible. This means the invertibility condition can be easily imposed by this simple initialization. Although this initialization cannot guarantee to result in an invertible residual network, intuitively, a bigger value of the initialisation of f_{11} increases the chance to train an invertible model. For the third experiment , none of the residual blocks are invertible. This experiment and the second experiment demonstrate that the invertibility of the trained neural network can be imposed by proper initialization on the convolution filters.

To check the classification performance for each initialization, we choose the model with the smallest validation error and get the test errors of 0.71%, 0.72%, and 0.76% for the Glorot Uniform initialization, the initialization with $f_{11} = 1$, and the initialization with $f_{11} = -2$, respectively.

To check the effectiveness of the proposed inverse algorithm, we inverse the classification models by recovering the input images from the last invertible layer (*i.e.*, the last BN layer). For each model, 100 samples per digit class are randomly selected and recovered from the last BN layer. For both models from experiment 1 and 2, all the 1000 images can be perfectly recovered by the proposed inverse algorithm.

Experimental results on an SHM dataset: To demonstrate the discriminative ability of the proposed residual network for structural damage identification, we conducted an experiment on a beam structure, which is described in Section 2.5.1. Acceleration responses generated via the finite element model of the beam structure are used as the input. While the stiffness reductions are used as the output. Details of the data generation and data preprocessing are described in Sections 5.4.1 and 5.4.2, respectively. Four datasets generated with four scenarios are used to evaluate the performance:

- Scenario 1: Baseline dataset without considering any measurement noise or modelling uncertainties.
- Scenario 2: Measurement noise dataset includes 30dB of white noise in the acceleration responses.
- Scenario 3: Uncertainty dataset includes 1% - 3% modelling uncertainties in the stiff-

ness parameters.

- Scenario 4: Measurement noise + Uncertainty dataset includes both modelling uncertainties (1% - 3%) and measurement noise (30dB).

27,182 samples are generated for the first two datasets, and 59,700 samples for the rest two datasets, respectively. The datasets are randomly split into three subsets, namely, 70% for training, 15% for validation, and another 15% for testing. The metrics, MSE and R-value, are used to report the performance of the invertible networks.

The test results are shown in Tables 6.1 and 6.2, respectively. A standard residual network (ResNet) is trained and compared with the proposed invertible ResNet. In terms of both MSE and R-value, the proposed invertible ResNet has achieved marginally improvements for the noise datasets in comparison with the standard ResNet. Moreover, it is observed that in general, the proposed invertible ResNet can provide accurate structural damage identifications. Thus, the proposed invertible ResNet preserves the discriminative ability for this regression task.

It should be noted that the proposed SDI-DenseNet in Chapter 5 performs the best for structural damage identification, even though the proposed invertible ResNet has marginally improved the results over the standard ResNet. Despite this, the proposed invertible ResNet may be used to generate adversarial examples that will help to increase the robustness and security of SHM frameworks. The invertible DenseNet will also be investigated for future studies.

Table 6.1: Performance evaluation on the Baseline dataset and the Measurement noise dataset.

Methods	**Metrics**	**Baseline**	**Measurement Noise**
ResNet	MSE	9.6×10^{-6}	1.9×10^{-4}
	R-value	0.999	0.986
Invertible ResNet	MSE	1.9×10^{-5}	1.5×10^{-4}
	R-value	0.999	0.989

6.2.4 Discussion on adversarial attacks via invertible neural networks

Deep neural networks have been discovered to be vulnerable to adversarial examples, *i.e.*, fabricated samples that are often barely recognizable by humans but can fool deep neural

Table 6.2: Performance evaluation on the Uncertainty dataset and the Measurement noise + Uncertainty dataset.

Methods	**Metrics**	**Uncertainty**	**Measurement Noise + Uncertainty**
ResNet	MSE	4.9×10^{-5}	3.2×10^{-4}
	R-value	0.996	0.968
Invertible ResNet	MSE	3.9×10^{-5}	2.1×10^{-4}
	R-value	0.997	0.979

networks easily. For instance, by introducing small perturbations to an original clean image, one can generate an adversarial image to misguide the image classifier and make an incorrect label. It is also known as adversarial attacks.

