Title
Machine Learning Based Seismic Structural Health Monitoring and Reconnaissance

Author
Li, Chenglong

Publication Date
2022

Peer reviewed|Thesis/dissertation

Machine Learning Based Seismic Structural Health Monitoring and Reconnaissance

by

Chenglong Li

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Khalid M. Mosalam, Chair
Professor Shaofan Li
Professor Adityanand Guntuboyina

Spring 2022

Machine Learning Based Seismic Structural Health Monitoring and Reconnaissance

Abstract

Machine Learning Based Seismic Structural Health Monitoring and Reconnaissance

by

Chenglong Li

Doctor of Philosophy in Engineering - Civil and Environmental Engineering

University of California, Berkeley

Professor Khalid M. Mosalam, Chair

Civil structures, including bridges, tunnels, and skyscrapers, are becoming susceptible to losing their intended functionality as they deteriorate through the service life. Furthermore, this situation is exemplified in the face of natural hazards and extreme events. Therefore, monitoring and rapid reconnaissance of the condition and health states of such structures is important for effective decision-making towards building more resilient infrastructure systems. Traditionally, such monitoring and reconnaissance efforts require onsite human inspection. However, given the growth of buildings and other infrastructure in urban centers, such an inspection process is infeasible because of limited human resources, financial burden, and time consuming efforts.

In this dissertation, methods to automate the monitoring and reconnaissance processes are proposed. First, methods are introduced to automate the data collection process, where information, that are highly relevant to the health states of structures as well as the entire infrastructure systems, are collected. Second, algorithms are introduced to automate the data processing, where results regarding the health states of structures and infrastructure systems are obtained. Such results are essential for the decision-making process to increase resiliency of the infrastructure systems. The most important technique to automate the above processes is Artificial Intelligence (AI), in particular, Machine Learning (ML) algorithms.

In the case of a single structure, the process of observing its response and determining its health state is called Structural Health Monitoring (SHM). A novel SHM framework utilizing Deep Learning (DL) is proposed. The framework is based on Long Short-Term Memory (LSTM) Encoder-Decoder architecture, which is a variant of the Recurrent Neural Network (RNN), applied to Time Series (TS) data. The TS data is processed through the LSTM network, where the information in the TS data is condensed into a *Latent Space Vector* (LSV), which is processed through traditional ML algorithms to output the structural health conditions, including the overall health conditions, the locations and severity of damage. To enforce the encoding (i.e., condensation) process of the TS into the LSV without information

loss, an Encoder-Decoder architecture is proposed. Moreover, a method for fast prediction of the structural responses, which uses variants of the LSTM network, as well as a novel network called Temporal Convolutional Network (TCN), is proposed, and these models (variants) are compared against each other in terms of the accuracy of predicting the structural response. The proposed models are anticipated to complement/replace the traditional physical simulations for faster prediction of the structural response when immediate results are required, e.g., for rapid decision-making.

On the data collection side of a single structure, the quality of data obtained from the sensor network is critical to the diagnosis (i.e., determination of the health conditions of the structure). If the sensors are not placed on locations that are sensitive enough to the structural damage, the collected data is not useful for the purpose of diagnosis. In this dissertation, an Optimal Sensor Placement (OSP) method is proposed. The *causal relationship* among the sensor recordings is identified and quantified through Directed Information (DI). In this method, the sensors are added sequentially, i.e., one sensor at a time, until the specified number of sensors (typically based on expert opinion and availability of resources) is satisfied. The new sensor is added at a location where the causal relationship with the existing sensors is the lowest to ensure low redundancy of the information stored in the array of sensors.

For the case of infrastructure on a regional (e.g., city) scale, a method to effectively collect reconnaissance results following an earthquake event is proposed. Social media posts by people near the source of the earthquake, news reports, as well as information from official resources, e.g., United States Geological Survey (USGS), are collected automatically following the earthquake event. Such information is subsequently summarized as a briefing, which provides valuable reference for further detailed reconnaissance (field investigation) and emergency response. The Natural Language Processing (NLP) method is adopted in the formulation of these briefings. Moreover, a practical method to quantify the regional recovery state (a step towards quantifying a metric for the resilience of the affected community) following the earthquake event is proposed. This is based on the number of relevant posts collected from the social media. The recovery is quantified as the averaged recovery states of several key aspects, e.g., water supply to the community, electricity supply to the community, and availability/resumption of the functionality of essential facilities, e.g., medical services by hospitals.

Contents

List of Figures

List of Tables

Listings

Acknowledgments

The author would like to thank Prof. Khalid M. Mosalam, who is the author's advisor, and acknowledge the financial support from the Pacific Earthquake Engineering Research (PEER) Center and the Taisei Chair of Civil Engineering at the University of California, Berkeley. The author would also like to thank Prof. Aditya Guntuboyina, who served as members of the author's Ph.D. qualifying examination and dissertation committees, and Prof. Shaofan Li, who served as a member of the author's Ph.D. dissertation committee. Thanks are also due to Prof. Matthew DeJong, Prof. Jack Moehle, and Prof. Laurent El Ghaoui, who served as members of the author's Ph.D. qualifying examination committee.

Moreover, the author would like to express his gratitude to Ms. Ang Li, Ms. Alicia Yi-Ting Tsai, Mr. Jiajian Lu, Mr. Minjune Hwang, Mr. Vedant Mathur, and Dr. Jing Lian. Thanks are also due to the members of Prof. Mosalam's research group, namely the STructural Artificial Intelligence Research (STAIR) lab, `http://stairlab.berkeley.edu/`, including but not limited to Dr. Yuqing Gao, Dr. Sifat Muin, Ms. Chrystal Chern, Dr. Jorge Archbold, and Mr. Pengyuan Zhai, and the members of the PEER center, including Dr. Selim Günay, Dr. Amarnath Kasalanati, and Mr. Gabriel Vargas, for their research collaboration (published and unpublished), technical assistance, and valuable companionship throughout this memorable journey.

Chapter 1

Introduction

1.1 Background & Motivation

The global population growth of the 20$^{\text{th}}$ century and its continuation into the current 21$^{\text{st}}$ century has resulted in the increase of infrastructure systems and buildings on a massive scale in urban centers. Such civil structures, including bridges, tunnels, and skyscrapers, are becoming susceptible to losing their designed functionality as they deteriorate through the service life. The latest report card for America's infrastructure issued by the American Society of Civil Engineers (ASCE) in 2021 gave a "$C-$" grade [2]. The report estimated that the cost associated with the reparation and improvement is over $4.59 Trillion. The National Academy of Engineering (NAE) of the United States has identified "restore and improve urban infrastructure" as the one of the grand engineering challenges of the 21$^{\text{st}}$ century [83]. This situation is further exemplified in the face of natural hazards. Being aware of the above challenges, efforts should be made towards more resilient communities and their infrastructure systems and buildings.

The deterioration of the health state of the infrastructure and buildings could be attributed to two sources: (1) the gradual deterioration due to aging, and (2) the sudden damage due to the effect of natural hazards, e.g., earthquakes, tsunamis, fires, hurricanes, or tornadoes. One of the most destructive natural hazards is earthquakes, where during major events, the structural components may experience slight to severe damage. Fig. 1.1 shows a schematic of the change of quality (represented quantitatively as a percentage of the best quality or normal operation) of the infrastructure due to the occurrence of a major earthquake at time t_0, which causes a sudden drop in functionality [12]. At time t_1, the infrastructure recovers to full functionality. A slightly modified equation, which is based on the equation originally proposed by Bruneau et al. [12], for quantifying the resiliency (γ_φ) of a particular system φ, such as a hospital, is expressed as follows [81]:

$$\gamma_\varphi = \frac{\int_{t_0}^{t^*} Q(t)dt}{t^* - t_0}, \tag{1.1}$$

where, t^* is the maximum acceptable recovery time where $t_1 \leqslant t^*$. Therefore, reducing the

recovery time, which is the time needed after the extreme event to restore the functionality (i.e., $t_1 - t_0$), is important to improve the resiliency. In that process, fast *monitoring* and *reconnaissance* of the conditions of infrastructure systems and buildings is needed to provide important information regarding the health state of the structure, e.g., appropriateness for immediate occupancy, functional recovery, and the need for reparation or even reconstruction. Such process is essential for the decision-making to reduce the recovery time.

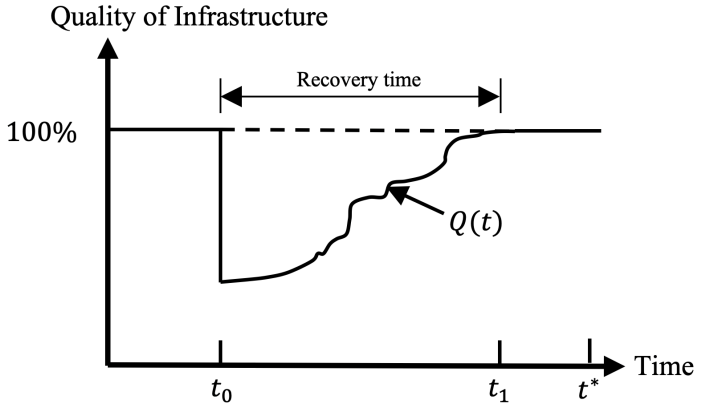


Figure 1.1: Change of the quality of infrastructure with time.

Traditionally, Structural Health Monitoring (SHM) and reconnaissance efforts require onsite human inspection. However, such traditional inspections are not conducted on a regular basis because of limited human resources, financial burden, and time consuming efforts. Given the vast amount of infrastructure systems and buildings, the total amount of needed labor and resources would make this process infeasible to be conducted on a regional scale. Moreover, in some cases, it is difficult to make onsite inspections where some damage states are hard to notice, or it is unsafe and risky to enter the building and make the necessary detailed inspection [26].

There are two major concerns related to human inspection of infrastructure systems and buildings: (1) The information collection process is cumbersome where the efficiency of data collection by humans is low, and (2) The judgement of the structural conditions on a regional scale, based on the data collected by a human, is slow, where the human needs to analyze the current information, and make judgements case by case. Such problems need to be solved in order to expedite the decision-making process. The main objective of the study presented in this dissertation is to **improve the automatic data collection and processing** for the monitoring and reconnaissance of infrastructure systems and buildings. This automation should have the following properties: (1) none or minimal human work is required, (2) fast monitoring, which is especially important after a major earthquake for rapid assessment,

and (3) high reliability, with equal or higher accuracy than human inspection. Concretely, this dissertation attempts to improve resiliency from the following two major aspects:

1. Identify methods to (partially) replace the human work on inspection through automatic data collection techniques; and
2. Provide algorithms to process the data collected automatically and output health conditions of infrastructure systems and buildings.

The main method that this study is founded on is Artificial Intelligence (AI) and, in particular, the Machine Learning (ML) algorithms. The AI technique is a collection of methods for the machine to learn and mimic the behavior of a human. Given the amount of computational power and storage capacity of computers, such AI techniques could collect and process information much faster than human beings. Therefore, the implementation of such techniques could reliably replace the labor-intensive tasks of SHM and reconnaissance, as discussed in this dissertation.

1.2 Overview of the Study

The presented study in this dissertation is divided into two parts: (1) Data collection; and (2) Data processing. The main contributions of these two parts are summarized separately in the following two sub-sections.

1.2.1 Data Collection

First, consider the case of a single structure. With the development of sensor technology, structural responses, e.g., displacement, acceleration, and strain, can be accurately recorded, which could provide key indicators of the structural damage. The measured structural responses can be transmitted to a remote central location, which is accessible by inspectors/engineers. Therefore, such monitoring systems reduce the need for humans to make onsite inspections. However, the quality of data obtained from the sensor network is critical to the diagnosis (i.e., determination of the health conditions of the structure). If the sensors are not placed on locations that are sensitive enough to the structural damage, the collected data is not useful for the purpose of the diagnosis. Moreover, there is a trade-off between the number of sensors and the diagnosis accuracy. As more sensors are used, the accuracy of the diagnosis becomes usually higher. However, the cost of the sensing system (including the cost of sensors themselves, installation, and maintenance) becomes higher. Furthermore, considering existing structures, in general, there is limited amount of sensors in infrastructural systems, e.g., bridges, and buildings. To address this problem, an Optimal Sensor Placement (OSP) method is proposed. The causal relationship among the sensor recordings is identified, which is quantified through Directed Information (DI). In this method, the sensors are added sequentially, i.e., one sensor at a time, until the specified number of sensors

(typically based on expert opinion and availability of resources) is satisfied. The new sensor is added at a location where the causal relationship with the existing sensors is the lowest to ensure low redundancy of the information stored in the array of sensors.

Next, consider the case of infrastructure systems and buildings on the regional scale. In the big data era, the data is not collected by a small group of people. Instead, the major technological companies collect the data from the massive number of users of their products. The users do not deliberately generate data but the data is collected from the existing information, e.g., users' posts, comments, or driving routes. Given the large number of users, the amount of data is huge. This inspired the data collection process in the study presented in this dissertation. For example, the seismic reconnaissance results could be obtained by collecting the data from the residents near the source of the earthquake, as well as other locations. In this dissertation, social media posts by people near the source of an earthquake event, news reports, as well as information from official resources, e.g., United States Geological Survey (USGS), are collected automatically after the event. Such information is summarized as an earthquake briefing, which provides valuable reference and guidance for further detailed reconnaissance (field investigation) and emergency responses. The Natural Language Processing (NLP) methods are adopted in the process of automated generation of these briefings.

1.2.2 Data Processing

In the broader context, the process of observing the response of a single structure and determining its health state is referred to as Structural Health Monitoring (SHM). As mentioned above, the data collected from the structural response could be transmitted to a remote central location. However, it is typically difficult to understand the transmitted structural response from these *continuously monitored complex structural systems*. This is attributed to different sources of hard-to-quantify uncertainties and also due to the massive amount of data mainly in the form of Time Series (TS). It is harder to interpret such TS data than data from visual inspections or testing core samples. Therefore, solely relying on sensor networks, will indeed reduce the need to make onsite inspection, but may sacrifice the accuracy. Moreover, such big data from sensor networks may even need more human experts' work to interpret these sets of TS. Therefore, this presses the need to develop a data analysis tool to interpret the data and make diagnosis decisions. With the advances of computational and statistical methods in computer science and statistics, data-driven SHM using ML is gaining more attention. In this dissertation, a novel SHM framework utilizing Deep Learning (DL) is proposed. The framework is based on a Long Short-Term Memory (LSTM) Encoder-Decoder architecture, a variant of the Recurrent Neural Network (RNN), applied to TS data. The TS data is processed through the LSTM network, where the information in the TS data is condensed into a *Latent Space Vector* (LSV), which is processed through traditional ML algorithms to output the structural health conditions, including the overall health conditions, and the location and severity of damage. To enforce the encoding (i.e.,

condensation) process of TS into the LSV without information loss, an Encoder-Decoder architecture is proposed.

Analyzing the structural response of a single structure under earthquake loading as TS at different locations is important to the estimation of the consequences of severe earthquakes. For example, in the Performance-Based Earthquake Engineering (PBEE) framework [78] [122], the Engineering Demand Parameters (EDP), such as peak interstory drift ratio and peak floor total acceleration, are used to respectively estimate the structural and non-structural losses. Such EDP could be obtained by finding the maximum values in the TS output obtained from simulations using physical models, e.g., using Finite Element Method (FEM). Typically, the FEM, herein, uses nonlinear time history analysis requiring costly time stepping and nonlinear iterative solution. Such time-consuming simulations would delay the decision-making that is based on the computed structural response. In this dissertation, faster methods for predicting the structural response are proposed, which use variants of the LSTM network, in addition to a novel network called Temporal Convolutional Network (TCN). These methods are compared against each other in terms of the accuracy of predicting the structural response. These methods could replace the tradition physical simulations for faster prediction of the structural response when immediate results are required.

Next, consider the case of infrastructures and buildings on the regional scale. A simple method to quantify the regional recovery state after an earthquake event is proposed. This recovery estimate is based on the number of relevant posts collected from the social media. The recovery is quantified as the averaged recovery states of several key aspects, e.g., water supply to the community, electricity supply to the community, and availability/resumption of the functionality of essential facilities, e.g., medical services by hospitals.

1.3 Organization of the Dissertation

This dissertation is organized into nine chapters and two appendices, summarized as follows:

- Chapter 1 introduces the motivation and the overview of the dissertation.
- Chapter 2 introduces the key theoretical foundation of the dissertation, including Structural Health Monitoring (SHM), Machine Learning (ML), Deep Learning (DL), information theory, and Directed Information (DI).
- Chapter 3 introduces the Finite Element Method (FEM) applied to several structures used by later chapters as examples.
- Chapter 4 introduces the proposed framework for SHM using Long Short-Term Memory (LSTM) applied to Time Series (TS) data of the structural response.
- Chapter 5 presents two example applications of the SHM framework proposed in Chapter 4.

- Chapter 6 introduces the structural response prediction method using DL applied to TS models and also presents two example applications.
- Chapter 7 introduces the Optimal Sensor Placement (OSP) method using DI and presents one example application.
- Chapter 8 introduces the regional reconnaissance method.
- Chapter 9 presents a brief summary and the main conclusions in addition to suggested future extensions.
- Appendix A presents the details of the FEM applied to the Reinforced Concrete (RC) frame in Chapter 3.
- Appendix B presents the details of the selected Ground Motion (GM) records used for the FEM models in this study.

Chapter 2

Theoretical Foundation

In this chapter, a brief introduction of the theoretical background of several topics used in this dissertation is given. Such theoretical background is helpful as a reference for the used key concepts. Abbreviations of the key terms are used frequently for brevity in this and the following chapters. These abbreviations are summarized at the end of the dissertation in the form of a list of Acronyms followed by a list of Symbols.

2.1 Structural Health Monitoring

Structural Health Monitoring (SHM) refers to the process of observing the response of a structure or a mechanical system over time in order to determine the current state of the structure's health. After extreme events, e.g. earthquakes, SHM is used for rapid condition screening and aims to provide, in near real time, reliable information regarding the integrity of the structure. The basic premise of most damage detection methods is that damage will alter the stiffness, mass, or energy dissipation properties of a system, which in turn will alter the measured dynamic response of the system. The process of implementing a damage detection strategy involves the observation of a system over time using periodically sampled dynamic response measurements from an array of sensors, the extraction of damage-sensitive features from these measurements, and the statistical analysis of these features to determine the current state of the system's health. Concretely, SHM can be described as a four-part process: (1) Operation evaluation, (2) Data acquisition and cleansing, (3) Feature extraction, and (4) Feature discrimination. Fig. 2.1 shows the processing chain of the SHM.

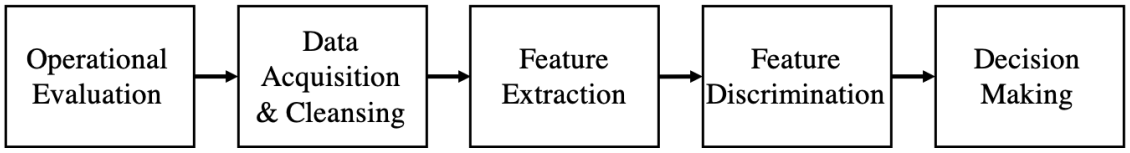


Figure 2.1: Processing chain of SHM.

The *operational evaluation* is the preliminary step of SHM to resolves the following:

1. Economic and/or life safety motives;
2. Definition of damage for the system;
3. Operational and environmental conditions of the system; and
4. Limitations on acquiring data in the operational environment.

The *data acquisition and cleansing* is the process of selecting the following:

1. Quantities to be measured;
2. Types of sensors to be used;
3. Locations where the sensors should be placed;
4. Number of sensors;
5. Sensor resolution;
6. Data recording and/or transmission bandwidth; and
7. Data acquisition/storage/transmittal hardware.

After the monitoring system is installed, the data is acquired and the raw data needs to be cleaned. Data cleaning or cleansing is the process of selectively choosing data to accept or reject for the subsequent feature selection process. Manual signal processing techniques such as filtering and decimation can be viewed as data-cleansing procedures that are commonly applied to the acquired data.

The *feature extraction* is the process of identifying damage-sensitive properties, and these properties allow one to distinguish between the undamaged and damaged structures. The features are application specific, while most of the feature extraction procedures inherently perform some form of data compression. Sometimes one could employ a combination of extracted features, rather than using a single feature, to improve the reliability of the damage detection. In order to identify features to be used, several methods are employed:

1. Past experience with measured data;
2. Laboratory specimen testing; and
3. Numerical simulation of the system's damage state making use of digital twins.

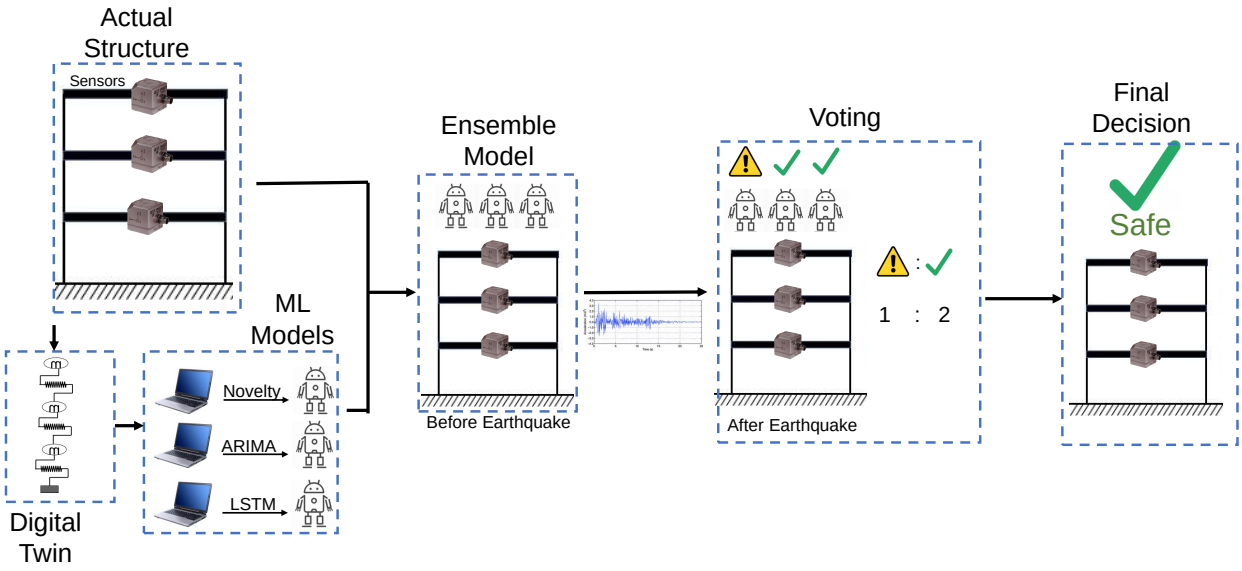


Figure 2.2: Ensemble framework for SHM of a structure, e.g., a building (Note: ML models of Novelty, ARIMA & LSTM are discussed below).