This vulnerability to adversarial attacks can introduce real-world threats to systems relying on deep neural networks, such as deep learning-based SHM systems. In the aspect of adversarial falsification [108], adversarial attacks can be categorized into false-positive attacks and false-negative attacks. For a structural damage identification task, a false-positive example can be defined as the false labelling of the undamaged state as damaged, whereas the false labelling of the damaged state as undamaged is a false-negative example [15]. Falsely labeling damaged structures as healthy poses a serious concern. This type of attack may lead to critical failure or worsening of structural health if it is left undetected.

Many research showed that adversarial training could improve the robustness of deep neural networks by injecting adversarial examples in the training stage [35, 46, 105]. [8] recently explored to achieved a deep understanding of adversarial examples through Invertible Residual Networks (i-ResNets) [12]. For the majority of classifiers that map high-dimensional input images to low-dimensional outputs, *i.e.*, the classification logits, most information is discarded during this mapping. However, their bijective network preserves all the classification irrelevant information in the latent representation, *i.e.*, non-logits. Though the goal of adversarial examples attacks is only to modify the logits, their results suggest that the current attack methods indeed also modify non-logits. In this regard, they propose a novel approach to generate high-quality adversarial examples using the bijective network by first modifying the latent representation and then using the invertibility property to obtain adversarial images in the input space. This research opens up the door for effective adversarial training via invertible neural networks. The proposed invertible residual network has the

same bijective mapping property as i-ResNets but presents weaker conditions under which a residual block is invertible. Therefore, the proposed invertible residual network can also be used for adversarial example generation and training, which in turn can improve the robustness and performance of deep learning-based SHM frameworks.

6.3 Summary

The invertibility of the residual networks with rectifier as activation functions has been investigated and conditions are presented for the residual units to be invertible. The conditions are addressed for general rectifier linear units and can be used to analyse the invertibility of other deep rectifier networks. A new fixed-point algorithm is also proposed to invert hidden layers of residual networks, and a simple residual neural network is designed in our experiments to validate the invertibility. The experimental results demonstrate that the proposed fixed-point inverse algorithms can be used to invert more general residual units which cannot be inverted by the existing fixed-point algorithm presented in i-ResNets [12]. After that, tight conditions have been presented for a residual block to be invertible. Compared with the Lipschitz-constant constraints imposed in i-ResNets, the presented condition is much weaker and easier to check. Experimental results demonstrate that the required conditions can be easily imposed through proper initialization on the convolution filters. In addition, experimental results show that the proposed invertible residual network preserves the discriminative ability which performs well for both image classification and structural damage identification. Moreover, recent studies show that invertible neural networks such as i-ResNets and AdvFlow [26] have been explored to generate adversarial examples. Thus, it is promising to apply the proposed invertible residual network to investigate adversarial attacks or generate adversarial examples, thereby improving the robustness and security of the deep learning-based SHM frameworks.

Chapter 7

Conclusion and Future Works

7.1 Conclusions

In this thesis, we have developed advanced deep learning models for vibration-based structural damage identification. Vibration signals such as frequencies, mode shapes, and acceleration responses are used as the input while the structural properties such as stiffness parameters are the output. The underlying relationship between the input and output are learned to perform efficient and effective damage identification. However, such vibration signals are often high-dimensional and noise-contaminated, and sometimes in multiple scales or have multiple physical meanings. Thus, we propose comprehensive studies on this specific application to overcome the above-mentioned obstacles. Further, we investigate the invertible conditions on residual neural networks. The invertibility property enables us to have a better understanding on the deep representations in the hidden layers and can be potentially used to generate adversarial examples to improve the robustness and security of deep learning-based SHM frameworks.

Below we summarise the contributions of this thesis along with each objective stated in Chapter 1, respectively.

- In Chapter 3, we proposed a parallel sparse autoencoder framework (Para-AF) that is feasible to deal with multi-scale data or data of multiple physical meanings. This framework consists of two main components: a parallel architecture-based dimensionality reduction component followed by a relationship learning component. The 1st component consists of multiple sparse autoencoders that performs dimensionality re-

duction and feature learning on the frequencies and mode shapes separately. The learnt features from the frequencies and mode shapes are then concatenated as one vector, which is used as the input to the relationship learning component. This component is a deep autoencoder with multiple hidden layers that captures the relationship between the learnt features and the structural stiffness parameters. Experiments are conducted on both the clean dataset and noise dataset to evaluate the performance of the proposed method. Also, the proposed Para-AF is compared with the state-of-the-art approach, SAF, which is a sequential model processing the frequencies and mode shapes together. The experimental results show that the proposed Para-AF provides a more reasonable process to normalize and extract features separately from frequencies and mode shapes, which significantly improves the accuracy and robustness of structural damage identification.

- In Chapter 4, we exploit a convolutional neural network (CNN) based method, that is, deep residual neural network (ResNet), to build a more efficient and effective structural health monitoring framework. Compared to the autoencoder-based model presented in Chapter 3, the proposed ResNet framework performs feature extraction and damage identification in the end-to-end learning process, where pre-training and two-stage learning strategies are no longer needed anymore. The framework makes use of the advantages of CNN such as sparse interactions, parameter sharing, and equivariant representations, which can be used for efficient feature learning from high-dimensional data. Moreover, skip connections adopted in this framework solves the gradient vanishing/exploding issues in very deep architectures. Comprehensive performance evaluation including numerical studies and experimental verifications are conducted to show the effectiveness of this framework for damage identification. Evidently, the proposed method shows high accuracy and good robustness, even when the data contains both modelling uncertainties and measurement noise. Moreover, significant improvements of this framework over the SAF and the well-known CNN methods are observed in experiments.

- In Chapter 5, we present a novel structural damage identification framework termed as SDI-DenseNet. Dense connectivity and feature concatenation are adopted in this framework. It also mitigates the gradient vanishing problem via dense connections,

while further strengthens feature propagation and information flow. Diverse levels of features are preserved and reused during training, which fits well to tasks using time-domain vibration response data, *e.g.*, acceleration responses. A large number of sensors are required to obtain the mode shapes used in Chapters 3 and 4, whereas a reduced number of sensors are needed to collect the acceleration responses explored in this study. Thus, it is of great practical value to conduct damage identification by using the time-domain vibration responses. A simply supported beam structure is investigated in the numerical study for structural damage identification considering both modelling uncertainties and measurement noise. A seven-storey steel frame structure tested in the laboratory is employed to evaluate the performance of the proposed SDI-DenseNet, in comparison with the CNN and ResNet. Both the numerical and experimental studies demonstrate the superior performance of the proposed SDI-DenseNet for identifying structural damage locations and quantifying damage severities. It is observed SDI-DenseNet outperforms other deep learning frameworks proposed in this thesis.

- In Chapter 6, an novel fixed-point algorithm is first proposed to invert hidden layers of residual networks, and a simple ResNet is designed to validate the invertibility without sacrificing the discriminative performance. Next, tight conditions for a residual block to be invertible are presented. Experimental results demonstrate that the required conditions can be easily imposed through proper initialization on the convolution filters. A new invertible ResNet is constructed and shown to perform well on both image classification and structural damage identification. It should be noted that the proposed tights conditions are muck weaker and easier to check compares to the Lipschitz-constant constraints imposed in i-ResNets. The proposed invertible ResNet has potential to be applied to increase the robustness and security of the deep learning SHM framework against adversarial attacks.

7.2 Future works

Despite the satisfactory performances achieved by all of the proposed approaches, there are still possible extensions and unaddressed problems that can be studied for future works:

- As presented in Chapter 3, the proposed Para-AF framework is feasible to deal with multi-scale data. Currently, we apply the framework to an SHM application, *i.e.*, struc-

tural damage identification. This framework can also be applied to other applications (not limited to SHM applications) where the input features are also in multiple scales or have multiple categories. Also, sparse autoencoders are used as the basic building blocks in this framework, in which the hidden layers are dense layers. It would be interesting to explore other types of autoencoders, such as convolutional autoencoders in the framework.

- The proposed deep ResNet framework in Chapter 4 performs a comprehensive study on the structural damage identification problem using modal information of structures. Compared to the previous study proposed in Chapter 3, this deep ResNet framework more efficient and robust to noise effects. To measure mode shapes used in this chapter, however, it is usually necessary to have a large number of sensors covering the entire structure, which is high-cost and impractical for large-scale structures. Future studies on exploring other types of vibration signals that require less sensors, *e.g.*, time-domain responses will be investigated.