The concept of a digital twin[1] can be integrated in an ensemble framework using several ML algorithms (some of which are discussed in subsequent sections of this dissertation) and making use of the majority voting, refer to Fig. 2.2, resulting in an enhanced, efficient, effective, and accurate damage detection method. This can improve post-extreme events (e.g., earthquakes) rapid assessment and decision-making and contributing significantly to more resilient communities.

The *feature discrimination* is the process of pattern recognition or damage identification. This process could be *unsupervised* (e.g., clustering) or *supervised* (e.g., regression analysis). Essentially, feature discrimination would complete the following tasks [102]:

[1]A digital twin is a digital replica of potential and actual physical assets, processes, people, places, systems, and devices to be used for various purposes. It integrates Internet of Things (IoT), Artificial Intelligence (AI), ML, ..., etc., to create a living digital simulation model to be updated as its physical counterpart changes by continuously learning from multiple sources to represent its near real-time status, working condition, or position. This system learns from itself, using sensor data that conveys various aspects of its operating condition; from human experts, e.g., engineers with deep and relevant industry domain knowledge; from other similar machines or fleets of machines; and from the larger systems and environment of which it may be a part. A digital twin integrates historical data from past machine usage to factor into its digital model. The concept and model of the digital twin was introduced in 2002 by Grieves [38] who proposed the digital twin as the conceptual model underlying product life-cycle management. The digital twin concept consists of three distinct parts: (1) the physical product, (2) the digital/virtual product, and (3) the connections between the two products, which are data that flows from the physical product to the digital/virtual product and information that is available from the digital/virtual product to the physical environment.

1. The existence of damage in the system;
2. The location of damage in the system;
3. The type of damage present;
4. The severity of damage; and
5. The remaining lifetime of the structural system.

One important task is to identify the reliability of the proposed algorithm, or in other words to validate the algorithm. To use the task of identifying existence of damage as an example, false indications of damage fall into two categories: (1) False positive (indication of damage when none is present) and (2) False negative (no indication of damage when damage is indeed present). From their definitions, one can see that false negative readings are more detrimental from structural safety point of view than false positive while the latter leads to unnecessary disruption to functionality.

Among the above four steps, feature extraction is the most important yet challenging step, and it receives most attention in the current literature. Based on different types of damage features, SHM is categorized into several categories. The first category is based on modal properties and wave forms. Numerous research focused on modal-based SHM, and only a few are cited here for brevity. Modal parameters of structural systems have commonly been determined using System Identification (SI) methods for damage detection and health monitoring [6][7]. Changes in the modal frequency and mode shapes have been considered as the damage features since the late seventies [14]. Zak et al. [128] examined the changes in natural frequencies and modes of vibration produced by delamination in composite plates. Hu & Afzal [44] used the change of mode shapes of vibration as the damage indicator and applied a novel statistical algorithm for tested timber beam structures. Mosalam & Arici [79] used SI results for SHM and experimented with instrumented bridges. Shi et al. [106] used the Hilbert-Huang Transform (HHT) method to identify the modal frequencies and damping ratios of the Shanghai World Financial Center (SWFC) subjected to both ambient and forced excitations. Pan et al. [88] used a combination of wavelet transform, HHT, and Teager-Huang Transform (THT) as the damage feature and experimented with a numerical cable-stayed bridge. The second category is based on features from time series. Muin & Mosalam [80] used Cumulative Absolute Velocity (CAV), which is correlated to the power of the earthquake motion as well as the number of load cycles (more precisely, CAV at a certain time is proportional to the power at that time, which is true for stationary as well as non-stationary TS [80]), as a post-earthquake damage assessment feature indicator and applied it to real buildings with recorded accelerations. The SHM framework that used CAV features, called Human-Machine Collaboration (H-MC) has been developed by Muin & Mosalam [82] making use of a *physics-based* probabilistic model, e.g., using FEM involving Uncertainty Quantification (UQ), and a *novelty detection* as a one-class ML classification. A family of Auto-Regressive (AR) models was found to be effective in capturing the damage

of structures. Nair et al. [84][55] used the Auto-Regressive Moving Average (ARMA) model and identified the first three AR components as the damage-sensitive feature. Mei et al. [75] used a combination of *coefficient-based* and *residual-based* approaches with ARMA eXogenous (ARMAX) model for undamaged and damaged states, and applied the method to a small-scale five-story frame. Gao et al. [35] used the Auto-Regressive Integrated Moving Average (ARIMA) model for damage identification of a shaking table steel test building. The third category is vision-based (including crack-based) SHM. Kong & Li [54] proposed crack detection under repetitive fatigue loads based on image overlapping. Ai et al. [3] proposed a region-based active contour framework with the intensity cluster energy and applied the algorithm to a high-speed railway system. Moreover, there are methods that are not classified into the above three categories. Moehle et al. [78][122] used the peak interstory drift ratio and peak floor total acceleration as the damage indicators in terms of the EDP in the context of Performance-Based Earthquake Engineering (PBEE) framework[2], Figs. 2.3 and 2.4. Other SHM methods include the utilization of Acoustic Emission (AE), guided-wave, ..., etc., which are not discussed further in this dissertation for brevity.

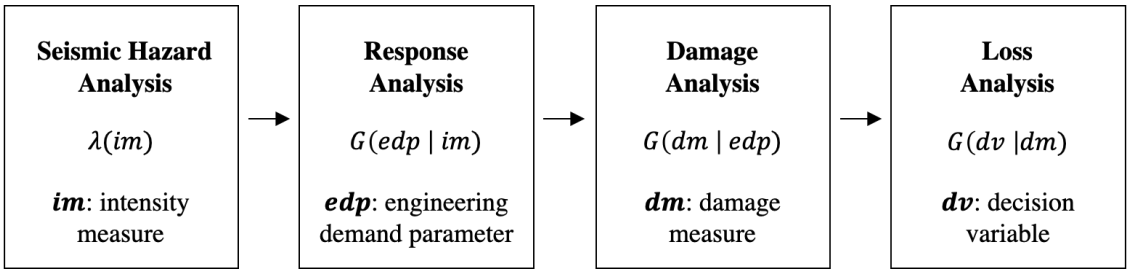


Figure 2.3: PEER PBEE framework [122] (Note: $\lambda(im)$ is the seismic hazard from *hazard analysis* to quantify the mean annual rate of exceedance of a given Intensity Measures im, e.g., rate at which Peak Ground Acceleration exceeds a specified value for a particular location in a given year. Conditional probabilities are obtained using *response analysis* for $G(edp|im)$, *damage analysis* for $G(dm|edp)$ involving Damage Measures dm, e.g., moderate damage, and *loss analysis* for $G(dv|dm)$ involving Decision Variables dv, e.g., downtime. These *four analyses* are combined using *total probability theory* to obtain $G(dv|im)$ for decision-making).

With the development of computational and statistical methods in computer science and statistics, data-driven SHM with ML is gaining more attention. The development of ML algorithms has been attempted in SHM as early as mid-nineties [109]. In most research efforts that applied ML algorithms in SHM, they are in combination with traditional damage features and apply different learning algorithms to make comparisons of the performance of the different algorithms. Lam et al. [59] used the changes in *Ritz vectors* as the features and trained an Artificial Neural Network (ANN) to identify the damage pattern, and

[2]The PBEE methodology has been under development since mid-nineties by the Pacific Earthquake Engineering Research (PEER) Center, `https://peer.berkeley.edu/`.

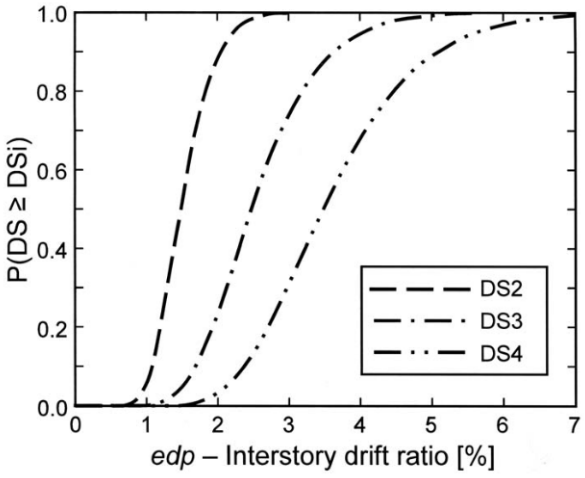


Figure 2.4: Fragility curves for Damage States (DS) used in PBEE [122] (P: Probability).

experimented with a numerical truss example. Pan et al. [88] applied a Support Vector Machine (SVM) algorithm to a combination of vibration-based features. Liang et al. [66] used a combination of several energy-based parameters and classified the severity of the post-earthquake damage state. Gao et al. [35] used a combination of ARIMA and ML for damage pattern analysis. To the best of the author's knowledge, currently, there are two important directions of the data-driven SHM. The first direction is vision-based SHM, which utilizes fixed cameras and movable ones mounted on Unmanned Aerial Vehicles (UAV's) to capture photographs of the structure, and the SHM system makes automatic identification of the structural state using Computer Vision (CV) methods. The second direction is TS-based SHM to process the transmitted TS data and analyze it using TS models. Sensor-based SHM, in combination with data analysis techniques, which could make the detection and assessment online and automated, is becoming the trend for critical civil infrastructures and buildings to ensure fast, efficient, and accurate assessment of the performance. Details of ML and Deep Learning (DL) methods are given in the next section.

2.2 Machine Learning & Deep Learning

2.2.1 Machine Learning

Machine Learning (ML) is a body of knowledge that attempts to construct computational relationships between the observed data and several computational rules. It is characterized by the fact that these computational rules are inferred (learned) from the bases of the observational evidences. Essentially, the *learning theory* is designed to address the following three main problems:

1. Classification, i.e., association of measured quantities with a class label;
2. Regression, i.e., construction of a mapping between a continuous input variable and a continuous output variable; and
3. Density estimation, i.e., estimation of the Probability Density Function (PDF).

ML algorithms are primarily divided into two categories: (1) supervised learning, and (2) unsupervised learning [100]. *Supervised learning* is the task of learning based on example input-output pairs. It infers a function from the labeled (annotated) training data. In contrast, *unsupervised learning* is a type of learning without pre-existing labels. Between supervised learning and unsupervised learning, there is also *semi-supervised learning* [17], an intermediate learning scheme that utilizes both supervised and unsupervised techniques. For the SHM tasks, most research activities are focusing on supervised learning.

There are various types of ML models, most of which have been employed in SHM tasks. The most common models are listed as follows:

- Linear Regression;
- Logistic Regression;
- Support Vector Machine (SVM);
- Neural Network (NN) including ANN, Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN);
- Random Forest;
- Decision Trees; and
- Bayesian Networks.

To train and evaluate the ML model, the dataset is split into a *training set*, an optional *validation set*, and a *test set*. The training set is used to train the model. The validation set is used to find the best model configuration from varying model *hyper-parameters*. The test set is used to evaluate the model. ML models should be generalizable, i.e., they should demonstrate robust performance to unseen data sets. In this regard, the model should achieve good performance not only on the training set, but also on the validation and test sets, which are unseen during the training process. Sometimes, the model fails to achieve good performance for both the training and test sets. This is called *under-fitting*. On the other hand, if the model achieves good performance for the training set, while it gives much worse performance for the unseen test set, then the model lacks generalization and suffers from *over-fitting*. Fig. 2.5 shows under-fitting, "perfect" fitting, and over-fitting cases of a hypothetical training data set, where black dots are the data points, and red lines are the ML model predictions.

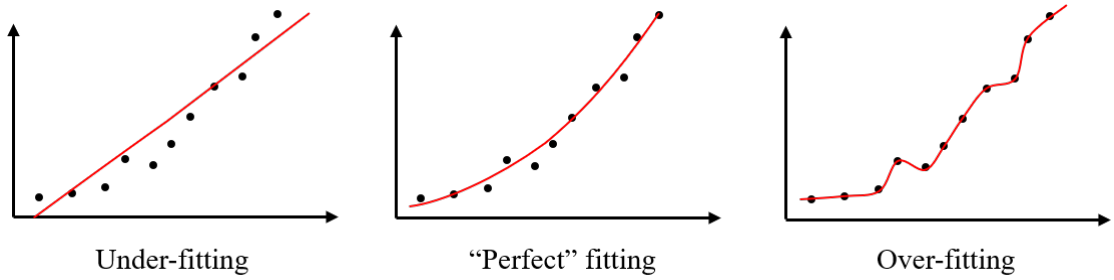


Figure 2.5: Illustration of the under-fitting, "perfect" fitting, and over-fitting of ML models.

To evaluate the performance of models for a classification task, the *confusion matrix* is often used. It is a specific table for visualizing the performance of the ML model, Fig. 2.6. Each element (cell) represents the number of instances of a predicted class for the actual corresponding class. For example, in the right part of Fig. 2.6, 10 data points are incorrectly classified by the ML model as negative, while the true label is positive. On the other hand, 100 data points are correctly classified by the ML model as negative, while the true label is indeed negative. Therefore, the confusion matrix of an accurate model should have more instances on the diagonal elements than on the off-diagonal ones.

	Predicted Positive	Predicted Negative
True Positive	True Positive (TP)	False Negative (FN)
True Negative	False Positive (FP)	True Negative (TN)

	Predicted Positive	Predicted Negative
True Positive	50	10
True Negative	5	100

Figure 2.6: General layout of the confusion matrix (left) and a specific example (right).

Different evaluation metrics are put forward to evaluate the performance of the ML models. One of the most important evaluation metrics is the *accuracy*, which is used frequently in this dissertation. In the context of Fig. 2.6, the accuracy is defined as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2.1}$$

where, as defined in the left part of Fig. 2.6, TP, TN, FP, and FN stand for True Positive, True Negative, False Positive, and False Negative, respectively. Using the example in the right part of Fig. 2.6, one obtains,

$$\text{accuracy} = \frac{50 + 100}{50 + 100 + 5 + 10} = 0.909 = 90.9\%.$$

2.2.2 Feature Selection

In ML problems that involve learning a "state-of-nature" from a finite number of data samples, where each data comes from a high-dimensional feature space, there is a trade-off in terms of the number features. If less features are selected and used, the trained model tends to be simple, and is likely to cause the problem of under-fitting, described above. In contrast, if more features are selected and used, the trained model tends to be complex, and is likely to cause the problem of over-fitting, described above. An explanation of the over-fitting is that as the dimension of the feature space (i.e., the number of features) increases, the volume of the feature space grows exponentially[3]. On such high-dimensional space, the data samples are sparsely distributed, and it is harder to learn a model that have enough generalization capability. This problem is called the *curse of dimensionality*. Therefore, feature selection and/or transformation is important to overcome this problem.

Algorithms to select from a set of features is studied by researchers [57][36] to alleviate the problem of the curse of dimensionality. In this dissertation, such feature selection methods are adopted as the key method to solve the problem of Optimal Sensor Placement (OSP). In the setting of this study, for a considered structure where the number of possible locations of sensor installations is high and known, a plan is to be proposed by selecting a subset of the possible locations where sensors will be installed. The ultimate goal of such plan is to ensure satisfactory performance of the structural diagnosis, while reducing the cost of the monitoring/diagnosis system by limiting the maximum number of sensors. Details of this topic are discussed in Chapter 7.

One method to make a subset selection is to explore all combinations of features, and select the subset of features that produces the most satisfactory results. This method is, in practice, impossible to implement, as the total number of non-empty combinations are $2^d - 1$, where, d is the total number of features. There is a trade-off between the ML model performance and the feature selection algorithm running time. If d is large, there is no algorithm that guarantees finding the optimal subset of features and runs in acceptable time. Therefore, algorithms have been proposed to find the sub-optimal or "reasonable" subset in acceptable time. There are two important heuristics of such algorithm, which are forward and backward step-wise selections. The *forward step-wise selection* starts with a null set of features (i.e., zero feature), and the best features are added to the set repeatedly (i.e., iteratively) until the preset desired number of features is satisfied (e.g., the validation errors started increasing instead of decreasing). The *backward step-wise selection* starts with all features, and the worse features are removed repeatedly. There are no obvious advantages between these two heuristics. In practice, forward step-wise selection is a better choice if a small subset of the features will be selected (as conducted in the present study of OSP), and backward step-wise selection is a better choice if most of the features will be kept.

[3]In other words, the volume of the feature space is an exponential function of the dimension of the feature space. An analogy to this phenomenon is that, the "volumn" of an n-dimensional hypercube is l^n, where l is the edge length of the hypercube.

2.2.3 Deep Learning

With the advancement of computational power that is boosted by high performance Graphic Processing Units (GPU's) and accessibility of large amount of data, Deep Learning (DL) is now gaining huge attention. Conventional ML techniques are limited in their ability to process natural data in their raw form [61]. In comparison to the traditional ML, DL uses multiple layers to progressively extract higher level features. Therefore, DL allows the machine to process the raw data and automatically discovers the representations. Through the process of greedy layer-by-layer progression [61], the model has higher expressiveness, which enables the model to learn complex patterns of the real data. Two basic building blocks of DL models are Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). In the rest of this section, the theory of the CNN and RNN as they pertain to this dissertation are presented in detail. It is to be noted that CNN is not the focus of this study but it is included here for completeness as it is a commonly used DL approach. On the other hand, RNN and its related Long Short-Term Memory LSTM model are utilized in this dissertation.

2.2.3.1 Convolutional Neural Network

A CNN is designed to process data that come in the form of multiple arrays. For example, a colored image composed of three two-dimensional (2-D) arrays containing pixel intensities in the three color channels (Red, Green, and Blue or RGB) [61]. They are widely adopted by the CV community. Fig. 2.7 shows a demonstration of the mechanism behind CNN in the identification of the species of the input image, which is an image of Samoyed (a breed of large herding dogs with thick, white, double-layer coats). Higher score on the top stands for higher possibility, and the network successfully identified the animal in the image as indeed a Samoyed dog.

The CNN has four key improvements over ordinary NN [61]. These are as follows:

1. Local connections: Units in one CNN layer are connected to *local* patches of the previous layer through a set of weights called a *filter bank*. This is in contrast to ordinary NN, which is Fully Connected (FC), i.e., all units in the previous layer is connected to all units in the next layer. Refer to Fig. 2.8 for a comparison between CNN and FC NN.

2. Shared weights: The filter bank is shared among all units, i.e., the weights are the same for all units in a layer because the local statistics of images are *location invariant*.

3. Pooling: Pooling layers could coarse-grain the position of the features and remove variance due to small shifts and distortions of the features. A common pooling scheme is *max-pooling*, which calculates the maximum of a local patch of neighboring units. It reduces the data dimension by outputting a single value for the local patch of neighboring units.

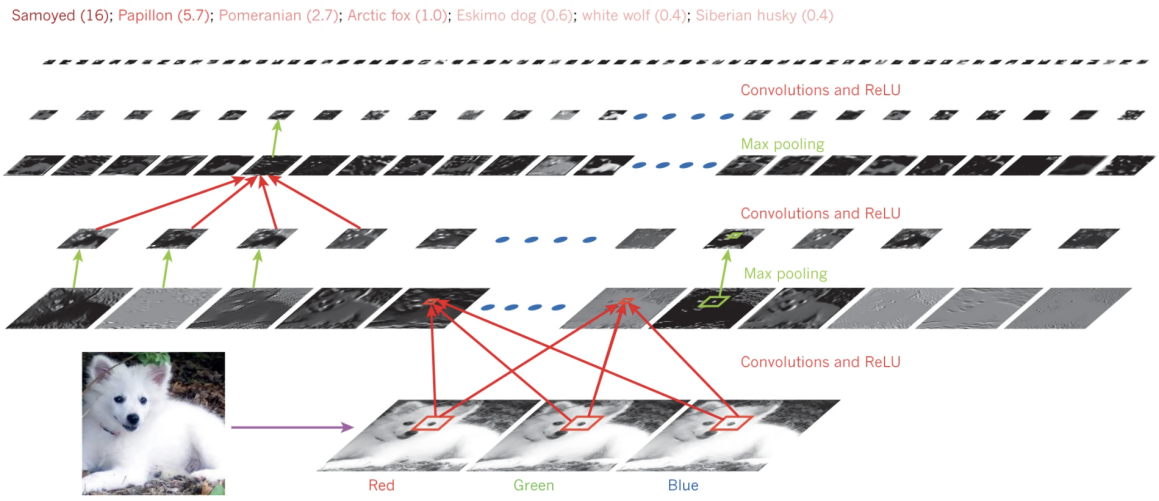


Figure 2.7: Outputs of layers of a CNN applied to an image of a Samoyed dog [61].

4. Multi-layer: In images, higher-level features come from the composition of the lower-level features. Pixel level features are combined into edges, edges are combined into motifs, motifs are combined into parts, and finally parts are combined into objects.

The mechanisms of CNN (in particular, filters and pooling) and a comparison between CNN and ordinary NN (in a FC sense) are shown in Fig. 2.8. Since CNN is not the main focus of this dissertation, these mechanisms of the CNN model are not presented herein in detail[4].

The CNN is widely used in SHM, particularly in the vision-based monitoring. Cha et al. [15][16] used CNN to detect concrete cracks and proposed a region-based algorithm for detecting multiple damage types. Bang et al. [10] used a pixel-level detection method for identifying road cracks in black-box images using deep CNN. Two drawback are related to the limitations from the available amount of images targeting SHM and from the required computing power as the CNN structure goes deeper and becomes complex. These limits affect the quality of the detection results. Gao & Mosalam [33] used Transfer Learning (TL), which improved the accuracy of damage state classification despite these limitations. To overcome the problem of lack of image data targeting SHM, data augmentation techniques, like Generative Adversarial Networks (GAN), are used [32], and a larger dataset is created [34]. As a side note, in terms of vision-based SHM, instead of deploying traditional sensor networks, a new method of image collection is using cameras mounted on Unmanned Aerial Vehicles (UAV's) and robots, which could overcome the problem that images from static cameras are sometimes hard or even impossible (because of access limitations and safety)

[4]Interested readers are referred to online resources, e.g., `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`.