- The proposed SDI-DenseNet framework in Chapter 5 exploits another type of vibration signal, *i.e.*, time-domain acceleration response for structural damage identification. This framework is superior to all the other methods proposed in this thesis. However, all these studies are limited to numerical modelling and laboratory test. Future studies on applying the proposed advanced deep learning based approaches for large-scale structures in the wild will be conducted. Besides, an invertible ResNet is proposed in this thesis and it marginally outperforms the standard ResNet for structural damage identification. The investigation of invertible DenseNets will also be considered as our future goal.

- The novel invertible ResNet proposed in Chapter 6 shows good discriminative performance on SHM datasets, which is competitive with the performance of the non-invertible ResNet. Though the proposed invertible ResNet indeed exhibits effective invertibility, the benefits of this property need to be further exploited for SHM applications. In real-world applications, if deep learning SHM frameworks are to be adopted, security is of the utmost importance. Some recent works discuss the invertibility may help people to have a better understanding of the distribution of adversarial

examples and improve the robustness of classifiers. In this case, we can apply the invertible ResNet further to improve the robustness and security of the SHM frameworks against adversarial attacks.

- The structural damage identification frameworks proposed in this thesis are basically deterministic point prediction models. For the SHM applications, Bayesian neural networks have great potentials because they account for uncertainties in model parameters (*i.e.*, weights and bias) and propagate this into the predictions. Uncertainty management is critical to support the decision-making in SHM applications. We consider applying the Bayesian neural networks for structural damage identification as one of our future research goals.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *Journal of Sound and Vibration*, 388:154–170, 2017.

[3] A. C. Altunışık, F. Y. Okur, and V. Kahya. Modal parameter identification and vibration based damage detection of a multiple cracked cantilever beam. *Engineering Failure Analysis*, 79:154–170, 2017.

[4] A. C. Altunışık, F. Y. Okur, S. Karaca, and V. Kahya. Vibration-based damage detection in beam structures with multiple cracks: modal curvature vs. modal flexibility methods. *Nondestructive Testing and Evaluation*, 34(1):33–53, 2019.

[5] S. An, F. Boussaid, and M. Bennamoun. How can deep rectifier networks achieve linear separability and preserve distances? In *International Conference on Machine Learning*, pages 514–523, 2015.

[6] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

[7] M. Azimi and G. Pekcan. Structural health monitoring using extremely compressed data through deep learning. *Computer-Aided Civil and Infrastructure Engineering*, 35(6):597–614, 2020.

[8] R. Bai, S. Bagchi, and D. I. Inouye. Exploring adversarial examples via invertible neural networks. *arXiv preprint arXiv:2012.13111*, 2020.

[9] N. Bakhary, H. Hao, and A. J. Deeks. Damage detection using artificial neural network with consideration of uncertainties. *Engineering Structures*, 29(11):2806–2815, 2007.

[10] Y. Bao and H. Li. Machine learning paradigm for structural health monitoring. *Structural Health Monitoring*, page 1475921720972416, 2020.

[11] W. Bayissa, N. Haritos, and S. Thelandersson. Vibration-based structural damage identification using wavelet transform. *Mechanical systems and signal processing*, 22(5):1194–1215, 2008.

[12] J. Behrmann, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks. *arXiv preprint arXiv:1811.00995*, 2018.

[13] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[14] V. Cabannes, F. Bach, and A. Rudi. Disambiguation of weak supervision with exponential convergence rates. *arXiv preprint arXiv:2102.02789*, 2021.

[15] M. D. Champneys, A. Green, J. Morales, M. Silva, and D. Mascarenas. On the vulnerability of data-driven structural health monitoring models to adversarial attack. *Structural Health Monitoring*, page 1475921720920233, 2020.

[16] C.-M. Chang, T.-K. Lin, and C.-W. Chang. Applications of neural network models for structural health monitoring based on derived modal properties. *Measurement*, 129:457–470, 2018.

[17] F.-K. Chang, J. F. Markmiller, J. Yang, and Y. Kim. Structural health monitoring. *System health management: with aerospace applications*, pages 419–428, 2011.

[18] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4545–4554, 2016.

[19] T. Q. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pages 9913–9923, 2019.

[20] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[21] R. W. Cottle, J.-S. Pang, and R. E. Stone. *The linear complementarity problem*. SIAM, 2009.