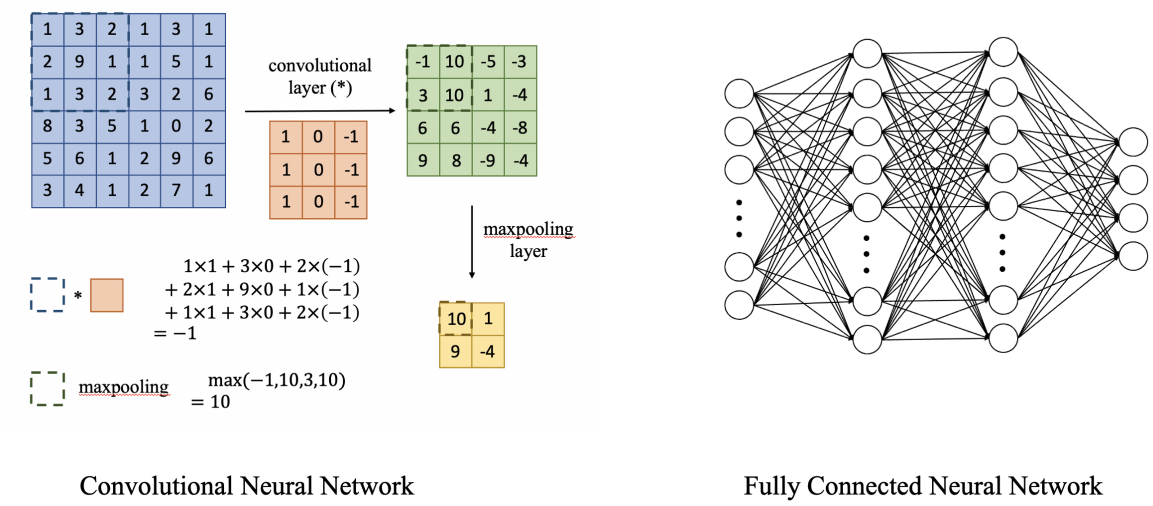


Figure 2.8: CNN vs. FC NN.

to capture after major earthquakes. The CNN can be used as TS models. Uni- or multi-parameter TS could be "scanned" by the filters to observe the local variations. However, the available SHM literature using CNN for TS is scarce. A more common DL sequence model for TS is the RNN.

2.2.3.2 Recurrent Neural Network

The RNN is widely used as a DL sequence model. The difference between RNN and ordinary NN is that in RNN both the input from the current step and the state information from the previous step are considered in the output of current step. In this way, RNN captures the *intrinsic correlation* among current step and previous steps. Fig. 2.9 shows the architecture of a simple RNN, where, x_t, h_t, and y_t are respectively the input vector, hidden state vector, and output vector, all at time step t. The hidden state vector implicitly contains information about the history of all the past elements of the sequence. Moreover, h_t (Eq. 2.2) and y_t (Eq. 2.3) depend on the current step input vector, x_t, and the previous step hidden state vector, h_{t-1}. In addition, the previous step hidden state vector, h_{t-1}, depends on the previous step input vector, x_{t-1}, and the hidden state vector before, h_{t-2}. Therefore, the hidden state vector at each step contains sequential information of all previous steps.

$$h_t = f(W_{hh}h_{t-1} + W_{hx}x_t + b_h), \tag{2.2}$$

$$y_t = f(W_{yh}h_t + b_y), \tag{2.3}$$

where, f is the activation function, W_{hh}, W_{hx}, and W_{yh} are weight matrices as shown in Fig. 2.9, and b_h and b_y are the respective bias vectors, Fig. 2.9. Note that in RNN, the weights

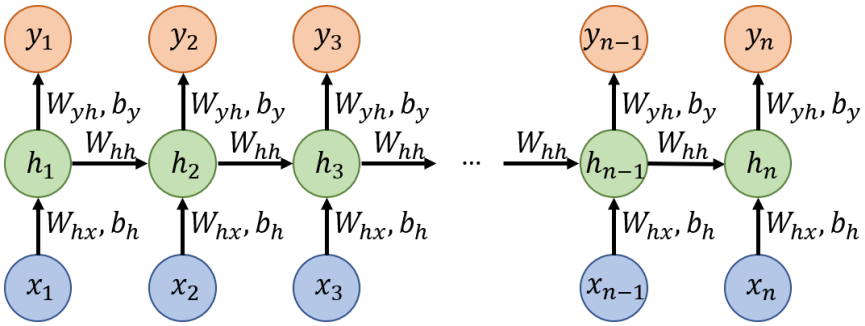


Figure 2.9: A simple RNN.

are shared among steps. In comparison to the ordinary NN, RNN can be seen as a very deep feed-forward network in which all the layers share the same weights [61]. Therefore, one problem of deep NN, called *gradient vanishing*, also takes place in RNN. Moreover, because RNN is "very" deep, the problem is exacerbated, thereby modifications to the original RNN architecture are put forward to resolve this problem, as discussed later in this subsection.

There are two basic variants of RNN. The first is the Bi-directional RNN, where the output vector at time t, y_t, depends on the input vector at time t, x_t, and the hidden state vectors at the previous step (time $t-1$), h_{t-1}, and at the next step (time $t+1$), h_{t+1}. In this way, current step not only considers previous steps but also future ones. The second is the Multi-layer RNN, or Stacked RNN, which stacks layers of RNN cells for each step. The lower layer of each step passes the hidden state vector to the upper layer as the input. Therefore, the hidden state vector not only goes to the next step at the same layer, but also to the same step at the next layer. In this way, the *complexity* and *expressiveness* of the RNN are improved. Generally, the number of layers for Multi-layer RNN is two to four, and three layers RNN is already considered "deep". Figs. 2.10 and Fig. 2.11 show the architectures of the Bi-directional RNN and Multi-layer RNN, respectively.

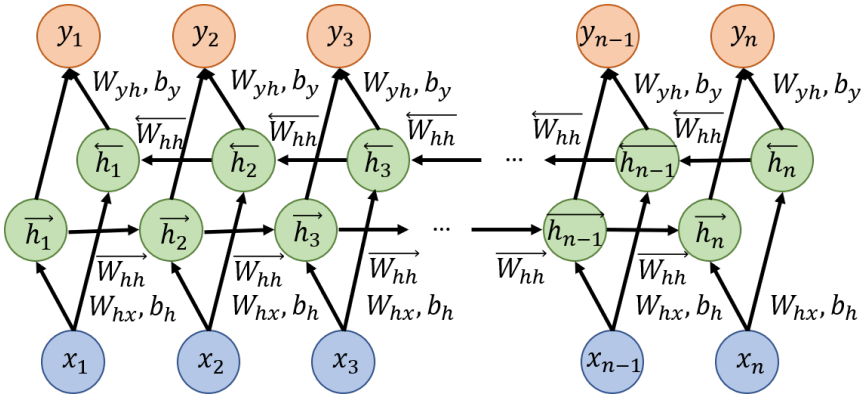


Figure 2.10: Bi-directional RNN.

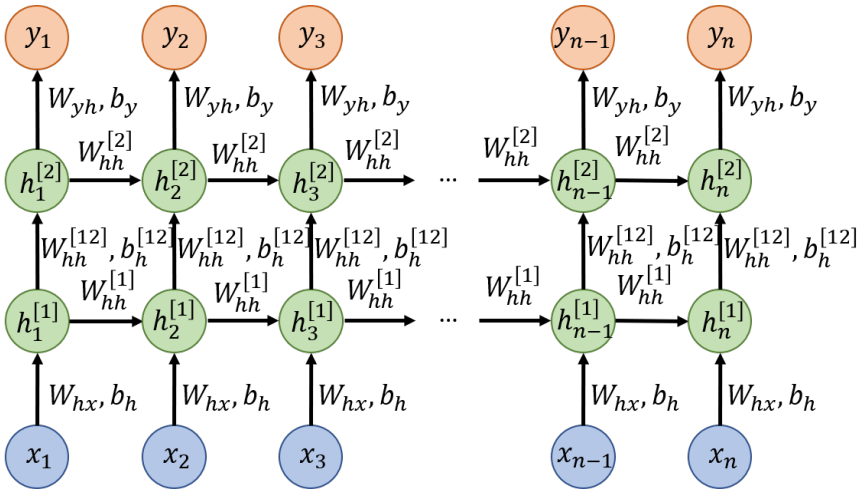


Figure 2.11: Two-layer (Stacked) RNN.

The training process of the RNN is similar to that of the ordinary NN, e.g., using *Gradient Descent* [99] to minimize the training *loss function*. However, as mentioned above, RNN has some intrinsic problems, e.g., gradient vanishing. NN uses *back propagation* to minimize the training loss function, and the gradient of the weights comes from the *chain rule* of derivatives of composite functions. In RNN, the gradient from the output is difficult to back propagate to affect the weights of earlier layers. Therefore, the final output value is more influenced by the last steps. The gradient vanishing problem makes the RNN hard to capture very long dependencies inside the sequential data. As an intuitive example, the model needs to choose between "was" and "were" to fill in the *BLANK* for this sentence: "The cat, (a long sentence), (*BLANK*) full." The correct answer, from human judgement, would be "was," as "cat" is singular. However, the RNN is not good at handling long range dependencies, and it forgets the number of cats mentioned at the beginning of the sentence after processing the long sentence in the middle.

Mathematically, in the forward propagation of the RNN model, the output at the final step n (for illustration, the *sigmoid function*[5] is used as the *activation function* herein) is:

$$\begin{aligned}
y_n &= \sigma(W_{yh}h_n + b_y) \\
&= \sigma(W_{yh}(\sigma(W_{hh}h_{n-1} + W_{hx}x_n + b_h)) + b_y) \\
&= \sigma(W_{yh}(\sigma(W_{hh}(\sigma(W_{hh}h_{n-2} + W_{hx}x_{n-1} + b_h)) + W_{hx}x_n + b_h)) + b_y) \\
&= \cdots \\
&= \sigma(W_{yh}(\sigma(W_{hh}(\sigma(W_{hh}(\cdots h_2 \cdots) + W_{hx}x_{n-1} + b_h)) + W_{hx}x_n + b_h)) + b_y) \\
&= \sigma(W_{yh}(\sigma(W_{hh}(\sigma(W_{hh}(\cdots(\sigma(W_{hh}h_1 + W_{hx}x_2 + b_h))\cdots) \\
&\quad + W_{hx}x_{n-1} + b_h)) + W_{hx}x_n + b_h)) + b_y).
\end{aligned} \tag{2.4}$$

[5] A sigmoid function, $\sigma(x)$, is a mathematical function commonly used in ML & DL. It has a characteristic "S"-shaped curve and is expressed as: $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$.

The gradient of the loss is (using the chain rule of derivatives of the composite functions) given by:

$$\begin{aligned}\frac{\partial L}{\partial W} &= \frac{\partial L}{\partial y_n}\frac{\partial y_n}{\partial h_n}\frac{\partial h_n}{\partial h_{n-1}}\cdots\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W} \\ &= \frac{\partial L}{\partial y_n}\frac{\partial y_n}{\partial h_n}\left(\prod_{t=2}^{n}\frac{\partial h_t}{\partial h_{t-1}}\right)\frac{\partial h_1}{\partial W},\end{aligned} \tag{2.5}$$

where, L is the loss function by the outputs compared to the *ground truth*. One could update the model parameters by:

$$W \leftarrow W - \alpha\frac{\partial L}{\partial W}, \tag{2.6}$$

where, α is the *learning rate*, a hyper-parameter that needs to be tuned. At step t, we have the following expression:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{hx}x_t + b_h). \tag{2.7}$$

Therefore, one could compute the derivative of h_t as follows:

$$\begin{aligned}\frac{\partial h_t}{\partial h_{t-1}} &= \sigma'(W_{hh}h_{t-1} + W_{hx}x_t + b_h)\cdot\frac{\partial}{\partial h_{t-1}}\left(W_{hh}h_{t-1} + W_{hx}x_t + b_h\right) \\ &= \sigma'(W_{hh}h_{t-1} + W_{hx}x_t + b_h)\cdot W_{hh},\end{aligned} \tag{2.8}$$

where, σ' is the derivative of the sigmoid function σ with respect to h_{t-1}. Plugging Eq. 2.8 into Eq. 2.5, the back propagated gradient is determined as follows:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y_n}\frac{\partial y_n}{\partial h_n}\left(\prod_{t=2}^{n}\sigma'(W_{hh}h_{t-1} + W_{hx}x_t + b_h)\cdot W_{hh}\right)\frac{\partial h_1}{\partial W}. \tag{2.9}$$

Without calculating the gradient explicitly, one could observe that Eq. 2.9 tends to vanish (approaches zero) when n is large. The reason is that the derivative of the activation function, i.e., σ', is smaller than 1. Therefore, the upper bound of the absolute value of the gradient approaches 0 when this derivative is multiplied for $(n-1)$ time steps, i.e., from $t = 2$ to n.

In order to overcome the above-mentioned gradient vanishing problem and learn the long term dependencies, variants of the RNN have been proposed. Among these variants, the most famous ones are Gated Recurrent Unit (GRU) and LSTM. A memory cell of the GRU is shown in Fig. 2.12. The GRU tries to overcome the gradient vanishing problem by using a gating mechanism, which could regulate the flow of information. Each GRU cell has two gates, which are called *update gate* and *reset gate*. The update gate decides what information to throw away and what new information to add from prior steps. The reset gate decides how much past information to forget. Both gates in Fig. 2.12 are implemented by a sigmoid layer, involving h_{t-1} and x_t to produce r_t for the reset gate, and z_t for the update gate. The outputs of this sigmoid layer are between 0 and 1. The GRU equations are as follows:

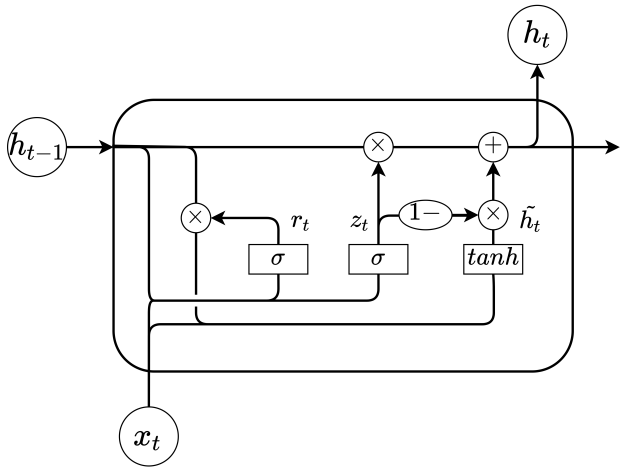


Figure 2.12: GRU cell [98].

$$z_t = \sigma(W_{zx}x_t + W_{zh}h_{t-1} + b_z), \tag{2.10}$$

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r), \tag{2.11}$$

$$\tilde{h_t} = \tanh(W_{hx}x_t + W_{hr}(r_t \odot h_{t-1}) + b_h), \tag{2.12}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h_t}. \tag{2.13}$$

where, W_{zx}, W_{zh}, W_{rx}, W_{rh}, W_{hx}, and W_{hr} are weight matrices, b_z, b_r, and b_h are respective bias vectors, z_t and r_t are the update and reset gate activation vectors, respectively, $\odot$ is the Hadamard product (element-wise multiplication, i.e., for two matrices of the same size A & B, $(A \odot B)_{ij} = A_{ij} \times B_{ij}$), and σ and tanh are the sigmoid and hyperbolic tangential activation functions, respectively. It is noted that although the GRU came later than the LSTM (approximately 17 years, as GRU and LSTM were proposed in 2014 [21] and 1997 [42], respectively), GRU is not as widely used as LSTM in current sequence models. The GRU was also explored in this study, and it was found that the GRU did not produce any improvement over the performance of the the LSTM (in terms of the training and validation loss, see Chapter 4 for explanation of these losses). Therefore, LSTM is used in this study and GRU is not discussed further in this dissertation.

2.2.3.3 Long Short-Term Memory

The LSTM was introduced by Hocheriter & Schmidhuber [42] in 1997. Similar to the GRU, it uses gating mechanism to control the flow of information. The architecture of the LSTM network and a typical cell configuration are shown in Figs. 2.13 and 2.14, respectively.

In comparison to the GRU, the LSTM model has one extra vector, called *cell state vector*, which is denoted as c_t. Each LSTM cell has 3 gates, which are called the *forget gate*, the

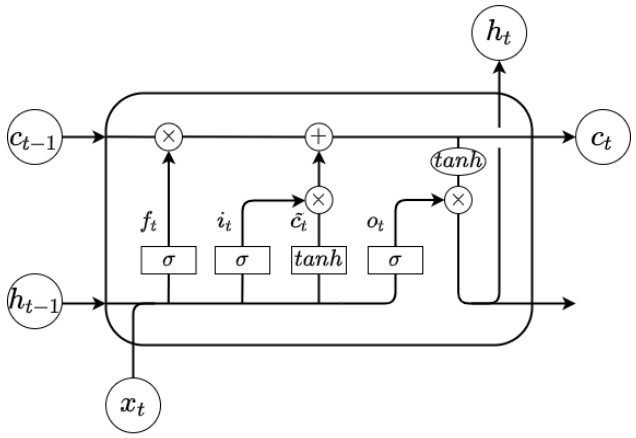


Figure 2.13: LSTM cell [45].

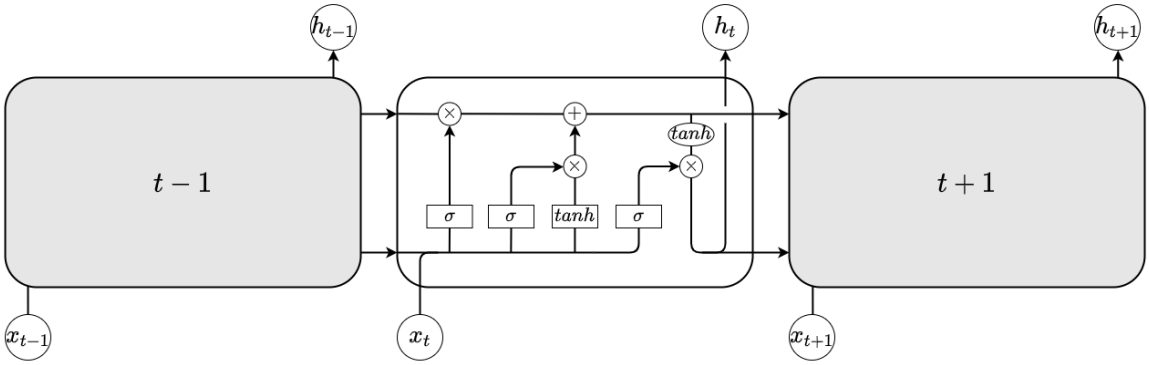


Figure 2.14: A typical LSTM cell configuration [117].

input gate, and the *output gate*. The forget gate decides what is relevant to keep from prior steps. It is implemented by a sigmoid layer called the "forget gate layer," which looks at h_{t-1} and x_t, and outputs a number between 0 and 1. This number is multiplied by c_{t-1}, as a discount for the cell state in the previous step. The input gate decides what information is relevant to add from the current step. It is also implemented by a sigmoid layer called the "input gate layer," which decides how much new information from h_{t-1} and x_t are added to the current cell state. Finally, the output gate determines what the next hidden state should be. It is implemented by a third sigmoid layer called the "output gate layer," which decides what part of the cell state the model is going to output as the hidden state for the current step. Accordingly, the output at each step depends on x_t, h_{t-1}, and c_{t-1}. An intuitive understanding of the LSTM is as follows: The cell state works like a conveyor belt, and it passes through the LSTM cell with limited interactions within the cell. Unless it

is intentionally forgotten, information from previous steps could be well-preserved. Each memory cell could selectively forget and remember sections of the previous states. The LSTM equations are as follows:

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f), \tag{2.14}$$

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i), \tag{2.15}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o), \tag{2.16}$$

$$\tilde{c_t} = \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c), \tag{2.17}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t}, \tag{2.18}$$

$$h_t = o_t \odot \tanh(c_t), \tag{2.19}$$

where, W_{fx}, W_{fh}, W_{ix}, W_{ih}, W_{ox}, W_{oh}, W_{cx}, and W_{ch} are weight matrices, b_f, b_i, b_o, and b_c are respective bias vectors, f_t, i_t, and o_t are the respective forget, input, and output gate activation vectors, and $\tilde{c_t}$ is the cell input activation vector.

Cho et al. [20] have shown that the hidden state could capture the semantically[6] and syntactically[7] meaningful representation of the data. They proposed an *Encoder-Decoder* structure, which is able to learn the mapping from a sequence of an arbitrary length to another sequence. It is seen that the encoder in this context is expected to extract the features from the input, and learn a good representation of the input data, which is proved by the fact that the original data can be reconstructed by using the decoder [63]. This structure has been used in the Natural Language Processing (NLP), e.g., an English sentence as the encoder input and its French translated sentence as the decoder output. Recently, this architecture is also used in TS [71]. Variants of the original structure are proposed, e.g., intuitive replacement of a simple RNN by a LSTM. One LSTM Encoder-Decoder structure is shown in Fig. 2.15.

The architecture in Fig. 2.15 consists of two LSTM networks that act as an encoder and a decoder pair. The encoder maps a variable-length source (input) sequence to a fixed-length vector, which is the hidden state vector of the last step, also known as the Latent Space Vector (LSV), as the internal representation of the input. On the other hand, the decoder maps the vector representation back to a variable-length target (output) sequence. In practice, a shallow FC NN is often added to the output of the decoder. An intuitive understanding of the encoder-decoder architecture is that the LSV is acting as a "bottleneck". In order for the decoder to decode the information of the encoder input, the hidden state needs to compress the input without significant information loss. Therefore, the hidden state is preserving the information of the input, and actions toward the original input could also be performed on the compressed LSV, which are typically easier to perform.

[6]Semantically refers to the study of the relationships between symbols or signs such words, phrases, sentences, and discourses, and what these elements of data in general mean or stand for.

[7]Syntactically refers to investigation of the rules, principles, and processes which determine the structure of sentences in human languages or in data in general.

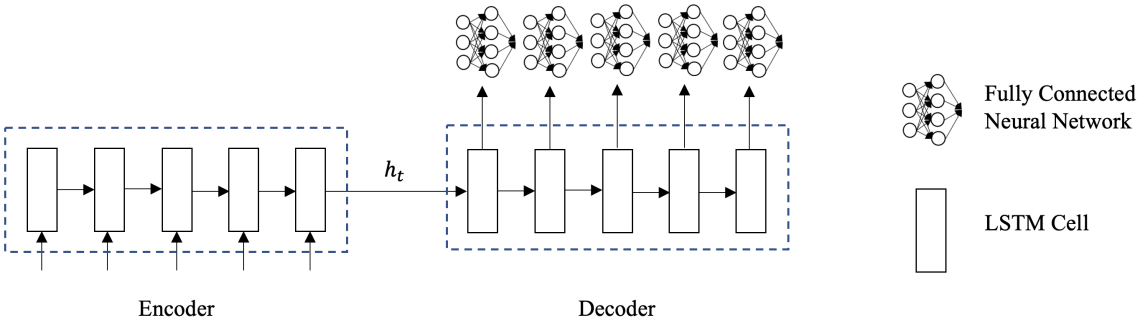


Figure 2.15: LSTM Encoder-Decoder architecture.

The RNN family of models has been used in many applications, including but not limited to speech recognition [37], text generation [112], and machine translation [20]. Despite the complexity of the LSTM model, it is much more frequently used in recent years than the traditional RNN model, because of its ability to capture long term dependencies. The RNN models could also be combined with CV models, which could capture the temporal dependencies of images. Some studies combined the RNN models with CNN [107]. Ng et al. [47] used the Recurrent CNN for video classification. Another research direction is the introduction of an *attention mechanism* [9], which initially comes with Encoder-Decoder structure and could improve the performance for long inputs for the encoder. This mechanism allows the model to automatically search for parts of an input that are relevant to the prediction in the decoder, by assigning weights to the encoder input and decoder output pairs. Higher relevant pairs receive higher score and the decoder output pays more attention to these input parts in the form of weighted sum of contributions from all the encoder inputs.

The RNN family of models is also widely used as TS models for tasks like forecasting and classification. In comparison to the TS models that are described previously (e.g., the family of Auto-Regressive models), which are mainly linear models, the RNN models introduce nonlinearity through the activation functions and capture the temporal and spatial dependencies. This capability expands the expressiveness of the RNN family of models and improves the quality of their identification. There are various TS forecasting competitions [70], and the RNN family of models was able to outperform traditional TS models. Again, in comparison to ordinary RNN, the LSTM is more widely used, as the number of steps in the considered TS is usually high. The Encoder-Decoder structure is also used in TS models. Malhotra et al. [71] used this structure to detect the inherently unpredictable ("anomaly") TS by observing the reconstruction error. Tang et al. [113] combined the attention mechanism with the Encoder-Decoder structure, and compared the results with the state-of-the-art models for several real world tasks (discriminating two actors transiting between yoga pose, and detecting transient electromagnetic events associated with lightning). Che et al. [18] addressed the *missing value problem* by using two representations of the

missing patterns, which are *masking* and *time interval*. The RNN family of models has been explored in many fields and experimented with several publicly available datasets [108].

2.3 Information Theory

Information theory is the scientific study of the quantification, storage, and communication of digital information. In this dissertation, the concept of Directed Information (DI) is essential to the proposed OSP algorithm described in Chapter 7. In this section, important relevant measures in information theory [104] are briefly described. Let $X_{1:T} = \{X_t\}$ and $Y_{1:T} = \{Y_t\}$, where, $t \in \{1, ..., T\}$ represent two uni-variate sequences. In other words, $X_{1:T}$ and $Y_{1:T}$ are abbreviated symbols representing the sequence from time step 1 to T for X_t and Y_t, respectively. In the information theory, DI describes the flow of information within two sequences, $X_{1:T}$ and $Y_{1:T}$. In particular, DI describes the *causal relationship* $X_{1:T} \to Y_{1:T}$, which holds, if the likelihood of $Y_{1:T}$ occurring alone is lower than the likelihood of $Y_{1:T}$ occurring conditioned on $X_{1:T}$, and $X_{1:T}$ occurs no later than the corresponding event in $Y_{1:T}$. Before the definition of DI is formally presented and discussed, two preliminary concepts, which are the *entropy* and *mutual information*, are presented first.

2.3.1 Entropy

Let X be a random variable with the occurring probability $p_x = P_X(x) = P_X(X = x)$. The *surprise* of event x occurring is defined as $-\log_2(p_x)$. The term quantifies how "surprising" it would be if an event occurred. For example, if $p_x = 1.0$, i.e., the event is *certain*, then the surprise of the event is $-\log_2(1.0) = 0.0$. In contrast, if $p_x = 0.0$, i.e., the event is *impossible*, then the surprise of the event is $-\log_2(0.0) = -\lim_{x \to 0.0^+}(x) = -(-\infty) = \infty$. The *entropy* is the average level of surprise of the variable's possible outcomes. Given the possible outcomes of X being in the *sample space* $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, with occurring probabilities $P_X(x_1), P_X(x_2), \ldots, P_X(x_n)$, respectively, the entropy of X is defined as (for brevity, the subscript 2 of the $\log_2$ operation in the equations of this section is dropped, i.e., unless otherwise specified, the base of the log operation is 2 by default in this section):

$$H(X) = -\sum_{x \in \mathcal{X}} P_X(x) \log\left(P_X(x)\right). \tag{2.20}$$

The idea of the information theory is that the "informational value" of a message depends on the the degree of surprise. If an event with high probability occurs (and accordingly the surprise of the event is low), the message carries little new information. In contrast, if an event with low probability occurs (and accordingly the surprise of the event is high), the message is much more informative. The entropy quantifies the average amount of information conveyed. A *conditional entropy* quantifies the average amount of information conveyed from one random variable Y given the value of another random variable X, with respective sample

spaces $\mathcal{Y}$ and $\mathcal{X}$, and is defined as follows:

$$H(Y|X) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log\left(\frac{P_{(X,Y)}(x,y)}{P_X(x)}\right), \tag{2.21}$$

where, $P_{(X,Y)}(x,y)$ is the joint probability distribution of random variables X & Y. Recall the relationship between joint, $P_{(X,Y)}$, marginal, P_X & P_Y, and conditional, $P_{Y|X}$ & $P_{X|Y}$, probabilities.

$$P_{(X,Y)}(x,y) = P_X(x)P_{Y|X=x}(y). \tag{2.22}$$

2.3.2 Mutual Information

The Mutual Information (MI) of two random variables is a measure of the *mutual dependence* between the two variables. For discrete random variables X and Y, the MI is as follows:

$$I(X,Y) = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log\left(\frac{P_{(X,Y)}(x,y)}{P_X(x)P_Y(y)}\right) \tag{2.23}$$

Making use of Eq. 2.22, MI can be expressed as the difference between the entropy of Y and the conditional entropy of Y given X:

$$\begin{aligned}
I(X,Y) &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log\left(\frac{P_{(X,Y)}(x,y)}{P_X(x)P_Y(y)}\right) \\
&= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log\left(\frac{P_{(X,Y)}(x,y)}{P_X(x)}\right) - \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log\left(P_Y(y)\right) \\
&= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_X(x)P_{Y|X=x}(y)\log\left(P_{Y|X=x}(y)\right) - \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log\left(P_Y(y)\right) \\
&= \sum_{x\in\mathcal{X}} P_X(x)\left(\sum_{y\in\mathcal{Y}} P_{Y|X=x}(y)\log\left(P_{Y|X=x}(y)\right)\right) - \sum_{y\in\mathcal{Y}}\left(\sum_{x\in\mathcal{X}} P_{(X,Y)}(x,y)\right)\log\left(P_Y(y)\right) \\
&= -\sum_{x\in\mathcal{X}} P_X(x)H(Y|X=x) - \sum_{y\in\mathcal{Y}} P_Y(y)\log\left(P_Y(y)\right) \\
&= -H(Y|X) + H(Y) = H(Y) - H(Y|X).
\end{aligned} \tag{2.24}$$

From Eq. 2.24, an intuitive understanding of the MI is as follows. When the dependency between the two random variables is low, which indicates that the information contained in X and Y are more "different," the amount of information conveyed from Y given the value of X is high, and subsequently $H(Y|X)$ is high. For a fixed value of $H(Y)$, the MI between X and Y is low. In contrast, if the dependency between the two random variables is high, the amount of information conveyed from Y given the value of X is low, and therefore

the MI between X and Y is high. There are two important characteristics of the MI: (1) Non-negativity, i.e., $I(X,Y) \geqslant 0.0$; and (2) Symmetry, i.e., $I(X,Y) = I(Y,X)$.

Similar to the conditional entropy, the *conditional mutual information* of random variables X & Y, given the random variable Z, is as follows:

$$I((X,Y)|Z) = \sum_{z\in\mathcal{Z}}\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} P_{(X,Y,Z)}(x,y,z)\log\left(\frac{P_Z(z)P_{X,Y,Z}(x,y,z)}{P_{(X,Z)}(x,z)P_{(Y,Z)}(y,z)}\right), \tag{2.25}$$

where, $\mathcal{Z}$ is the sample space of Z, $P_{(X,Y,Z)}(x,y,z)$ is the joint probability distribution between random variables X, Y & Z, and $P_{(X,Z)}(x,z)$ & $P_{(Y,Z)}(y,z)$ are the joint probability distributions between random variables X & Z and Y & Z, respectively.

2.3.3 Directed Information

Given the two sequences $X_{1:T}$ and $Y_{1:T}$, where, T is the length "time" of observation, the DI is defined [73] as follows:

$$\begin{aligned} I(X_{1:T}\to Y_{1:T}) &= \sum_{t=1}^{T} I\left((X_{1:t},Y_t)\,|Y_{1:t-1}\right) \\ &= H(Y_{1:T}) - \sum_{t=1}^{T} H\left(Y_t|\,(X_{1:t},Y_{1:t-1})\right). \end{aligned} \tag{2.26}$$

An intuitive interpretation (similar to the one for MI), that the DI is an important metric for quantifying the causal relationship, is as follows. When Y_t has little dependence on $X_{1:t}$, then the amount of new information obtained from Y at time step t given the sequence of $X_{1:t}$ is high. Subsequently, the conditional entropy $H\left(Y_t|\,(X_{1:t},Y_{1:t-1})\right)$ is high. If $H(Y_{1:T})$ is fixed, the DI of $X_{1:T}\to Y_{1:T}$ would be low since it is the difference between $H(Y_{1:T})$ and the sum of the conditional entropy, $H\left(Y_t|\,(X_{1:t},Y_{1:t-1})\right)$, over all time steps. Similar deduction could be made for the converse case, i.e., when Y_t has high dependence on $X_{1:t}$. One important distinction of DI from MI is that the DI is asymmetric, i.e., in general, $I(X_{1:T}\to Y_{1:T}) \neq I(Y_{1:T}\to X_{1:T})$ as illustrated by the example in the following subsection.

2.3.4 DI Example

A simple example is presented in this subsection to illustrate the idea of DI. Assume for a certain city, the occurrence of snowing ($X_t = 1$) each day follows a Bernoulli distribution, with probability of snowing as 0.5 (i.e., $X_t \sim Ber(0.5)$). If it snowed in the previous day, the ground will freeze (i.e., $Y_t = 1$ if $X_{t-1} = 1$), where the ground freezing condition is represented by the random variable Y. Therefore, the process could be expressed as a causal relationship ($X_{t-1} = 1 \to Y_t = 1$ & $X_{t-1} = 0 \to Y_t = 0$, where, 0 & 1 represent that the corresponding events will "not occur" & will "occur", respectively). After observing the

status of snowing and freezing for T days, we could obtain two sequences $X_{1:T}$ and $Y_{1:T}$. Because of the simple setting in this example, analytical solutions of DI can be obtained for $I(X_{1:T} \to Y_{1:T})$ and $I(Y_{1:T} \to X_{1:T})$.

First, calculate $I(X_{1:T} \to Y_{1:T})$ as follows:

$$
\begin{aligned}
I(X_{1:T} \to Y_{1:T}) &= \sum_{t=1}^{T} I\left((X_{1:t}, Y_t) \,|Y_{1:t-1}\right) \\
&= \sum_{t=1}^{T} \sum_{X_{1:t}} \sum_{Y_t} \sum_{Y_{1:t-1}} P_{(X_{1:t},Y_t,Y_{1:t-1})} \log\left(\frac{P_{Y_{1:t-1}}\, P_{(X_{1:t},Y_t,Y_{1:t-1})}}{P_{(X_{1:t},Y_{1:t-1})}\, P_{(Y_t,Y_{1:t-1})}}\right) \\
&= \sum_{t=1}^{T} \sum_{X_{1:t},Y_{1:t}} P_{(X_{1:t},Y_{1:t})} \log\left(\frac{P_{Y_{1:t-1}}\, P_{(X_{1:t},Y_t,Y_{1:t-1})}}{P_{(X_{1:t},Y_{1:t-1})}\, P_{(Y_t,Y_{1:t-1})}}\right) \\
&= \sum_{t=1}^{T} \sum_{X_{1:t},Y_{1:t}} P_{(X_{1:t},Y_{1:t})} \log\left(\frac{P_{(Y_t|(X_{1:t},Y_{1:t-1}))}}{P_{Y_t|Y_{1:t-1}}}\right) \\
&= \sum_{t=2}^{T} \sum_{X_{1:t},Y_{1:t}} P_{(X_{1:t},Y_{1:t})} \log\left(\frac{1}{0.5}\right) \\
&= \sum_{t=2}^{T} \log\left(\frac{1}{0.5}\right) = \sum_{t=2}^{T}(1) = T-1.
\end{aligned}
\tag{2.27}
$$

In the above derivation, the first four lines respectively correspond to the definition of DI, the definition of the conditional MI, the reorganization of random variables, and the definition of the conditional probability. The fifth line corresponds to the calculation of probabilities, where, $P_{(Y_t|(X_{1:t},Y_{1:t-1}))} = 1$ because the value of Y_t is certain given X_t and $P_{Y_t|Y_{1:t-1}} = 0.5$ because Y_t is independent of $Y_{1:t-1}$, i.e., $P_{Y_t|Y_{1:t-1}} = P_{Y_t} = P_{X_{t-1}} = Ber(0.5)$. Note that t starts at 2 in the fifth line, because Y_{t-1} does not exist for $t = 1$ as $t - 1 = 0$. The sixth line holds making use of the definition of the expectation of a constant, which is just the constant itself.

Second, calculate $I(Y_{1:T} \to X_{1:T})$ as follows:

$$\begin{aligned}
I(Y_{1:T} \to X_{1:T}) &= \sum_{t=1}^{T} I\left((Y_{1:t}, X_t)\,|X_{1:t-1}\right) \\
&= \sum_{t=1}^{T} \sum_{Y_{1:t}} \sum_{X_t} \sum_{X_{1:t-1}} P_{(Y_{1:t},X_t,X_{1:t-1})} \log\left(\frac{P_{X_{1:t-1}}\, P_{(Y_{1:t},X_t,X_{1:t-1})}}{P_{(Y_{1:t},X_{1:t-1})}\, P_{(X_t,X_{1:t-1})}}\right) \\
&= \sum_{t=1}^{T} \sum_{Y_{1:t},X_{1:t}} P_{(Y_{1:t},X_{1:t})} \log\left(\frac{P_{X_{1:t-1}}\, P_{(Y_{1:t},X_t,X_{1:t-1})}}{P_{(Y_{1:t},X_{1:t-1})}\, P_{(X_t,X_{1:t-1})}}\right) \\
&= \sum_{t=1}^{T} \sum_{Y_{1:t},X_{1:t}} P_{(Y_{1:t},X_{1:t})} \log\left(\frac{P_{(X_t|(Y_{1:t},X_{1:t-1}))}}{P_{X_t|X_{1:t-1}}}\right) \\
&= \sum_{t=2}^{T} \sum_{Y_{1:t},X_{1:t}} P_{(Y_{1:t},X_{1:t})} \log\left(\frac{0.5}{0.5}\right) \\
&= \sum_{t=2}^{T} \log\left(\frac{0.5}{0.5}\right) = \sum_{t=2}^{T}(0) = 0.
\end{aligned} \tag{2.28}$$

The above derivation is similar to the one in Eq. 2.27 and analogous explanations apply. Two observations could be drawn from this example:

1. The DI is, in general, asymmetric, i.e., $I(X_{1:T} \to Y_{1:T}) \neq I(Y_{1:T} \to X_{1:T})$. Even though $X_{1:T}$ is a relevant sequence for $Y_{1:T}$, $X_{1:T}$ is not determined by $Y_{1:T}$. Therefore, DI of $Y_{1:T} \to X_{1:T}$ is zero, i.e., there is no causal relationship. This demonstrates the ability of the DI to quantify causal relationships; and

2. Even for such a simple example, the theoretical derivation of the DI is cumbersome and can be very difficult. Therefore, in real practices, the analytical solutions are not obtained. Instead, methods for estimating the DI have been proposed by several researchers [46]. In this dissertation, estimations of the DI are used. More details are given in Chapter 7.

A final remark about the DI is related to the final result of $I(X_{1:T} \to Y_{1:T})$ in Eq. 2.27. The value is clearly non-decreasing as the length of observation T increases. In that case, the DI depends on T, and it poses a challenge for comparing values of different causal relationships. Therefore, the DI is divided (normalized) by the length of observation, and such value is used. Thus, the averaged value of the DI, from Eq. 2.27, is as follows:

$$\frac{1}{T} I\left(X_{1:T} \to Y_{1:T}\right) = \frac{T-1}{T} = 1 - \frac{1}{T} \to 1 \quad \text{as} \quad T \to \infty. \tag{2.29}$$

Such averaged value can be used to comparable several causal relationships. In this dissertation, unless otherwise specified, such averaged value over a long range of observation T is reported and used in the proposed OSP algorithm.

Chapter 3

Modeling and Simulation of Example Structures

In this chapter, two models based on the Finite Element Method (FEM) are described in detail. These simulated results are treated as virtual experimental data for validating the algorithms proposed in the following chapters.

3.1 Planar Reinforced Concrete Moment Frame

A model of a 3-story planar (i.e., two-dimensional (2-D) in the $X - Y$ plane where the axes X, Y & Z are shown in Fig. 3.1) Reinforced Concrete (RC) moment resisting frame is developed using the FEM. The commercial software DIANA (DIsplacement ANAlyser, Version 10.3 [29]) is used to develop the model. This structural system is commonly used as a lateral load resisting system for earthquake-resistant design [67]. The height of each story is 14 ft (4.27 m), and the bay length is 24 ft (7.32 m). The cross-sections of all columns (in the $Z - X$ plane) are square with side length of 2 ft (610 mm). For simplicity and irrespective of being interior or exterior and being in the first, second, or third floor, each column has 14 reinforcing bars (rebars) (6 on each side layer to mainly resist the bending moment about the Z axis, i.e., M_Z, and 2 in the middle layer), where each rebar has a cross-sectional area of 1 in^2 (6.45 cm^2, U.S. rebar #9). The cross-sections of all beams (in the $Y - Z$ plane) are rectangular with the same dimensions (width of 2 ft (610 mm) and depth of 3.5 ft (1,067 mm)). Each beam has 22 rebars (8 on the top and bottom layers to mainly resist M_Z, and 6 in a single middle layer), where, as for the columns, each rebar has a cross-sectional area of 1 in^2 (6.45 cm^2). Refer to Fig. 3.2 for the typical cross-sections of the columns and beams. It should be noted that the layout of rebars here is not conventional from a design and construction point of view, but it is used in this 2-D model for simplicity. The closed stirrups (for the beams) and ties (for the columns) have rebar cross-sectional area of 0.31 in^2 (2.00 cm^2, U.S. rebar #5) and center-to-center spacing of 6 in (127 mm). The material properties for concrete and steel reinforcements are specified in Tables A.1 and

A.2, respectively. It is assumed that a single set of concrete material properties is used for the core and cover concrete elements[1]. For the dynamic response, Rayleigh damping [23] is applied in the model with coefficients determined using the natural frequencies calculated as discussed below in this section. The damping ratio for the first two modes is taken as 5%, which is a common value for such RC frames.

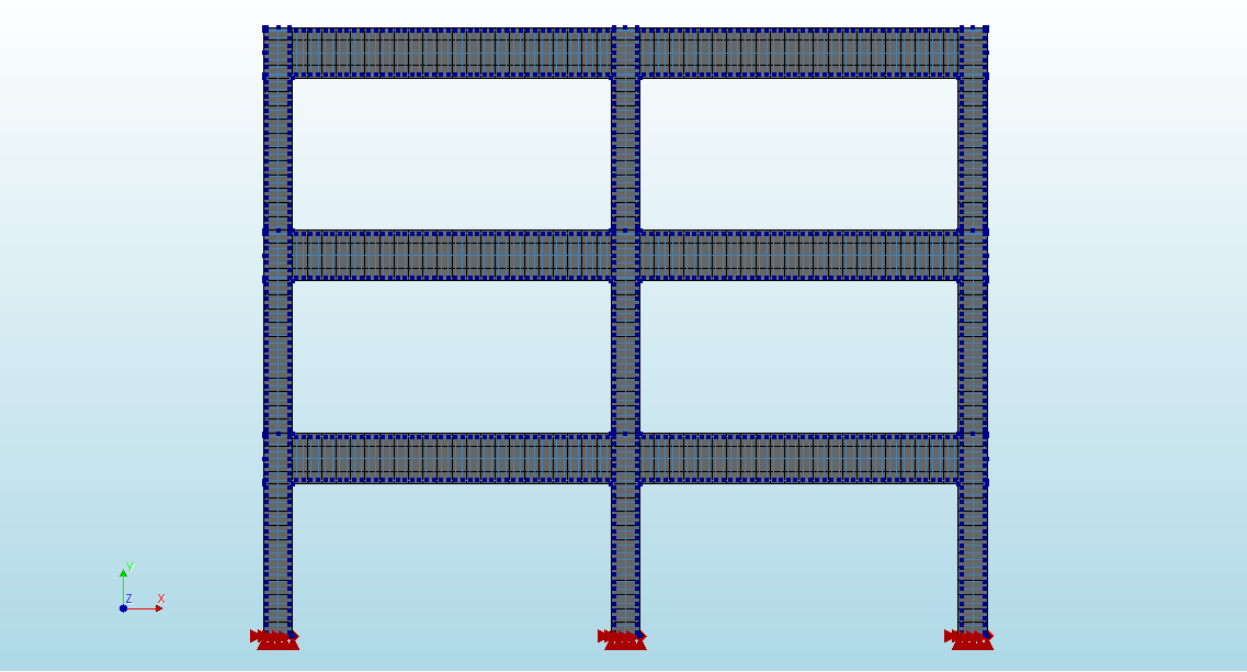


Figure 3.1: FEM model of the three-story, two-bay RC frame model using DIANA [29].