[22] G. Deboeck and T. Kohonen. *Visual explorations in finance: with self-organizing maps*. Springer Science & Business Media, 2013.

[23] Z. Ding, J. Li, and H. Hao. Structural damage identification using improved jaya algorithm based on sparse regularization and bayesian inference. *Mechanical Systems and Signal Processing*, 132:211–231, 2019.

[24] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[25] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[26] H. M. Dolatabadi, S. Erfani, and C. Leckie. Advflow: Inconspicuous black-box adversarial attacks using normalizing flows. *arXiv preprint arXiv:2007.07435*, 2020.

[27] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016.

[28] W. Fan and P. Qiao. Vibration-based damage identification methods: a review and comparative study. *Structural health monitoring*, 10(1):83–111, 2011.

[29] C. R. Farrar, S. W. Doebling, and D. A. Nix. Vibration–based structural damage identification. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 359(1778):131–149, 2001.

[30] C. R. Farrar and K. Worden. An introduction to structural health monitoring. *New Trends in Vibration Based Structural Health Monitoring*, pages 1–17, 2010.

[31] K. Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[32] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE international conference on computer vision*, pages 1134–1142, 2015.

[33] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[34] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[35] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[36] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

[37] T. Guo, L. Wu, C. Wang, and Z. Xu. Damage detection in a novel deep-learning framework: a robust method for feature extraction. *Structural Health Monitoring*, 19(2):424–442, 2020.

[38] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[39] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[40] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[41] G. E. Hinton, T. J. Sejnowski, et al. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[42] J. J. Hopfield. Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5):3–10, 1988.

[43] R. Hou, Y. Xia, and X. Zhou. Structural damage detection based on l1 regularization using natural frequencies and mode shapes. *Structural Control and Health Monitoring*, 25(3):e2107, 2018.

[44] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[45] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[46] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015.

[47] O. Huth, G. Feltrin, J. Maeck, N. Kilic, and M. Motavalli. Damage identification using modal data: Experiences on a prestressed concrete bridge. *Journal of Structural Engineering*, 131(12):1898–1910, 2005.

[48] A. Ilyas, L. Engstrom, and A. Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018.

[49] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[50] 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*, 72:303–315, 2016.

[51] L. Jiang, X. Ma, S. Chen, J. Bailey, and Y.-G. Jiang. Black-box adversarial attacks on video recognition models. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 864–872, 2019.

[52] J. M. Joyce. Kullback-leibler divergence., 2011.

[53] B. Karlik and A. V. Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[54] H. Khodabandehlou, G. Pekcan, and M. S. Fadali. Vibration-based structural condition assessment using convolution neural networks. *Structural Control and Health Monitoring*, 26(2):e2308, 2019.

[55] J.-T. Kim, Y.-S. Ryu, H.-M. Cho, and N. Stubbs. Damage identification in beam-type structures: frequency-based method vs mode-shape-based method. *Engineering structures*, 25(1):57–67, 2003.

[56] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[57] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[58] X. Kong, C.-S. Cai, and J. Hu. The state-of-the-art on framework of vibration-based structural damage identification for decision making. *Applied Sciences*, 7(5):497, 2017.

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[60] J. Kruse, L. Ardizzone, C. Rother, and U. Köthe. Benchmarking invertible architectures on inverse problems. *arXiv preprint arXiv:2101.10763*, 2021.

[61] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[62] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[63] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808. Citeseer, 2007.

[64] J. Li and H. Hao. A review of recent research advances on structural health monitoring in western australia. *Structural Monitoring and Maintenance*, 3(1):33, 2016.

[65] J. Li, S. Law, and Y. Ding. Substructure damage identification based on response reconstruction in frequency domain and model updating. *Engineering Structures*, 41:270–284, 2012.

[66] J. Li, S. Law, and H. Hao. Improved damage identification in bridge structures subject to moving loads: numerical and experimental studies. *International Journal of Mechanical Sciences*, 74:99–111, 2013.

[67] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *International Conference on Machine Learning*, pages 3866–3876. PMLR, 2019.

[68] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi. Deep text classification can be fooled. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul 2018.

[69] S. Lin, J. N. Yang, and L. Zhou. Damage identification of a benchmark building for structural health monitoring. *Smart materials and structures*, 14(3):S162, 2005.