In terms of the considered seismic weights, superimposed dead load is added to the dead weight, which is the weight of RC structure itself. The live load is not considered in the seismic weight, as it does not generally contribute to the horizontal inertia forces. Moreover, for the floor tributary area of the considered framed structure, it is assumed that the total number of bays (also with bay length of 24 ft (7.32 m)) in the out-of-plane direction is four, and two such identical frames are designed. Therefore, each frame supports the horizontal seismic weight of 2 out-of-plane bays (i.e., 4 bays divided by 2 frames). Therefore, each frame supports a floor tributary area for the seismic weight made of 48 ft (14.63 m, 2 bays

[1]In the column section, the core is in the section interior, and the cover is on the section periphery. It is a common practice to model the effect of reinforcement on confinement of the concrete core regions. For simplicity, the core and cover concrete are not distinguished herein. This is deemed to be justified, because the main purpose of this FEM model is to test the applicability of the LSTM model proposed in this dissertation to identify "undamaged" versus "damaged" states of structures. So long as an "undamaged" state is defined (even though the model is simplified), this FEM model is capable of achieving the intended purpose of the proposed model

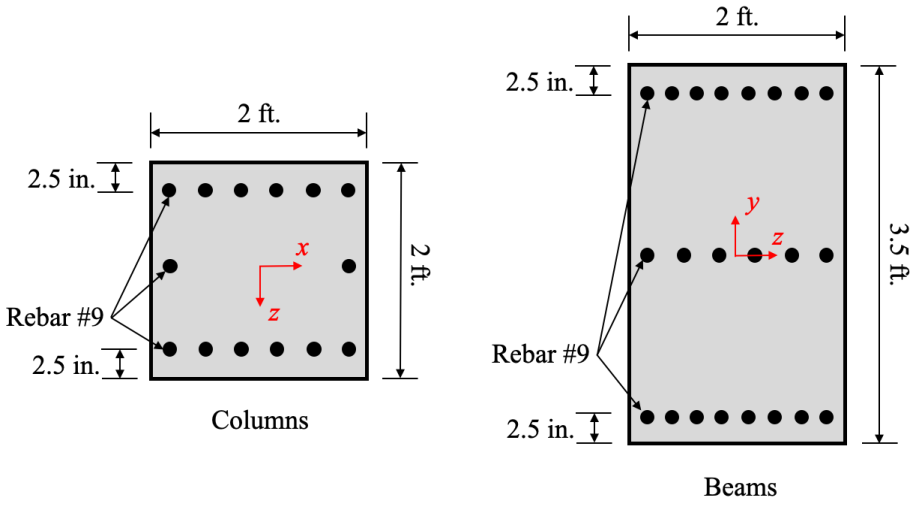


Figure 3.2: Cross-sections of the RC frame model columns (left) and beams (right).

× 24 ft (7.32 m) for each bay) in the out-of-plane direction. The assumed superimposed dead loads for the second & third floors and for the roof are shown in Tables 3.1 and 3.2, respectively. These values are based on estimations from real project design. In addition to these so-called flat loads, vertical cladding loads are added, which are assumed to be 15 psf (0.72 kPa or kN/m^2). Note that the partitions are not over the entire area, so the considered seismic weight is intuitively taken as half the real weight. Therefore, the partitions items in Tables 3.1 and 3.2 are divided by 2. Similarly, for the mechanical equipment on the roof, the load is divided by 2. Note that NWC and 18 ga W2 in Tables 3.1 and 3.2 refer to Normal Weight Concrete and W2 composite floor deck form made of 18 gauge steel, respectively. The allowance for additional fill considers approximately 10% weight of the corresponding concrete fill.

Table 3.1: Superimposed dead flat load for the 2nd & 3rd floors of the RC framed structure.

Items	**Description**	**Load (psf)**
Concrete fill	4.5 in. NWC	69
Steel deck	18 ga W2	3
Allowance for additional fill	10% of fill	7
Ceiling, mech., misc.	Includes fireproofing	14
Partitions		20/2 = 10
Total		103 (4.93 kPa)

Based on the above information, the total superimposed dead load for the second and

Table 3.2: Superimposed dead flat load for the roof of the RC framed structure.

Items	**Description**	**Load (psf)**
Concrete fill	3.5 in. NWC	56
Steel deck	18 ga W2	3
Allowance for additional fill	10% of fill	6
Ceiling, mech, misc.	Includes fireproofing	14
Mech. equip. (misc.)		20/2 = 10
Total		89 (4.26 kPa)

third floors per unit length of the frame (in-plane direction) is obtained as follows:

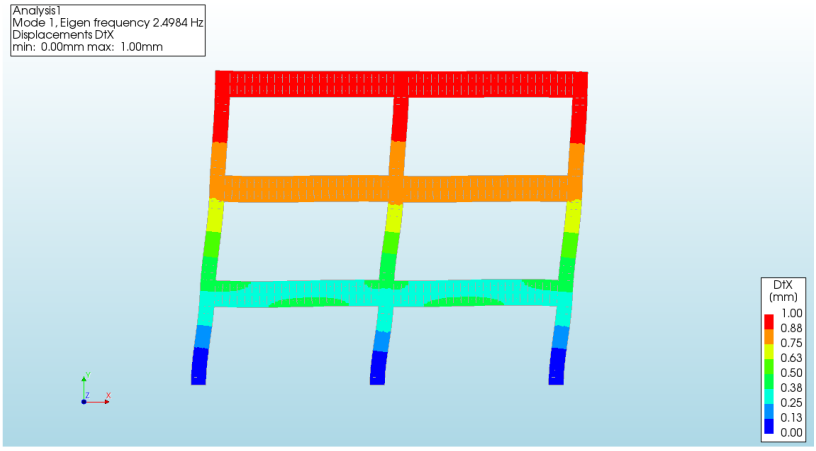
$$103 \text{ psf} \times 48 \text{ ft} + 15 \text{ psf} \times 14 \text{ ft} = 5,154 \text{ lb/ft } (75.22 \text{ kN/m}),$$

where, 14 ft (4.27 m) is the height of each story and also that of the vertical cladding. The reason that the weight of the cladding is multiplied by the height of the story is that the cladding system is vertical and it is assumed to be suspended from the beams. The total superimposed dead load for the roof per unit length of the model is as follows:

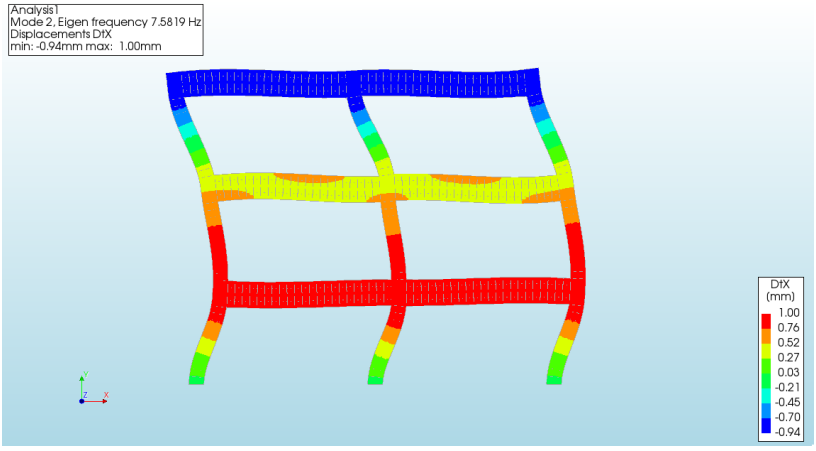
$$89 \text{ psf} \times 48 \text{ ft} + 15 \text{ psf} \times 14 \text{ ft} = 4,482 \text{ lb/ft } (65.41 \text{ kN/m}).$$

The model is first analyzed using the *eigen solver*. The calculated natural periods are 0.400, 0.132, and 0.082 sec. for the first three modes (the corresponding natural frequencies are 2.498, 7.582, and 12.256 Hz, respectively), refer to Fig. 3.3. From the ASCE 7-16 Eq. (12.8-8) [24], the estimation of the first mode natural period for structures not exceeding 12 stories above the base, where the seismic force-resisting system consists entirely of concrete or steel moment-resisting frames and the average story height is at least 10 ft (3.05 m), is equal to $0.1N$, where N is the number of stories. For the case here, the estimation of the natural period is 0.3 sec. Therefore, the computed natural period using the FEM for the three-story RC frame model is about 33% higher than the ASCE code approximate estimate [24], which is deemed acceptable. The approximate mode shape vector for the first mode of the frame (viewed as a three Degree of Freedom (DOF) system of horizontal displacements at the three floor levels) is $[0.370, 0.773, 1.000]$.

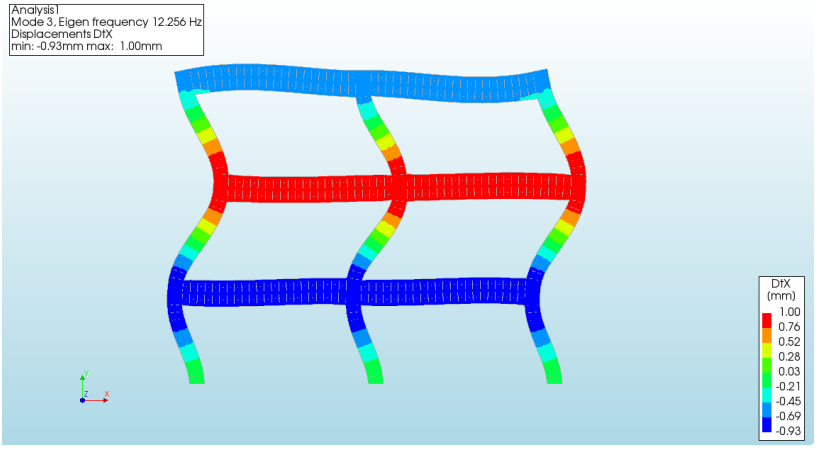
In order to simulate the response of the RC framed structure under earthquakes, the bases of the first (ground) story columns are fixed to the ground, and uniform Ground Motion (GM) excitation is applied at all the column bases. The applied GM records are selected and scaled from the Pacific Earthquake Engineering Research (PEER) Center Next Generation Attenuation (NGA)-West2 Database [91]. Appendix B lists all the applied GM records in the analyses of the RC framed structure. After the uniform excitation is applied, horizontal accelerations in the middle of each floor are recorded. In order to simulate the real sampling capacity from the real world accelerometers, the sampling frequency is selected as 400 Hz, which means that the time step size is taken as $1/400 = 0.0025$ sec. More details of this RC



(a) Fist mode shape ($f_1 \approx 2.5$ Hz)



(b) Second mode shape ($f_2 \approx 7.6$ Hz)



(c) Third mode shape ($f_3 \approx 12.3$ Hz)

Figure 3.3: First three mode shapes of the three-story RC frame using the FEM.

frame model are included in Appendix A. It should be noted that, even though nonlinear material properties are specified (as in Tables A.1 and A.2, respectively, for concrete and steel reinforcements), the structural responses are within the elastic range, because the intensity of the applied GM is somewhat low.

3.2 Space Concentrically Braced Steel Frame

A model of a 3-story, 3-bay space (i.e., three-dimensional (3-D)) Concentrically Braced Steel (CBS) (tension only) frame is established using the FEM. It is modeled after a full-scale shaking table experiment conducted in Tongji University, China, Fig. 3.4. The original test structure has a total height of 10.02 m (3.34 m for each story). The structure is designed per Code for Seismic Design of Buildings of China (CSDBC [43]). The main horizontal force-resisting system is represented by the diagonal braces, which have varying cross-sections and material properties for different stories. In the X-direction, tension-only diagonal braces are installed in the interior frames, while in the Y-direction, tension-only diagonal braces are installed in the exterior frames. In the shaking table experiment, the slenderness of the diagonal braces ensured that they were tension-only, and their compressive strength was negligible. From the shaking table experiments, the main damage is caused by the yielding or loosening of the diagonal braces after the earthquake loading. Fig. 3.4 shows the design views and a photograph of the test structure, and Fig. 3.5 shows the yielding and loosening of the diagonal braces. More information about this experiment can be found in [19].

The developed FEM model simulated the responses of the original structure under different GM load cases with varying intensities, where both linear and nonlinear responses could be simulated. In the case of nonlinear responses, the propagation of damage could also be simulated. The computational modeling of the test structure is conducted using *OpenSeesPy* [87], a widely adopted system for earthquake engineering simulations with a Python 3 interpreter. The model is shown in Fig. 3.6. The sizes of the whole structure and the components followed exactly the experiment. The corresponding weights, using the estimated values from the experiment, are applied as floor seismic masses and concentrated nodal gravity forces, assuming uniform distribution over the floors. The nodes at the base are fixed. Since the main damage came from the braces, the beams and columns are modeled elastically using *elasticBeamColumn* elements [87], while the braces are modeled using *ElasticPPGap* material[2] [87], Fig. 3.7, to model the yielding and loosening of the braces on the tension side.

[2]The *ElasticPPGap* (Elastic-Perfectly Plastic Gap) material is used to model the tension-only braces. After experiencing inelastic elongation and unloading in one loading cycle, the braces have "permanent" residual elongation. Such elongation causes the braces to loosen, and both the modulus of elasticity and the stress drop to zero. After the braces are loaded again in the next cycle, both the modulus of elasticity and the stress become non-zero only if the new elongation is larger than the maximum elongation in the previous cycles, which is approximately the maximum inelastic residual elongation.

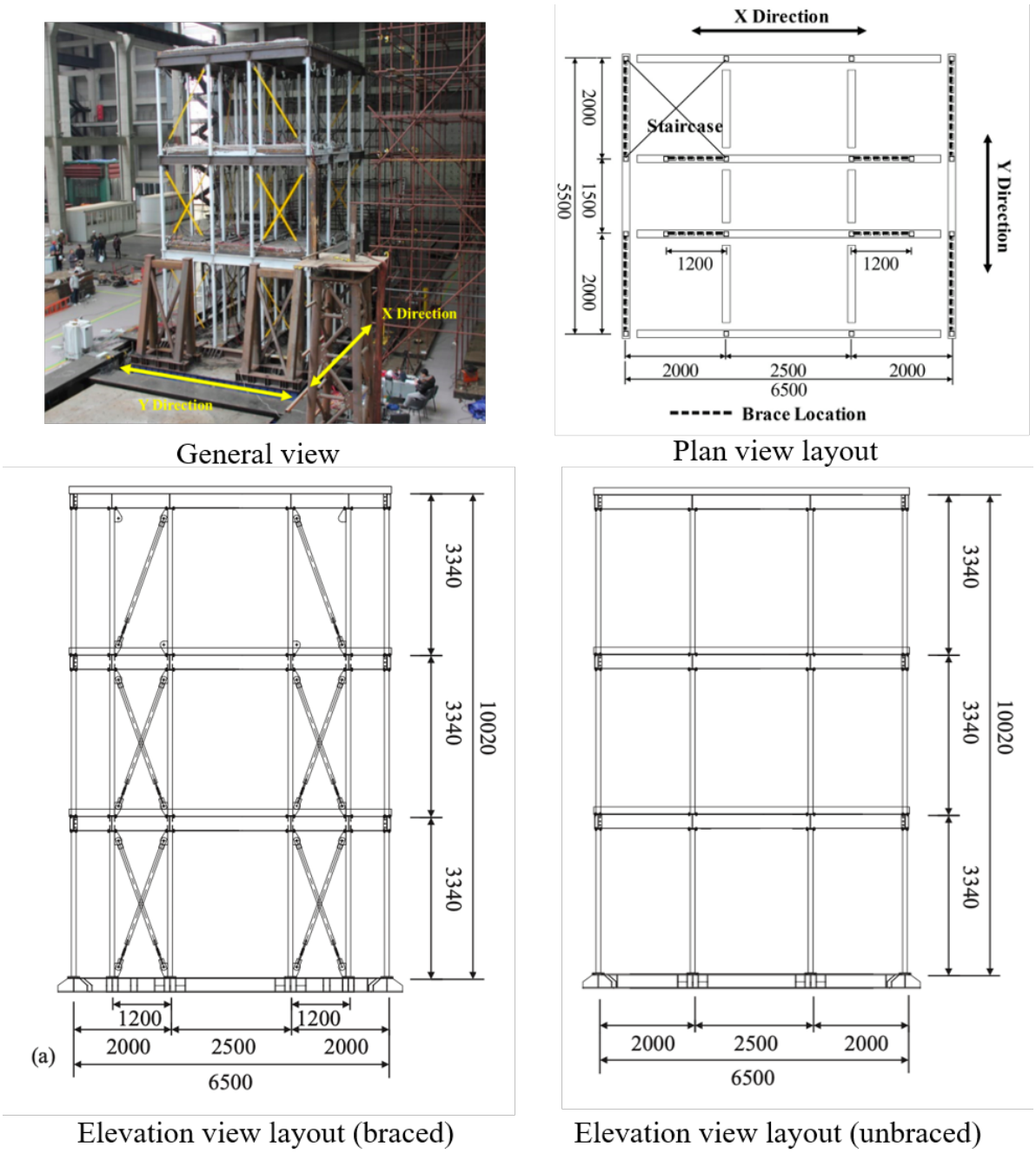


Figure 3.4: CBS frame for the shaking table tests (unit: mm) [19].

The sectional and material properties reasonably followed those in the experiment [19], within the limitations and options of OpenSeesPy [87]. Model optimization is conducted to adjust a subset of such properties to reduce the difference of several key structural response

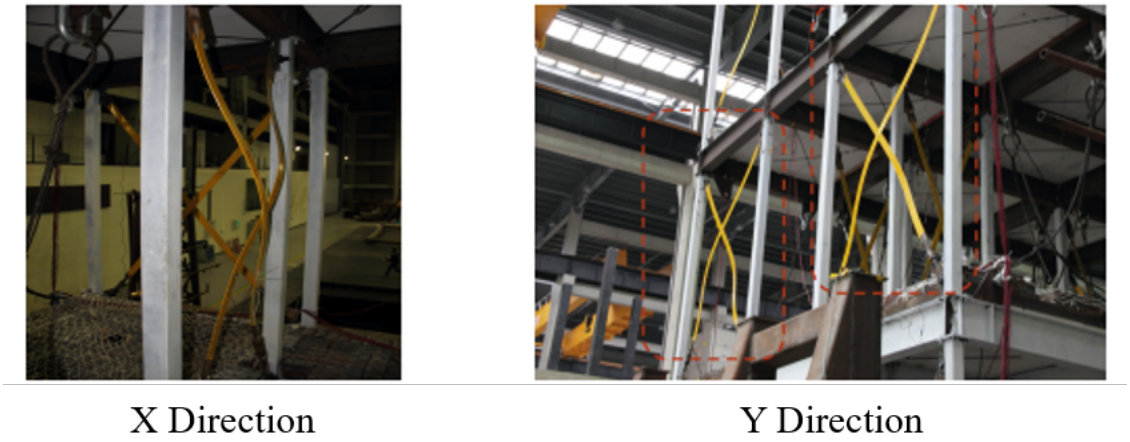


Figure 3.5: Diagonal braces yielding and loosening in the original shaking table tests [19].

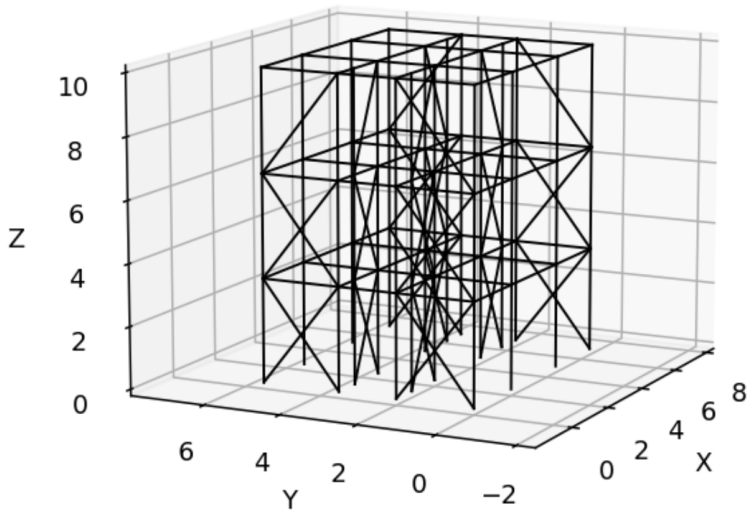


Figure 3.6: FEM model of the CBS frame using OpenSeesPy [87].

parameters between the experiment and the FEM model. Details and results of the model optimization approach is described in the next Section 3.3.

In order to simulate the response of the braced steel frame under earthquakes, similar to the RC frame model, discussed in Section 3.1, the bases of first (ground) story columns are fixed in the ground, and uniform GM excitation is applied at all the column bases. Unconditional Selection (US) method [65], a spectrum shape matching GM selection and modification method, is used to select and scale the GM's from the PEER NGA-West2

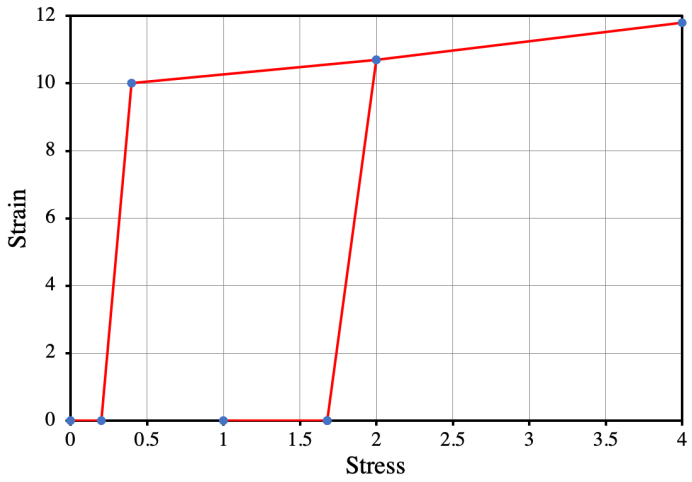


Figure 3.7: Stress-strain relationship of ElasticPPGap material [87].

Database [91]. Appendix B lists all the applied GM's in the analyses of the braced steel frame structure. These GM's are first applied at the base, then the two horizontal components for each GM are swapped and applied at the base again as a separate case. Nonlinear transient analyses are conducted by adopting *Newton-Raphson iterative solver* [87] (convergence norm set for displacement at 10^{-12}) and *Newmark β time integrator* ($\gamma = 0.5$ and $\beta = 0.25$) [23]. It is expected that higher GM intensities are likely to cause the loosening of more braces, and subsequently causing more severe floor damage.

In the experiments, sensors (including accelerometers, displacement transducers, and strain gauges) were installed. The accelerometers were installed at the center and two corners of every floor to measure the response in the two horizontal directions. Fig. 3.8 shows the accelerometer locations. In the FEM model, accelerations at the nodes as well as the center of each floor are recorded. The sampling frequency for the analyses was selected as 100 Hz, i.e., the time step size is taken as $1/100 = 0.01$ sec.

3.3 Genetic Algorithm & Model Optimization

As mentioned above, a Genetic Algorithm (GA) is adopted to optimize the computational model of the braced steel frame, discussed in Section 3.2. In particular, a subset of structural properties (e.g., sectional area, moment of inertia, and material yielding stress) are tuned to reduce the difference of several key structural response parameters (natural period and maximum drift ratio) between the experiment and the FEM model. The set of the structural response parameters and the structural properties are summarized in Tables 3.3 and 3.4, respectively. In order to formulate the problem as an *optimization problem*, a target

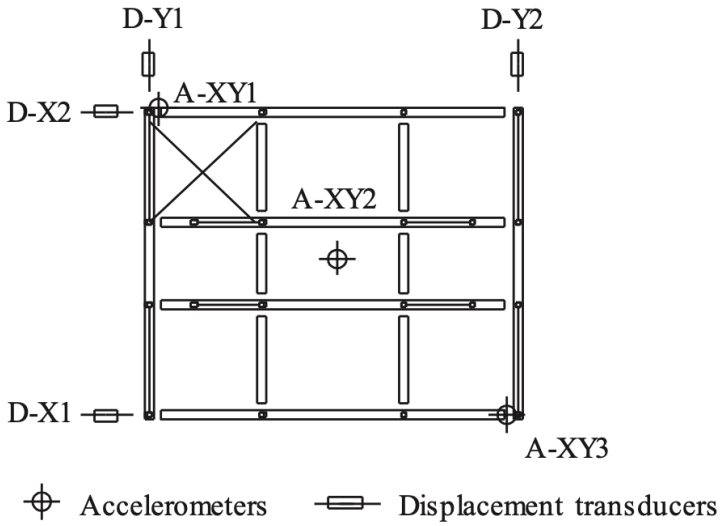


Figure 3.8: Accelerometers and displacement transducers in the shaking table tests [19].