[70] Y.-z. Lin, Z.-h. Nie, and H.-w. Ma. Structural damage detection with automatic feature-extraction through deep learning. *Computer-Aided Civil and Infrastructure Engineering*, 32(12):1025–1046, 2017.

[71] K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

[72] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[73] R. Mathias and J.-S. Pang. Error bounds for the linear complementarity problem with a p-matrix. *Linear Algebra and Its Applications*, 132:123–136, 1990.

[74] T. M. Mitchell et al. Machine learning. 1997.

[75] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[76] D. Montalvao, N. M. M. Maia, and A. M. R. Ribeiro. A review of vibration-based structural health monitoring with special emphasis on composite materials. *Shock and vibration digest*, 38(4):295–324, 2006.

[77] J. Mottershead, M. Friswell, G. Ng, and J. Brandon. Geometric parameters for finite element model updating of joints and constraints. *Mechanical systems and signal processing*, 10(2):171–182, 1996.

[78] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[79] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Hybrid models with deep and invertible features. *arXiv preprint arXiv:1902.02767*, 2019.

[80] R. M. Neal. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113, 1992.

[81] A. Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[82] W. S. Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[83] G. Oliveira, F. Magalhães, Á. Cunha, and E. Caetano. Vibration-based damage detection in a wind turbine using 1 year of data. *Structural Control and Health Monitoring*, 25(11):e2238, 2018.

[84] B. A. Olshausen and D. J. Field. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004.

[85] C. S. N. Pathirage, J. Li, L. Li, H. Hao, W. Liu, and P. Ni. Structural damage identification based on autoencoder neural networks and deep learning. *Engineering structures*, 172:13–28, 2018.

[86] C. S. N. Pathirage, J. Li, L. Li, H. Hao, W. Liu, and R. Wang. Development and application of a deep learning–based sparse autoencoder framework for structural damage identification. *Structural Health Monitoring*, 18(1):103–122, 2019.

[87] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[88] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[89] M. Ranzato, C. Poultney, S. Chopra, Y. LeCun, et al. Efficient learning of sparse representations with an energy-based model. *Advances in neural information processing systems*, 19:1137, 2007.

[90] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[91] A. Saberi, A. A. Stoorvogel, and P. Sannuti. Inverse filtering and deconvolution. *International journal of robust and nonlinear control*, 11(2):131–156, 2001.

[92] H. Schneider and G. P. Barker. *Matrices and linear algebra*. Courier Corporation, 2012.

[93] S. Shanmuganathan. Artificial neural network modelling: An introduction. In *Artificial neural network modelling*, pages 1–14. Springer, 2016.

[94] R. Shwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[95] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[96] Y. Song, C. Meng, and S. Ermon. Mintnet: Building invertible neural networks with masked convolutions. In *Advances in Neural Information Processing Systems*, pages 11002–11012, 2019.

[97] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[98] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[99] D. Takeuchi, K. Yatabe, Y. Koizumi, Y. Oikawa, and N. Harada. Invertible dnn-based nonlinear time-frequency transform for speech enhancement. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6644–6648. IEEE, 2020.

[100] J. Turian, J. Bergstra, and Y. Bengio. Quadratic features and deep architectures for chunking. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 245–248, 2009.

[101] R. Wang, Chencho, S. An, J. Li, L. Li, H. Hao, and W. Liu. Deep residual network framework for structural health monitoring. *Structural Health Monitoring*, page 1475921720918378, 2020.

[102] R. Wang, L. Li, and J. Li. A novel parallel auto-encoder framework for multi-scale data in civil structural health monitoring. *Algorithms*, 11(8):112, 2018.

[103] J. Whang, Q. Lei, and A. G. Dimakis. Compressed sensing with invertible generative models and dependent noise. *arXiv preprint arXiv:2003.08089*, 2020.

[104] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[105] Y. Wu, D. Bamman, and S. Russell. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1783, 2017.

[106] Y. Xia and H. Hao. Measurement selection for vibration-based structural damage identification. *Journal of Sound and Vibration*, 236(1):89–104, 2000.

[107] Y. Yan, L. Cheng, Z. Wu, and L. Yam. Development in vibration-based structural damage detection technique. *Mechanical systems and signal processing*, 21(5):2198–2211, 2007.