(objective or cost) function is defined as follows:

$$f(u_{\text{real},i}, u_{\text{model},i}) = \sum_i w_i \frac{|u_{\text{real},i} - u_{\text{model},i}|}{u_{\text{real},i}}, \tag{3.1}$$

where $u_{\text{real},i}$ and $u_{\text{model},i}$ are the natural period and maximum drift ratio obtained from the experiment measurements (i.e., target values) and the FEM model simulations, respectively, and w_i are the weight parameters, used to adjust the relative importance of the structural response parameters in the target function. The i is the index of structural response parameters as listed in Table 3.3. The natural period is, in general, regarded as a more important response parameter for the preliminary design of structures, and its weight is taken higher than that of the maximum drift. The differences between $u_{\text{real},i}$ and $u_{\text{model},i}$ are normalized by $u_{\text{real},i}$ to reduce the effect of parameter value on the relative weights (e.g., if the value of the natural period is larger than that of the drift ratio, without normalization, the contribution from the natural period on the target function would be artificially higher than that of the drift ratio). It is noted that the natural period and maximum drift radio are adopted here because these values were clearly recorded in the experiment before and after loading. In the shaking table tests, White Noise (WN) scanning was conducted to measure the natural period in two directions (X & Y). The natural periods obtained from the WN was larger after GM application with higher intensities, as more braces were loosened and the structure became more flexible. Maximum drift values were recorded using the displacement transducers. In terms of GM cases in the experiment, two load cases are selected for matching between the experiments and the simulations: (1) GM from 1940 California Imperial Valley earthquake recorded at El Centro, scaled such that the Peak Ground Acceleration

(PGA) are $0.035g$ and $0.03g$ (where g is the acceleration of gravity), in X and Y directions, respectively, and (2) GM from 1995 Kobe earthquake (Kobe), scaled such that the PGA is $0.4g$ in X-direction. The structural response parameters, and the corresponding weights, are summarized in Table 3.3.

The optimization problem is formulated as follows:

$$\min_{\theta} f(u_{\text{real},i}, u_{\text{model},i}), \tag{3.2}$$

where θ is a set of structural properties to be tuned. The list of θ is shown in Table 3.4.

Two notes about the structural properties:

1. In the setting of the GA, a set of choices should be provided for each property. In this study, the choices are quantified as the ratio (scale) between the property values of choice and the approximate values estimated from the experiment. Such scaling factors, rather than the true property values, are used as the true θ to be tuned.

2. The structural properties in Table 3.4 related to braces indeed represent a set of these properties. Only a single scale factor is used within each set. For example, the sectional area of braces in the third row represents a set of sectional areas for different floors and directions (while in the experiment, different sectional areas are designed). However, a single scale factor is used within the set, i.e., the sectional areas of braces in the different floors and the two horizontal directions have the same scale factor. In this way, the optimization efficiency is improved. Otherwise, the number of properties to be tuned is prohibitively large. On the other hand, among different rows in Table 3.4, e.g., sectional area of braces and damping ratio of the whole structure, different scale factors are used. Therefore, in total, only 6 scale factors are tuned, corresponding to the 6 rows in Table 3.4.

As mentioned above, a (non-classical) GA [77] is used to optimize the target function, according to the minimization shown in Eq. 3.2. It is adopted here as an efficient alternative to full grid search (i.e., exploration of all combinations of the structural properties, [105]). The GA mimics the process of "natural selection" to generate high-quality solutions to optimization problems. Mutation, crossover and selection operators are adopted in GA to mimic the process of evolutionary algorithms. The reason that the GA adopted here is non-classical is that the choice of selection for each variable is not binary, i.e., more than two choices are available (the classical GA mimics the bits in genetic sequence, and such bits are binary). The full algorithm is as follows:

- Initialize population: Generate a set of hypotheses P_0, where 0 indicates the 0^{th} generation. The set size is unchanged in the following iterations.

- Iterate over t, where t indicates the t^{th} generation, as follows:

  - Evaluate the target function for all hypotheses that belong to P_t.

Table 3.3: Set of structural response parameters (u).

Index	Response Parameter	Loading	Direction	Location (floor)	Weight
1	Natural Period (sec)	Before Loading	X		0.2
2		Before Loading	Y	-	0.2
3		After El Centro GM	X	-	0.1
4		After El Centro GM	Y	-	0.1
5		After Kobe GM	X	-	0.1
6		After Kobe GM	Y	-	0.1
7	Maximum Drift Ratio	After El Centro GM	X	1	0.05
8		After El Centro GM	X	2	0.05
9		After El Centro GM	X	3	0.05
10		After El Centro GM	Y	1	0.05
11		After El Centro GM	Y	2	0.05
12		After El Centro GM	Y	3	0.05
13		After Kobe GM	X	1	0.05
14		After Kobe GM	X	2	0.05
15		After Kobe GM	X	3	0.05

Table 3.4: Set of structural properties (θ) to be tuned.

Component(s)	Property
Columns & Beams	Sectional Area
Columns & Beams	Sectional Moment of Inertia
Braces	Sectional Area
Braces	Yielding Stress
Braces	Hardening Ratio
Whole Structure	Damping Ratio

- Selection: Select a subset of individuals that have better fitness function values. Note that each individual could be selected more than once.
- Crossover: Probabilistically select pairs of hypotheses from P_t as parents, and produce offsprings by copying a subset of bits from one parent, and the complementary subset of bits from another parent. The process of crossover is demonstrated in Fig. 3.9.
- Mutation: Probabilistically select a small set of bits and change their values to other options. In that case, the next generation of population P_{t+1} is generated. The process of mutation is demonstrated in Fig. 3.9.

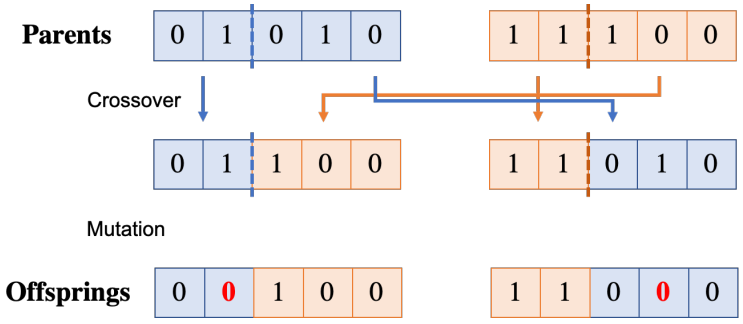


Figure 3.9: Demonstration of the crossover and the mutation operations in the GA.

The iterations in the GA above could stop when the highest fitness value of individuals in the population stopped improving over the generations, or the maximum number of iterations is reached. In our case, the maximum number of iterations is set as 20, and the individual with the lowest target function value over all generations is selected as the final optimization result. The target function value (i.e., the value calculated using Eq. 3.1, with weights in Table 3.3 and response parameters from the simulations before implementing the GA algorithm) is found to be 0.3084. After implementing the algorithm, the target function value of the best individual is found to be 0.1605. Therefore, the target function value

Table 3.5: Comparison of structural response parameters from the shaking table tests ($u_{\text{real},i}$) and the tuned FEM model after optimization ($u_{\text{model},i}$).

Index	**Response Parameter**	$u_{\textbf{real},i}$	$u_{\textbf{model},i}$
1	Natural Period (sec)	0.56	0.56
2		0.53	0.48
3		0.56	0.56
4		0.53	0.48
5		1.14	1.27
6		1.52	1.27
7	Maximum Drift Ratio	0.0011	0.0010
8		0.0013	0.0010
9		0.0016	0.0008
10		0.0007	0.0010
11		0.0007	0.0008
12		0.0009	0.0008
13		0.0133	0.0114
14		0.0184	0.0152
15		0.0308	0.0226

is reduced by a half after the application of the GA algorithm. At the first glance, this reduction is not significant (especially in comparison to loss reduction in gradient descent for ML models). However, the reduction is reasonable in this case, because the initial structural properties before the algorithm come from the estimations from the experiment, i.e., the target function value is already low before the GA algorithm iterations. Furthermore, the real and model response parameters after tuning are listed in Table 3.5. For brevity, in Table 3.5, only the indices are used to identify the structural response parameters. Detailed description of these indices is found in Table 3.3. A close match between the measured values and simulated ones are observed in Table 3.5, especially when the structure is within the elastic range. Moreover, a better match of the natural period than the maximum drift ratio is observed, which is expected because of larger weights for the natural period than those for the maximum drift ratio. For completeness, the structural properties (θ) after optimization are shown in Table 3.6.

Table 3.6: Set of structural property (θ) values after optimization.

Component(s)	**Property**	**Location (floor)**	**Value**
Columns	Sectional Area (mm^2)	-	2,030
Beams	Sectional Area (mm^2)	-	2,786
Columns	Sectional Moment of Inertia (mm^4)	-	2.71×10^6
Beams	Sectional Moment of Inertia (mm^4)	-	3.88×10^7
Braces (X-direction)	Sectional Area (mm^2)	1	360
Braces (X-direction)	Sectional Area (mm^2)	2	288
Braces (X-direction)	Sectional Area (mm^2)	3	224
Braces (Y-direction)	Sectional Area (mm^2)	1	256
Braces (Y-direction)	Sectional Area (mm^2)	2	224
Braces (Y-direction)	Sectional Area (mm^2)	3	192
Braces (Thickness = 4 mm)	Yielding Stress (MPa)	-	352
Braces (Thickness = 6 mm)	Yielding Stress (MPa)	-	484
Braces (Thickness = 8 mm)	Yielding Stress (MPa)	-	447
Braces (Thickness = 4 mm)	Hardening Ratio (%)	-	0.26
Braces (Thickness = 6 mm)	Hardening Ratio (%)	-	0.25
Braces (Thickness = 8 mm)	Hardening Ratio (%)	-	0.26
Whole Structure	Damping Ratio	-	0.1

Chapter 4

Structural Health Monitoring Framework Using Long Short-Term Memory Encoder-Decoder Architecture

4.1 Introduction

In Chapter 2, the current progress of Structural Health Monitoring (SHM) and the Recurrent Neural Network (RNN) family models are discussed in detail. Irrespective of the robustness and high fidelity of the RNN family of models, few papers in the literature are addressing the use of RNN in the field of structural engineering. These few papers mainly target the structural Time Series (TS) response predictions. Zhang et al. [130] used the Long Short-Term Memory (LSTM) model to predict a building response, with Ground Motion (GM) as the input. Zhang et al. [129] used the LSTM network for a dam displacement prediction. Guo et al. [39] used the LSTM model to predict the deflection of a bridge structure. Kuyuk & Susumu [58] used the LSTM network to predict the type of earthquake. Tanvi et al. [11] also used the LSTM network to model the sequence of earthquakes and predict their future trend. Therefore, LSTM-based SHM is a promising direction to investigate, as it has the potential to improve the accuracy in structural health identification.

In this chapter, a proposed SHM framework, utilizing the LSTM Encoder-Decoder network, is introduced. Two specific design considerations, which are the *loss function* and the data processing technique, are discussed.

4.2 The Proposed Framework

Starting from this chapter, this study will mainly utilize the measured or simulated TS as input data, i.e., measurement from traditional sensor networks or nodal computations using

the Finite Element Method (FEM). The acquired data types for traditional sensors can be the dynamic response quantities (e.g., force, strain, acceleration, or displacement) or environmental quantities (e.g., temperature, humidity, wind speed, or GM). For this dissertation, the focus is on the dynamic response quantities of structures, in particular, acceleration measurements from accelerometers or computed counterparts from simulations. It is noted that the accelerometer typically converts the system's acceleration into displacement of a seismic mass, which is subsequently converted into an electrical signal to be sampled into digital data. The mechanism of such an accelerometer is shown in Fig. 4.1. Herein, the proposed SHM framework is applied to structures where either sensors are installed on critical locations or simulated models of the structures are developed with computed responses of nodal points (representing "virtual" measurements). In this data extraction process, expertise is generally required to design the layout of the sensor systems of the real experiment and also for the virtual one using the computational models needed for simulations. Optimization of the sensor layout, which can minimize the number of sensors (and thereby reduce the operation cost in the case of real experiment or field deployment for monitoring) while capturing the response, is an important topic that is discussed in Chapter 7 as an Optimal Sensor Placement (OSP) problem. In this and the following chapter, the sensor layout are assumed to be given where the measurements in each time step is expressed as a vector. Each element of this vector corresponds to a measurement from one sensor. Moreover, the measurement from the accelerometers is discrete, which is sampled at a fixed frequency, which is a key specification of the accelerometer and its data acquisition system. This is in contrast to the real response, which is continuous as a function of time. Therefore, in the context of the adopted LSTM model, the measurement at each time step t is the input of one step of the LSTM model. Therefore, the whole response of the structure is collected into a two-dimensional (2-D) tensor, where each row corresponds to the response at each time step, and each column corresponds to the response at each sensor location. Fig. 4.2 demonstrates the data collection for one step.

The proposed model is shown in Fig. 4.3, making use of an Encoder-Decoder architecture, Fig. 4.4. The LSTM Encoder network is used to encode the acceleration TS at all sensor locations into a Latent Space Vector (LSV), which is the *hidden state vector* (i.e., h_t in Fig. 2.14) at the last time step[1]. Subsequently, the LSTM Decoder network is used to decode the LSV to recover the original TS. In other words, the model is learning an identity mapping with a bottleneck[2]. The difference (e.g., absolute error or squared error) between the original TS and the decoded TS, i.e., the *loss function* is used to measure the quality of the model. Since the acceleration input of each time step is a vector, both the *temporal correlation* among

[1]As seen in Fig. 2.14, the LSTM network takes input x_t from all time steps, and output h_t for all time steps. In this proposed model, only h_t from the last time step is used, i.e., even though h_t for other time steps are calculated, they are not subsequently used. The reason that only the LSV is used is explained in detail in the next two paragraphs. In summary, the LSV is the output of the encoder, and it is the "bottleneck," or the compressed representation of the original TS. It is used to complete the real damage diagnosis tasks.

[2]Identity mapping refers to the process where the original TS is converted to itself after the encoding (compression) and decoding (recovery) operations. The bottleneck is the LSV after encoding operation.

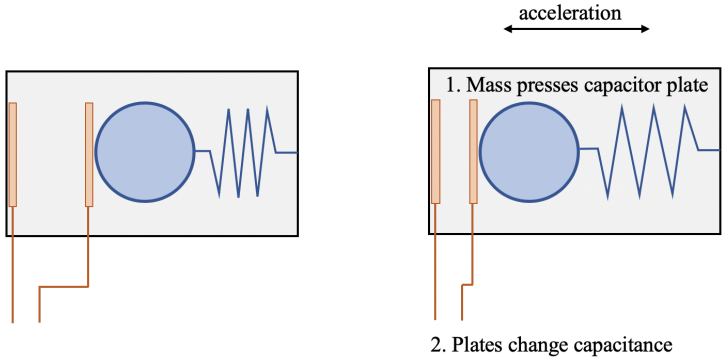


Figure 4.1: Internal mechanism of the capacitive accelerometer at rest (left) and when subjected to acceleration (right).

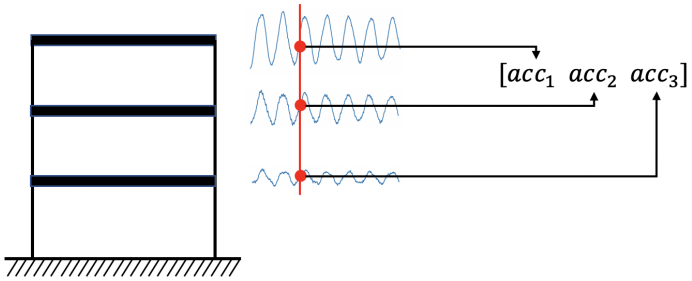


Figure 4.2: Accelerometer measurements of the structural response of a three-story frame.

time steps, which is the intention behind the RNN, and the *spatial correlation* among sensor locations are considered. This is useful for structural damage localization, as it can identify the damage within each floor from the acceleration measurements of the corresponding floor, roof, or possibly other locations.

The first LSTM network, i.e., the Encoder network, is a *many-to-one* network, because the input of each LSTM cell is the measured/calculated responses from a one time step, while the output is only the hidden state vector at the last step. The second LSTM network, i.e., the Decoder network, is a *one-to-many* network, because the input is the LSV from the Encoder network, while the output of each LSTM cell is the reconstructed TS for each time step. For the considered architecture of these LSTM networks, refer to Fig. 2.15 and its corresponding discussion in Chapter 2. Therefore, for the rest of this dissertation, the response is described per units of time steps, rather than the time units in the usual sense (e.g., seconds). Recall that for the Decoder network, the output from each LSTM cell is connected to a shallow Fully Connected (FC) Neural Network (NN), and the output of this FC NN is the final reconstructed TS. In this study, the network can be two layers, the

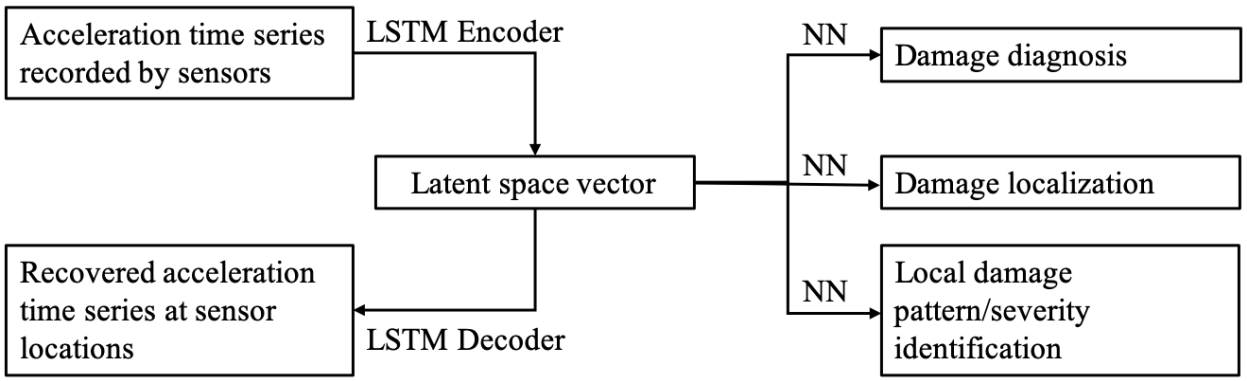


Figure 4.3: Proposed SHM framework.

Figure 4.4: An Encoder-Decoder architecture.

input layer and the output layer without hidden layers in between. The number of units in the output layer is equal to the number of sensors where the output of each unit is the reconstructed acceleration for one sensor at each time step. Since the dimension of the LSV is generally much smaller than the original TS data tensor, from the perspective of mapping from the acceleration TS to the LSV, the model is making a significant *data compression* on the original TS. Ideally, this can be a lossless compression, but in general, this compression does not correspond to zero loss. This is not a problem herein because, due to the inherent randomness of the measurement error, a completely lossless compression is not desired, as it may learn the errors inside the measurements, which causes *over-fitting* of the LSTM model.

The proposed framework is expected to compress the TS without loss of important TS information, so the learned LSV is an accurate representation of the structural response that is a function of the structural conditions. Therefore, the LSV is also a representation of the structural performance status. Moreover, since the LSV is a compressed lighter form, it is easier to manipulate, and can be applied with ordinary ML algorithms to complete the designated tasks. Accordingly, the proposed model is robust to solving various problems in SHM. In the case when a new task is to be performed, training a new LSTM network, which requires long training times as seen in the Chapter 5, is generally not required in the Encoder-Decoder architecture. Instead, the new task is trained on the LSV's, which is much faster. In other words, only one LSTM network needs to be trained for a structure in the beginning, and the subsequent manipulations are processed on the lightweight LSV's. The idea is similar to Transfer Learning [33], where only a small segment of a large network is fine-tuned. Under the belief that the rest of the network is robust, fine-tuning the small

segment ensures good performance of the whole model for specific tasks while keeping the training process efficient. In terms of the SHM tasks, the LSV is used to learn the structural damage conditions. Moreover, it is connected to several FC shallow NN for the purpose of solving different classification problems. For the overall diagnosis, the task is a *binary classification* on whether the structure has or has not experienced any damage at any location. For the local damage state detection, the task is separate binary classifications on each floor/structural element. For the local damage severity identification, the task is separate *multi-class classifications* on the damage severity for each floor/structural element. Note that the used NN in Fig. 4.3 are different from the NN used in the Decoder network in Fig. 2.15, as they serve different purposes.

The proposed model is regarded as a pre-trained model before the earthquake event. After the occurrence of one major earthquake, the measurement is fed into the model as one dataset, and the model outputs (predicts) the results within milliseconds, under the circumstances that the sensors are working properly and the data transmission is not interrupted. In that case, rapid decisions (in real time or near real time) can be made to achieve high efficiency and automation leading to more resilient communities by benefiting from such real time or near real time data-driven SHM.

4.3 Structural Dynamic Loss Function

In training the Encoder-Decoder network, the model is minimizing the recovery loss. The objective of this minimization is to recover the acceleration time series at all sensor locations. However, one should notice that the response at some locations are higher than other locations. For example, during an earthquake, the acceleration at the roof (top) floor is typically higher than the first floor, as shown in Fig. 4.2. In terms of training the network, the model will try to learn more from locations that have higher acceleration values, because minimizing the loss from these time locations will significantly reduce the total loss.

Mathematically, the governing equations of motion for a linear Multi Degree of Freedom (MDOF) system are expressed as follows [23]:

$$\boldsymbol{m}\ddot{\boldsymbol{u}} + \boldsymbol{c}\dot{\boldsymbol{u}} + \boldsymbol{k}\boldsymbol{u} = \boldsymbol{p}(t),$$

where $\boldsymbol{u}$, $\dot{\boldsymbol{u}}$, and $\ddot{\boldsymbol{u}}$ are respectively the displacement, velocity, and acceleration vectors where each element in these vectors corresponds to one Degree of Freedom (DOF) and superposed dots imply derivatives with respect to the time, t. Moreover, $\boldsymbol{m}$, $\boldsymbol{c}$, and $\boldsymbol{k}$ are respectively the mass, damping, and stiffness matrices. Finally, $\boldsymbol{p}(t)$ is the time-varying external force vector.

The acceleration vector $\ddot{\boldsymbol{u}}$ of a MDOF system can be expanded in terms of the modal contributions. Accordingly, the dynamic response of the system can be expressed as follows:

$$\ddot{\boldsymbol{u}}(t) = \sum_{r=1}^{N} \boldsymbol{\phi_r} \ddot{q}_r(t),$$

where $q_r(t)$ is a scalar modal coordinate that depends on t and corresponds to mode number $r = 1, 2, \ldots, N$, and $\boldsymbol{\phi_r}$ is the mode shape (spatial distribution defined by the DOF) of this r^{th} mode. For stiffer structures with a smaller natural period (higher natural frequency), the first (few) modes tend to control the response. Assuming for practicality and simplicity that the first mode controls the response (typically, the first mode controls because $\ddot{q}_1(t) \gg \ddot{q}_i(t), i = 2, 3, \ldots, N$), the acceleration at each DOF is roughly proportional to $\boldsymbol{\phi_1}$.

In the Encoder-Decoder network, the recovery loss is larger for a DOF with a larger mode shape value. Therefore, the total recovery loss is "dominated" by such DOF. The model selectively fits well to such DOF, as it can significantly reduce the total training error. However, it is desired to learn from all MDOF equally well, while preserving the relative spatial magnitude among these MDOF. The solution to rectify this phenomenon in the training phase is to assign weights to different MDOF, which are inversely proportional to the mode shape value of each DOF. In this way, the mode shape effect on the loss function is eliminated and each DOF has roughly equal contribution to the loss. Thus, the model will roughly pay equal attention to all MDOF. Accordingly, the loss function proposed in this study for time t is as follows:

$$L(\hat{y}, y) = \frac{1}{nDOF} \frac{1}{nStep} \sum_{i=1}^{nDOF} \frac{1}{f(\phi_{i1})} \left(\sum_{t=1}^{nStep} L(\hat{y}(i,t), y(i,t)) \right), \tag{4.1}$$

where ϕ_{i1} is the 1^{st} mode shape value at DOF i, $\hat{y}(i,t)$ and $y(i,t)$ are the reconstructed and true values of the sensor output at DOF i and time step t, and $L(\hat{y}(i,t), y(i,t))$ is the unweighted loss function, which decides on the choice of the function $f(\phi_{i1})$ applied to the mode shape ϕ_{i1}. For example, if the error function is chosen as the *mean absolute error* (i.e., $L(\hat{y}(i,t), y(i,t)) = |\hat{y}(i,t) - y(i,t)|$), then f should be the identity function (i.e., $f(x) = x$). On the other hand, if the error function is chosen as the *mean squared error* (i.e., $L(\hat{y}(i,t), y(i,t)) = (\hat{y}(i,t) - y(i,t))^2$), then f should be the squared function (i.e., $f(x) = x^2$). In this way, the weighted loss value $L(\hat{y}, y)$ corresponding to each DOF is roughly the same. For the ease of illustration, in the following sections, the mean absolute error with f being the identity function is called *absolute loss function*, and the mean squared error with f being the squared function is called *mean squared loss function*.

4.4 Data Processing & Model Training

In this section, the adopted data processing technique is illustrated. The process is divided into three steps: (1) segmentation, (2) an optional smoothing, and (3) normalization.

Segmentation: TS is segmented into several smaller ones, which have fixed length, to facilitate the training of the model. Segmentation is also an augmentation technique to increase the total number of TS used in training and testing. In this regard, as discussed

later, the proposed model is applicable even with limited data size[3]. Sliding window is used to segment the TS. In order to capture the cyclic response, as well as to ensure computational efficiency, the window size should be slightly larger than one natural period. The window stride is a *hyper-parameter* that needs to be tuned. The adopted segmentation technique in this study is illustrated in Fig. 4.5.

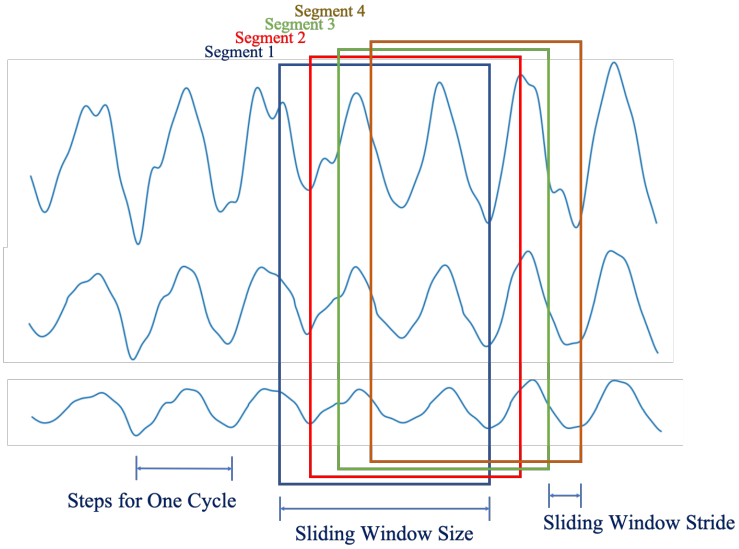


Figure 4.5: Data segmentation.

Smoothing: For real world data, there is noise accompanying the measured data. A common way to quantify (model) the noise in the measurements is by assuming a *normal distribution*, with zero mean and some variance. Averaging the data locally can reduce the variance. In that case, Moving Average (MA) technique can be used to remove the noise and smooth the TS. The window size of the MA is also a hyper-parameter. The larger the window size, the smoother the data would be. However, larger window size can also lead to the loss of information.

Normalization (Standardization): Normalization ensures that values of all TS segments are within the same range. There are two main reasons that each TS segment is

[3]Given a small data size, the over-fit problem is exacerbated. However, as discussed later, the proposed model avoids this problem. Even in one full length TS from one GM, there are variations of the response where all segments start from various phases of the cyclic response. One way to ensure this is by making the sliding window stride (s), i.e., sliding the window over s time steps, and the number of steps for one cycle (n) co-prime (two integers are said to be co-prime if the only positive integer that divides both of them is 1), Fig. 4.5. In this case, the least common multiple is $s \times n$, i.e., it takes n slides to make n segments before the staring phases of segments coincide, thus assuring that the starting phases of the segments are distributed evenly. If s and n are not co-prime, more segments start from the same phase. The robustness of this approach is demonstrated in the numerical experiments discussed in Chapter 5, where the proposed model gives high prediction accuracy values for the GM cases that are totally unseen before.

normalized. Firstly, it can eliminate the effect of the scale and rule out the possibility of *information leakage* (e.g., higher acceleration may indicate more severe damage, but this is not what the network is expected to learn from) when the network is trained. Secondly, in the training process, the model equally learns from all TS segments. Without normalization, the model will try to learn more from the TS which has higher scale, because minimizing the loss from such TS will significantly reduce the total loss. In this case, learning from TS segments with smaller scale would be regarded as useless, because it cannot effectively reduce the total loss. In that regard, each TS segment is normalized by subtracting the mean and then dividing by the difference between the maximum and minimum values. Note that the TS segments are normalized separately. However, for each segment, all records from the different sensors are normalized by the same scale, i.e., all elements in the columns (sensors) corresponding to one row (one time step) of the 2-D tensor are subtracted by the same mean, and then divided by the same difference between the maximum and minimum values. Accordingly, the relative magnitude among measurements from different sensors is preserved, which is important to capture the spatial correlation.

Following the above process of three steps, the data segments are obtained, each represented as a 2-D tensor, with dimensions of the *time step* $\times$ *number of sensors*. The obtained dataset is subsequently split and shuffled into training, "optional" validation, and test sets. Note that in order to avoid information leakage, all segments from one GM are solely put into the training set or the test set, as the data from the same GM tend to be similar. The input and output of the model are summarized in the following list of **Problem Definitions**. The entire input dataset is a three-dimensional (3-D) tensor that is defined as $\mathbf{X}_{1:T} = \{X_{1:T}^{j}, j \in \mathcal{M}\}$, where $\mathcal{M} = \{1, \ldots, m\}$ is the data segment index. $X_{1:T}^{j} = \{X_{1:T}^{(j,i)}, i \in \mathcal{S}\}$ denotes the multi-dimensional segment j with time steps 1 to T that contain $|\mathcal{S}|$ uni-dimensional TS $X_{1:T}^{(j,i)}$, where $|\mathcal{S}|$ is the *cardinality*, i.e., the number of elements in a set, of the sensor set $\mathcal{S}$. Note that $X_{t}^{(j,i)}$ denotes the response at time step t in $X_{1:T}^{(j,i)}$. Specifically, $\mathbf{X}_{1:T} \in \mathbb{R}^{|\mathcal{M}|\times|\mathcal{C}|\times T}$, $\mathbf{X}_{1:T}^{j} \in \mathbb{R}^{|\mathcal{C}|\times T}$, $\mathbf{X}_{1:T}^{(j,i)} \in \mathbb{R}^{T}$, and $\mathbf{X}_{t}^{(j,i)} \in \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. On the other hand, $\mathbf{Y} = \{Y^{j}, j \in \mathcal{M}\}$ and $\hat{\mathbf{Y}} = \{\hat{Y}^{j}, j \in \mathcal{M}\}$ are the *true* and *predicted* labels, respectively, in terms of the damage of segment j. The value of $\mathbf{Y}$ depends on the task, i.e., for damage diagnosis and localization, the class labels are damaged/undamaged (quantified as 1/0); for local damage severity/pattern identification, three or more classes are used, corresponding to different damage severity/pattern classes, which are quantified by a natural number for each class.

Unless otherwise specified, in this dissertation where the ML models are trained, the training process is conducted using `Keras` package [22], a high-level DL Application Programming Interface (API), and Python toolkit `scikit-learn` [90]. The models are trained on a Windows platform (Graphic Processing Units (GPU): GeForce RTX 2080 Ti, Central Processing Units (CPU): 3.70 GHz Intel Core i7-8700K, Memory: 32 GB, Solid State Drive) with a GPU boost.

Chapter 5

Applications of the Proposed Structural Health Monitoring Framework

In this chapter, the proposed Structural Health Monitoring (SHM) framework described in Chapter 4 is applied for two appication examples. These are the planar Reinforced Concrete (RC) moment frame, and the space Concentrically Braced Steel (CBS) frame. The models using the Finite Element Method (FEM) are described in Chapter 3.

5.1 Planar Reinforced Concrete Moment Frame

5.1.1 Data, Label, and Training Process

The data sets, used for training and testing the Long Short-Term Memory (LSTM) Encoder-Decoder network in this chapter, are obtained from the simulated Time Series (TS) data using models based on the FEM. The segmented TS data sets, refer to Chapters 3 & 4, are based on a sampling frequency of 400 Hz, and accordingly a time step size of $1/400 = 0.0025$ sec. Since the natural period of the structure is 0.4 sec, Chapter 3, the time steps required to capture a full natural period are at least $400 \times 0.4 = 160$. This justifies the need to use LSTM cells, where long term dependencies need to be captured. In this application example, a window size of 197 steps, which is about 1.25 cycles, is selected with a window stride of 20 steps.

In order to test the robustness of the LSTM model and account for the real world measurement from sensors, random noise is added to the acceleration data [62] after normalization, which is the process described in Chapter 4. The noise followed a Gaussian distribution, with zero mean and standard deviation of 0.05 (recall that the acceleration TS is normalized), and added to the acceleration TS inputs. Example of the test set segment is shown in Fig. 5.1 (The reconstructed TS is discussed later). The loss function used in this application is

the mean squared error. For normally distributed, denoted $\mathcal{N}$, noise, this loss function is used because training the model using squared error is equivalent to the maximum likelihood estimation of the model, which predicts the real value without the noise[1].

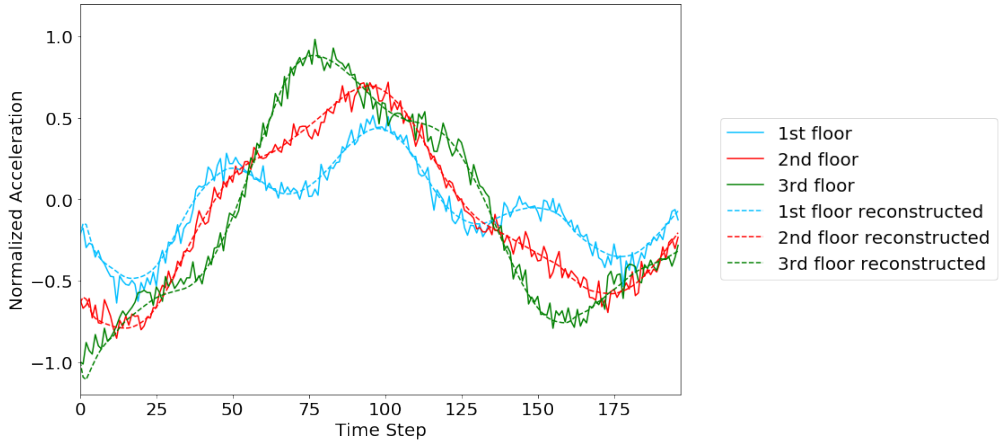


Figure 5.1: Example of the original and reconstructed TS of the RC frame.

The labels for training and testing the model are obtained from the damage status of structure at hand. In this application, the damaged status of the RC framed structure is quantified as the stiffness degradation of the columns of the different floors, which causes the change of the horizontal accelerations at the sensor DOF locations. For the columns, a decrease of stiffness of 10% or 20% is applied, where the columns have remaining 90% or 80% stiffness, respectively. In this and the following application example, three SHM tasks are put forward, and each task had separate set of labels (for different floors). The three tasks are:

[1]**Derivation**: Suppose the noise $\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$, then the output $y_i \sim \mathcal{N}\left(f(X_i), \sigma^2\right)$, where f is the model function, and X_i is the input data. Thus, the Probability Density Function (PDF) of y_i (in log form) is:

$$\ln p(y_i) = -\frac{\left(y_i - f(X_i)\right)^2}{2\sigma^2} - \text{constant}. \tag{5.1}$$

Therefore, the log likelihood function is obtained as follows:

$$\begin{aligned} \ell(f; X, y) &= \ln\left(p(y_1)p(y_2)\dots p(y_n)\right) \\ &= \ln p(y_1) + \ln p(y_2) + \dots + \ln p(y_n) \\ &= -\frac{1}{2\sigma^2}\sum\left(y_i - f(X_i)\right)^2 - \text{constant}. \end{aligned} \tag{5.2}$$

Therefore, using maximum likelihood on the model function f is equivalent to estimating f by least-squares. Since the noise is normally distributed, the maximum density of y_i occurs when it is equal to the true value $f(X_i)$, i.e., when the error is zero. Therefore, the maximum likelihood estimation attempts to eliminate the influence of the normally distributed noise, and predicts the real value without the noise.

1. **Structural overall damage diagnosis**: The whole structure is regarded as damaged if the stiffness of any column is reduced by specified amount.

2. **Damage localization**: The floors are identified as being damaged if the stiffness of the columns are reduced by specified amounts at these floors.

3. **Local damage pattern/severity identification**: Similar to the above task, the floors are identified as being damaged if the stiffness of the columns are reduced by specified amounts at these floors, and the model is classified as: (i) no damage, (ii) 90% stiffness, and (iii) 80% stiffness.

One of the main difference between this application and the following one is that the damage states are clearly pre-defined and they are directly reflected (explicitly modeled) in the FEM model. Such clear damage states and corresponding labels are important for evaluating the validity of the proposed SHM model, albeit this situation does not represent the real cases of structures. In the second application, the damage states are determined from the simulations mimicking real field damage conditions. Note that in the current application, all the Ground Motion (GM) is applied to all stiffness degradation cases in the simulations. Therefore, this simulation implicitly guarantees that the dataset is balanced.

An important design consideration of the proposed SHM model using LSTM is the number of units in each LSTM time step cell, which is the dimension of the Latent Space Vector (LSV). Less number of units means the model is simpler and easier to train, since there are fewer model weight parameters. More number of units means the model is more complex and expressive, and it will generally take longer to train. Simpler models could lack expressiveness and can lead to under-fitting. More complex models could lack generalization capacity and can lead to over-fitting. Therefore, models with different number of units are explored, and the model that produces the best performance with reasonable computational effort is selected. In this application, models with # of units = 10, 20, 50, 100, 200, and 1,000 are trained. A symmetric architecture between encoder and decoder is adopted, i.e., the number of units in each LSTM cell for the encoder and decoder are the same. The training process starts with a *learning rate* of 0.001, and the *rate decay* is adopted to reduce the learning rate when the loss stopped decreasing for 50 epochs. *Early stopping* is adopted during the training process if the model stopped training when the validation loss is not improving for 1,000 epochs. The maximum number of epochs is set to 5,000 for computational efficiency. The models with the least loss for the validation set are saved for later usage.

5.1.2 Results

The results for the RC frame application are discussed in this subsection. Models with different # of units are compared. It should be noted that in the tables of the results below, underlined findings of the loss or accuracy values correspond to the best (least loss or highest accuracy) model in the considered set of models. The accuracy is quantified as the ratio between the number of correct classification cases and that of all classification cases,

see Section 2.2.1). Note that the presented accuracy results are for the testing set, which is totally unseen in the whole training process.

Loss and training time: The losses using the training and validation sets, as well as the model training time are listed in Table 5.1. It is observed that for # of units = 50, the model produces minimum validation loss, which means that the compression from the TS data into the LSV is the most lossless. For # of units = 10 or 20, the training loss and validation loss are both higher, where these simple models under-fits the data. For models with # of units $\geqslant$ 50, the training loss tends to practically stay the same or decrease (an exception is the training loss of the model with very high # of units = 1,000), and the validation loss tends to consistently increase. Therefore, these complex models over-fits the data, as they fail to generalize well to the unseen validation dataset. Since the differences in the loss values may not be very clear for selecting the optimal number of units (e.g., loss values for models with 50 and 100 units are very close), for the following classification tasks, the accuracy results are presented for models with # of units = 20, 50, 100, and 200. In terms of the training time, in general, models with more units take more time. However, it sometimes happens that more complex models take shorter time, e.g., models with # of units = 100 and 200 compared to that with # of units = 50. This is attributed to early training, where the model stops training when the validation loss is not decreasing after 1000 epochs of training. Moreover, some complex models may converge faster with fewer total number of epochs before stopping. One example[2] of a reconstructed TS is shown in Fig. 5.1. It is observed that through the process of reconstruction, the difference between the original TS (in solid lines) and the reconstructed TS (in dashed lines) are minimal. The reconstructed TS is smoother, implying that the noise from the original TS is removed.

Table 5.1: Training loss, validation loss, and training time for the RC frame.

# of Units	**10**	**20**	**50**	**100**	**200**	**1,000**
Training loss	5.78e-3	1.07e-3	5.10e-4	5.15e-4	4.41e-4	8.86e-4
Validation loss	5.87e-3	1.45e-3	1.28e-3	1.61e-3	2.47e-3	2.75e-3
Training time (min.)	62.31	68.25	87.51	55.69	49.13	123.39

Task 1. Structural overall damage diagnosis: The results of this task are shown in Table 5.2. It is observed that, in general, the LSTM model achieve an accuracy of higher than 90% and the one with # of units = 50 has the highest accuracy of more than 95%.

Task 2. Damage localization: The results of this task are shown in Table 5.3. The accuracy results for floor 1 are always higher than those of the other two floors. One possible

[2]This example corresponds to the model with # of units = 50 for data obtained from simulations using earthquake GM at Livermore (1980) measured at Morgan Terr Park. The used earthquake has a moment magnitude $M_w = 5.42$ and distance (taken as the closest horizontal distance to the vertical projection of the rupture, namely the "Joyner-Boore" distance, R_{jb}) of 7.94 km. For a complete list of applied GM, refer to Appendix B.

Table 5.2: Accuracy results of Task 1 for the RC frame.

# of Units	**20**	**50**	**100**	**200**
Accuracy (%)	91.4	<u>95.4</u>	92.5	92.4

explanation is that the reduction of stiffness at the first floor affects the response of the first floor, as well as all floors above it. This subsequently affects the recorded (computed in the present application) accelerations at all floors. In contrast, if the stiffness is reduced in the third floor, for example, only the acceleration at the roof is significantly influenced, and the accelerations of the lower floors do not significantly change. Since the LSTM model takes the input from all recorded accelerations, the TS changes in multiple locations make it easier for the LSTM model to distinguish between damage/no damage cases. Therefore, it is easier to detect the damage in the first floor than the others. The results are in general about or higher than 90% for all three floors and for models with any # of units.

Table 5.3: Accuracy results of Task 2 for the RC frame.

# of Units		**20**	**50**	**100**	**200**
Accuracy (%)	Floor 1	93.4	96.2	96.4	<u>96.6</u>
	Floor 2	89.6	92.7	<u>92.8</u>	90.4
	Floor 3	89.6	<u>92.8</u>	91.0	91.3

Task 3. Local damage pattern/severity identification: The results of this task are shown in Table 5.4. Similar to Task 2, the accuracy for the first floor local damage severity identification is the highest compared to the other two floors. It is also observed that the results for floor 3 are slightly higher than 80%.

Table 5.4: Accuracy results of Task 3 for the RC frame.

# of Units		**20**	**50**	**100**	**200**
Accuracy (%)	Floor 1	92.5	<u>96.5</u>	92.2	92.4
	Floor 2	83.4	<u>91.2</u>	84.0	85.7
	Floor 3	80.6	83.4	81.7	<u>83.9</u>

From the above, the model with # of units = 50 produces the best results in general. Models with # of units = 100 and 200 produced reasonable results with slightly less accuracy. On the other hand, the model with # of units = 20, due to under-fitting, has the worst results. In terms of classification tasks, floor 1 is the easiest, while floor 3 is the hardest. Fig. 5.2 plots the confusion matrices for this model for the three floors (see Chapter 2 for the definition and interpretation of the confusion matrix where the column labels in the matrices are the

true labels, while the row labels are the predicted ones). For floor 1, the results are quite satisfactory, and the model could completely distinguish between 80% stiffness and other cases. However, for floor 3, it is hard to distinguish, in particular, between 90% stiffness (the intermediate case between no damage and 80% stiffness) and the other two cases. Moreover, it is interesting to note that the classification is mainly one class away, i.e., the probability of misclassification between no damage and 80% stiffness is much lower (about one order of magnitude less), compared to other misclassification cases (no damage versus 90%, and 90% versus 80%).