[108] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[109] C.-B. Yun, J.-H. Yi, and E. Y. Bahng. Joint damage assessment of framed structures using a neural networks technique. *Engineering structures*, 23(5):425–435, 2001.

[110] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[111] T. Zhang, S. Biswal, and Y. Wang. Shmnet: Condition assessment of bolted connection with beyond human-level performance. *Structural Health Monitoring*, 19(4):1188–1201, 2020.

[112] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[113] J. Zhuang, N. C. Dvornek, X. Li, P. Ventola, and J. S. Duncan. Invertible network for classification and biomarker selection for asd. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 700–708. Springer, 2019.

[114] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.

Appendix A

Attribution Statement

Chapters 3 to 6 of this thesis are based on works that have been published with joint-authorship. We hereby make an authorship attribution statement to clarify the contribution of individual authors.

Chapter 3 is based on the publication:

- **Wang, R.**, Li, L., & Li, J. (2018). A novel parallel auto-encoder framework for multi-scale data in civil structural health monitoring. *Algorithms*, 11(8), p.112.

	Conception and Design	**Data Acquisition and Manipulation**	**Programming**	**Experiments**	**Interpretation and Discussion**	**Manuscript Writing and Revision**	**Total Contribution**
Co-author 1 **(Ruhua Wang)**	✓	✓	✓	✓	✓	✓	60%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Ling Li)	✓	×	×	×	✓	✓	20%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Jun Li)	✓	✓	×	×	✓	✓	20%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							

Chapter 4 is based on the publication:

- **Wang, R.**, Chencho, An, S., Li, J., Li, L., Hao, H., & Liu, W. (2020). Deep residual network framework for structural health monitoring. *Structural Health Monitoring*, p.1475921720918378.

	Conception and Design	**Data Acquisition and Manipulation**	**Programming**	**Experiments**	**Interpretation and Discussion**	**Manuscript Writing and Revision**	**Total Contribution**
Co-author 1 **(Ruhua Wang)**	✓	✓	✓	✓	✓	✓	50%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Chencho)	✓	✓	×	×	✓	✓	10%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Senjian An)	✓	×	×	×	✓	✓	10%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 4 (Jun Li)	✓	×	×	×	✓	✓	10%
Co-author 4 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 5 (Ling Li)	✓	×	×	×	✓	✓	10%
Co-author 5 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 6 (Hong Hao)	✓	×	×	×	✓	✓	5%
Co-author 6 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 7 (Wanquan Liu)	✓	×	×	×	✓	✓	5%
Co-author 7 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							

Chapter 5 is based on the publication:

- **Wang, R.**, Li, J., An, S., Hao, H., Liu, W., & Li, L. (2021). Densely connected convolutional networks for vibration based structural damage identification. *Engineering Structures*, 245, 112871.

	Conception and Design	**Data Acquisition and Manipulation**	**Programming**	**Experiments**	**Interpretation and Discussion**	**Manuscript Writing and Revision**	**Total Contribution**
Co-author 1 **(Ruhua Wang)**	✓	✓	✓	✓	✓	✓	50%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Jun Li)	✓	✓	×	×	✓	✓	10%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Senjian An)	✓	×	×	×	✓	✓	10%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 4 (Hong Hao)	✓	×	×	×	✓	✓	10%
Co-author 4 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 5 (Wanquan Liu)	✓	×	×	×	✓	✓	10%
Co-author 5 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 6 (Ling Li)	✓	×	×	×	✓	✓	10%
Co-author 6 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							

Chapter 6 is based partly on the publication:

- **Wang, R.**, An, S., Liu, W., & Li, L. (2021). Fixed-point algorithms for inverse of residual rectifier neural networks. *Mathematical Foundations of Computing*, 4(1), p.31.

	Conception and Design	**Data Acquisition and Manipulation**	**Programming**	**Experiments**	**Interpretation and Discussion**	**Manuscript Writing and Revision**	**Total Contribution**
Co-author 1 **(Ruhua Wang)**	✓	✓	✓	✓	✓	✓	70%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Senjian An)	✓	×	×	×	✓	✓	10%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Wanquan Liu)	✓	✓	×	×	✓	✓	10%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 4 (Ling Li)	✓	✓	×	×	✓	✓	10%
Co-author 4 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							