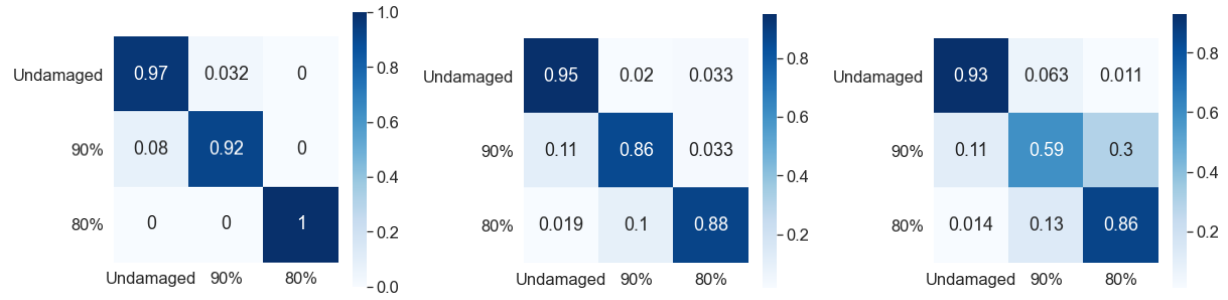


Figure 5.2: Confusion matrices of Task 3 for the model with # of units = 50 for the RC frame: floor 1 (left), floor 2 (middle), and floor 3 (right).

5.2 Space Concentrically Braced Steel Frame

5.2.1 Data, Label, and Training Process

As described in Section 5.1, the used data are obtained from the simulated TS data from the FEM. For simplicity, only the data from the X-direction is considered herein where the recordings from the center points of the floors are used. The sampling rate of the recordings from the model simulations using FEM is 100 Hz. The maximum natural period observed in the original shaking table testing is 1.36 sec. Thus, in order to capture the response of one natural period, 1.36 sec$\times$100 Hz = 136 time steps per cycle of response are needed. A window size of 160 time steps is adopted in this application. In this application, only one segment is used from the entire GM for each case.

For the damage quantification, the structure is regarded as damaged if the braces experienced inelastic elongation. The forces (and the elongations) from the braces are recorded after simulations to determine the damage states. For each floor, three damage levels are determined, namely, *no damage* (ND, no loosening of any brace in the floor), *partial loosening* (PL, loosening of less than or equal to 75% of all braces in the floor), and *all loosening* (AL, loosening of more than 75%, i.e., almost all braces, in the floor). Therefore, one of the main difference between this application and the above one is that the damage states are

determined during the simulation rather than pre-defined (by reducing the column element thickness of the 2-D FEM model) during the modeling. This is a more difficult task than that in the previous application, but it is a more realistic case.

The boundaries between the damage levels, especially PL and AL, are blurry. In other words, there is no sudden jump of the damage or response among the damage levels, especially near the boundaries of these damage levels. This is expected to pose challenges for the classification of these damage states, where clear boundaries are to be imposed on the data with somewhat continuous response variation. Accordingly, it is difficult to classify data near the boundaries of such damage levels as discussed with the results below. In this application, models with # of units = 20, 50, 100, 200, and 500 are explored. Similar to the RC frame application, early stopping is used during the training.

5.2.2 Results

Loss and training time: The loss values of the training and validation datasets are listed in Table 5.5 where underlined findings of the loss or accuracy values correspond to the best (least loss or highest accuracy) model in the considered set of models. It is clear from these results that the model with # of units = 100 produced the minimum loss. The accuracy results for all models with different # of units are presented for all classification tasks.

Table 5.5: Training loss, validation loss, and training time for the CBS frame.

# of Units	**20**	**50**	**100**	**200**	**500**
Training loss	48.24e-3	3.68e-3	2.09e-3	3.51e-3	3.73e-3
Validation loss	50.94e-3	6.52e-3	3.20e-3	7.36e-3	6.39e-3
Training time (min.)	13.73	16.39	16.39	24.44	59.64

Task 1. Structural overall damage diagnosis: The accuracy results are listed in Table 5.6. It is clear that the model achieved an accuracy higher than 90% for # of units = 50 or 100. On the other hand, the accuracy of models with # of units = 20 & 500 are the worst, most probably due to the respective under-fitting and over-fitting problems.

Table 5.6: Accuracy results of Task 1 for the CBS frame.

# of Units	**20**	**50**	**100**	**200**	**500**
Accuracy (%)	88.3	91.1	92.2	88.9	86.1

Task 2. Damage localization: The accuracy results are shown in Table 5.7. Similar to Task 1, for the model with # of units = 50 or 100, the classification achieved an accuracy higher than 90%. Moreover, the accuracy for lower floors are mostly higher for these models.

Table 5.7: Accuracy results of Task 2 for the CBS frame.

# of Units		20	50	100	200	500
Accuracy (%)	Floor 1	85.6	92.8	95.0	92.7	87.8
	Floor 2	88.3	92.8	92.8	92.2	88.3
	Floor 3	87.2	92.2	90.0	88.9	82.2

As explained in the RC frame application, it is easier to identify the damage in lower floors.

Task 3. Local damage pattern/severity identification: The classification of this task distinguishes between UD, PL, and AL classes. Therefore, the accuracy results are generally lower than that of Task 2. The model with # of units = 100 has the highest overall accuracy being higher than 90% for the first two floors, and slightly lower than 90% for the third floor. Again, the model with # of units = 20 has the worse performance.

Table 5.8: Accuracy results of Task 3 for the CBS frame.

# of Units		20	50	100	200	500
Accuracy (%)	Floor 1	83.8	90.0	95.5	90.0	88.9
	Floor 2	79.4	89.4	90.0	88.9	82.8
	Floor 3	81.7	88.9	89.4	85.5	80.0

From the above results, the model with # units = 100 is the most robust one for all tasks of this application. Fig. 5.3 plots the accuracy confusion matrices for the model with # of units = 100. It is observed that the classification is mainly one class away, i.e., almost no UD cases are classified as AL cases, and vice versa. Moreover, as explained above, it is harder to classify damage cases near the boundary of the damage state labels, because the damage severity is continuous in nature. This is also true for a human expert attempting to make such strict classifications of continuous damage state into discrete classes. This explains the observed misclassification errors for adjacent classes.

5.3 Discussions

In this section, one possible modification of the model architecture, a multilayer (stacked) LSTM model, is explored to investigate possible improvements of the proposed framework. In addition, the results from the t-distributed Stochastic Neighbor Embedding (t-SNE) visualization is demonstrated to justify the learned LSV's.

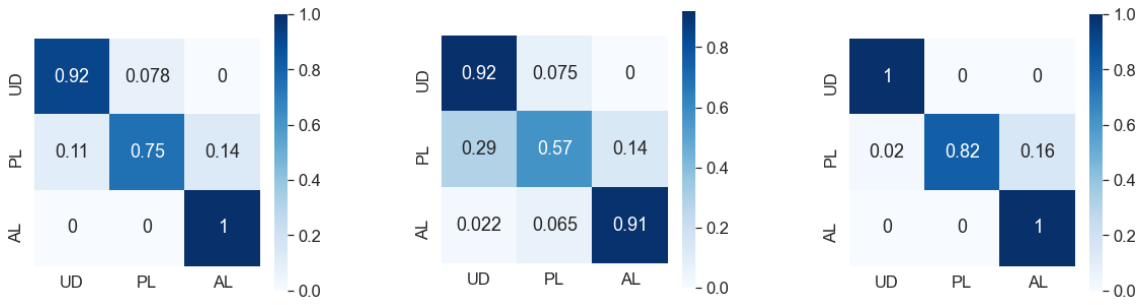


Figure 5.3: Confusion matrices of Task 3 for the model with # of units = 100 for the CBS frame: floor 1 (left), floor 2 (middle), and floor 3 (right).

5.3.1 Two-layer LSTM

Apart from the single layer LSTM, multilayer (stacked) LSTM architecture could be used to increase the expressiveness of the model. The mechanism of stacked LSTM is discussed in Chapter 2. Fig. 5.4 shows the stacked LSTM Encoder-Decoder architecture used herein. Note that the hidden state vector at layer 2 is passed from encoder to decoder, and used in the classification tasks. A symmetric structure between encoder and decoder is adopted. It is noted that in the tables of the results below, underlined findings of the loss values correspond to the best (least loss) model among the considered set of models. Moreover, for comparison between two- and one-layer models, underlined findings for the two-layer model correspond to the cases where better results are obtained than their one-layer counterparts.

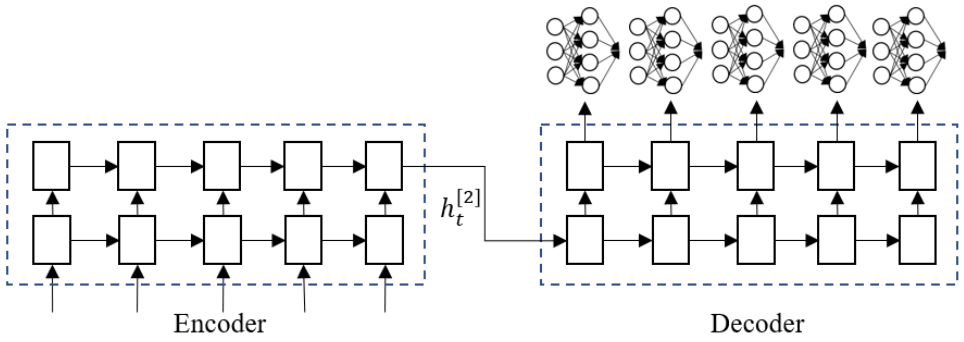


Figure 5.4: Two-layer LSTM Encoder-Decoder Network.

For brevity, only the results for the CBS frame are shown. Table 5.9 shows the loss and training time. The model with # of units = 20, 50, 100, and 200 are trained. More complex networks with more units, e.g. 500, are not trained, where from the one-layer models, more units led to their over-fitting problems. The loss values are not significantly different from

those for the one-layer networks. In terms of the training time, not surprisingly, it is longer than the case for the one-layer network with a factor of 2, in general.

Table 5.9: Training loss, validation loss, and training time for the CBS frame, 2-layer LSTM.

# of Units	**20**	**50**	**100**	**200**
Training loss	5.11e-3	2.56e-3	1.64e-3	2.24e-3
Validation loss	7.33e-3	5.87e-3	7.33e-3	6.39e-3
Training time [Tt] (min.)	29.55	33.40	37.71	53.17
Tt (min.) for 1-layer	13.73	16.39	16.39	24.44
Tt ratio: 2-layer/1-layer	2.15	2.04	2.30	2.18

Tables 5.10 to 5.12 list the accuracy for the three considered tasks. In comparison to the results for the one-layer networks, no significant overall improvement is observed with the exception of models with # of units = 20 & 50 showing some improvements. A possible explanation for this exception is that one-layer networks with # of units = 20 & 50 under-fitted the data and the two-layer counterparts somewhat resolves this problem by using more complex networks. However, for models with # units = 100, since the one-layer network fits the data almost perfectly, increasing the number of layers can over-fit the data leading to lower accuracy for the two-layer networks than the one-layer ones in this case.

Table 5.10: Accuracy results of Task 1 for the CBS frame, 2-layer vs. 1-layer LSTM models.

# of Units	**20**	**50**	**100**	**200**
Accuracy (%) for 2-layer	90.0	91.1	90.6	86.7
Accuracy (%) for 1-layer	88.3	91.1	92.2	88.9

Table 5.11: Accuracy results of Task 2 for the CBS frame, 2-layer vs. 1-layer LSTM models.

# of Units		**20**	**50**	**100**	**200**
Accuracy (%) for 2-layer	Floor 1	93.3	94.4	92.8	91.1
	Floor 2	92.2	93.3	91.7	90.6
	Floor 3	91.1	92.2	89.4	87.2
Accuracy (%) for 1-layer	Floor 1	85.6	92.8	95.0	92.7
	Floor 2	88.3	92.8	92.8	92.2
	Floor 3	87.2	92.2	90.0	88.9

Given the observations from these computer experiments, it is concluded that using two-layer networks does not significantly improve the results. Considering the trade-off between

Table 5.12: Accuracy results of Task 3 for the CBS frame, 2-layer vs. 1-layer LSTM models.

# of Units		**20**	**50**	**100**	**200**
Accuracy (%) for 2-layer	Floor 1	90.0	94.4	91.7	90.0
	Floor 2	90.0	91.1	88.9	88.3
	Floor 3	86.1	88.9	88.9	85.6
Accuracy (%)	Floor 1	83.8	90.0	95.5	90.0
	Floor 2	79.4	89.4	90.0	88.9
	Floor 3	81.7	88.9	89.4	85.5

model complexity (training time) and accuracy, using the one-layer networks is already adequate enough for the tasks at hand. Therefore, models with more layers, i.e., three-layer or four-layer networks, are not explored in this study.

5.3.2 t-SNE Visualization

The LSV could be visualized to view its clustering properties. One of the most popular visualization technique is the t-SNE visualization developed by van der Maaten & Hinton [69]. It is a nonlinear dimensionality reduction technique. Specifically, it models each high dimensional vector onto low dimensional space (1 to 3 dimensions generally) for ease of visualization. This is performed in such a way that similar objects are mapped as nearby points and dissimilar objects are mapped as distant points with high probability. Mathematically, given high-dimensional space vectors, $x_1, x_2, \ldots, x_n$, the t-SNE algorithm first computes the probabilities as follows:

$$p_{ij} = \left(p_{i|j} + p_{j|i}\right)/2, \tag{5.3}$$

$$p_{a|b} = \frac{\exp\left(-\parallel x_b - x_a \parallel^2 / 2\sigma_b^2\right)}{\sum_{k \neq b} \exp(-\parallel x_b - x_k \parallel^2 / 2\sigma_b^2)}, \tag{5.4}$$

where $\parallel \bullet \parallel$ is the ℓ_2 norm of $\bullet$ (i.e., $\parallel z \parallel = \sqrt{z^T z} = \sqrt{\sum_{i=1}^{n} z_i^2}$ for $z \in \mathbb{R}^n$) and σ_i is the variance of the Gaussian distribution that is centered on data point x_i. These probabilities are mapped to the low dimension vector $y_1, y_2, \ldots, y_n$, making use of the following definition:

$$q_{ij} = \frac{(1+ \parallel y_i - y_j \parallel^2)^{-1}}{\sum_{k \neq i} (1+ \parallel y_i - y_k \parallel^2)^{-1}}. \tag{5.5}$$

This mapping is determined by minimizing the Kullback-Leibler divergence (KL-divergence) [56] of distribution P from distribution Q. This KL-divergence is expressed as follows:

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log\left(p_{ij} / q_{ij}\right). \tag{5.6}$$

The t-SNE is used to map the LSV's onto a 3-D space using the Python toolkit `scikit-learn` [90]. Fig. 5.5 shows the t-SNE mapping for all data points corresponding to the one-layer LSTM framework with # of units = 100, which produces best classification results for Task 3 for the damage pattern/severity of the first floor of the CBS frame application[3]. From this visualization, it is observed that, as discussed above, there is no clear separation (boundaries) between the continuous damage classes. The points are not strictly clustered based on the defined labels by a human expert. This explains the misclassification error of the damage pattern identification. It is to be noted that from right to left in Fig. 5.5, the damage classes are more severe (i.e., points closer to the left are more likely to correspond to a more severe damage class). That explains why the NN can classify these damage states.

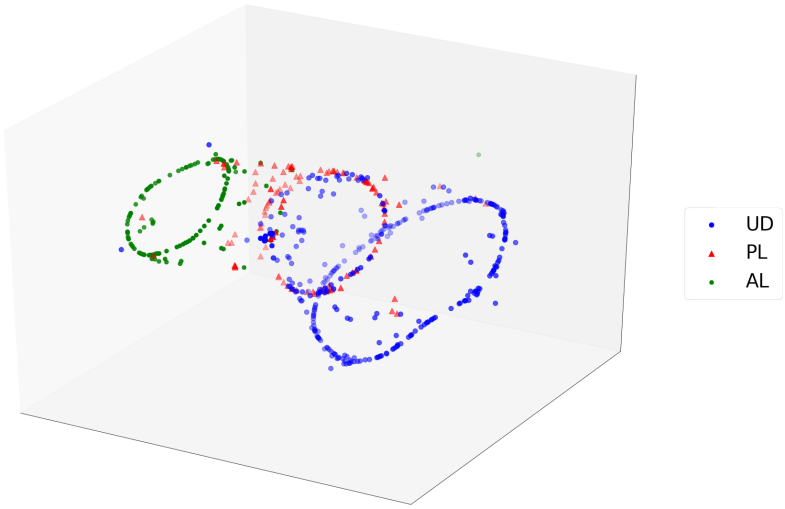


Figure 5.5: t-SNE visualization of Task 3 for the model with # of units = 100 for the CBS frame.

[3]t-SNE is a visualization technique, and it is not suitable for quantitative interpretation. Thus, its axes and scales are removed deliberately in Fig. 5.5 for better visualization of the clustering.

Chapter 6

Structural Response Prediction Using Deep Neural Network

6.1 Introduction

Analyzing the structural responses as Time Series (TS) at different locations is important for the estimation of consequences of severe earthquakes. For example, in the Performance-Based Earthquake Engineering (PBEE) framework [78][122], the Engineering Demand Parameters (EDP), such as peak interstory drift ratio and peak floor total acceleration, are used to estimate losses. The EDP can be obtained by finding the maximum values in the TS output. Traditionally, the nonlinear response of structures under earthquakes are computed using physical models, e.g., Finite Element Method (FEM). In general, nonlinear time history analyses are time consuming because of the need for step-by-step time integration (e.g., using Newmark β method [23]) and nonlinear iterative solution (e.g., using Newton-Raphson method [23]) for each time step. The problem of computational efficiency is more significant in the case when the number of structures to be analyzed is excessive as in the regional seismic damage assessment [121] and when rapid results are needed for making decisions. Therefore, more computationally efficient methods should be employed to expedite the process of predicting the structural responses at critical locations of these structures.

In recent studies, the Long Short-Term Memory (LSTM) networks have attracted attention for structural response predictions (e.g., [130][129][39]). However, most of these studies have used "vanilla"[1] LSTM networks (with varying number of LSTM layers). In the recent studies, LSTM in combination with other basic building blocks of deep Neural Network (NN) have been designed, e.g., convolution [68] and attention [112][20] layers. These combinations were capable to handle specific tasks and improve the prediction accuracy.

In this dissertation, a fast method for predicting the responses is developed and validated in terms of accuracy by comparing to existing models. This method uses a few variants of the

[1] In computer science, vanilla is the term used to refer to an algorithm not customized or updated from its original form.

LSTM network, as well as a novel network called Temporal Convolutional Network (TCN). The inputs to all models are the unidirectional uniform Ground Motion (GM) acceleration TS. On the other hand, the output of the models are the acceleration TS from specified critical structural locations. It is observed through: (i) the layer-by-layer forward propagation mechanism, (ii) the nonlinear activation mechanism, and (iii) the gating mechanism inside the LSTM cells, that the models proposed here are able to accurately predict the response of structures under earthquakes. This is validated using recorded real sensor measurements from the California Strong Motion Instrumentation Program (CSMIP), as well as the simulated data from the Concentrically Braced Steel (CBS) frame model described and analyzed in Chapters 3 and 5. The performance of these models are compared in terms of the correlation, error distribution, and training time.

6.2 The Considered Models

Five models (with varying number of model parameters) are designed and considered in this study. Some models are well-established ones, which are adopted here without any modifications (e.g., LSTM). Some models are inspired by others (e.g., LSTM+attention, and LSTM+convolution). Similar to the Structural Health Monitoring (SHM) framework described in Chapter 4, the input TS at different time steps is the input for different cells of the LSTM network. The main difference between the models proposed in this chapter and the SHM framework is that, the hidden state vectors, h_t, at ALL time steps are recorded, and such hidden state vectors are used to output the responses of the structure at the corresponding time steps.

Before the models are described, the used notations are summarized here for better readability. Let x_t be the input GM vector at time step t. For the unidirectional uniform excitation, x_t is a scalar value[2]. Let y_t be the output response prediction vector at time step t, where its cardinality (size) is the number of locations in the structure (e.g., building) of interest. Let h_t be the hidden state vector, which is the output of the LSTM cells. It is noted that h also denotes the hidden layer outputs if the TCN is used, where the output of the i-th layer at time step t is denoted as $h_t^{[i]}$.

6.2.1 One-layer LSTM

The one-layer LSTM is used as the first and simplest model. The input for each cell is the recorded GM at each time step. The hidden state vector, h_t, at each time step is connected to ordinary Fully Connected (FC) NN to output the responses at different locations at each time step, Fig. 6.1.

[2]When the GM excitation is not uniform, e.g., different excitations applied at different locations of the base, or when using multidirectional GM, x_t becomes a vector.

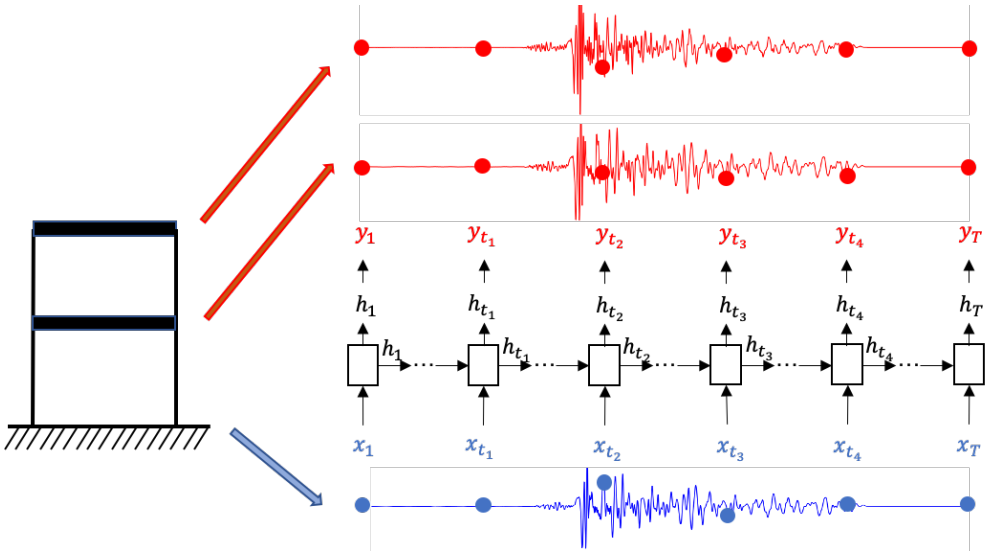


Figure 6.1: One-layer LSTM model.

6.2.2 Two-layer LSTM

The two-layer LSTM cells are stacked to increase the model expressiveness. The hidden state vectors, h_t, of the first layer cells are used as the input for the second layer cells, and the structural response is obtained from the output of the second layer cells. The FC NN is used to connect the h_t vectors of the second layer cells to the predicted responses, Fig. 6.2.

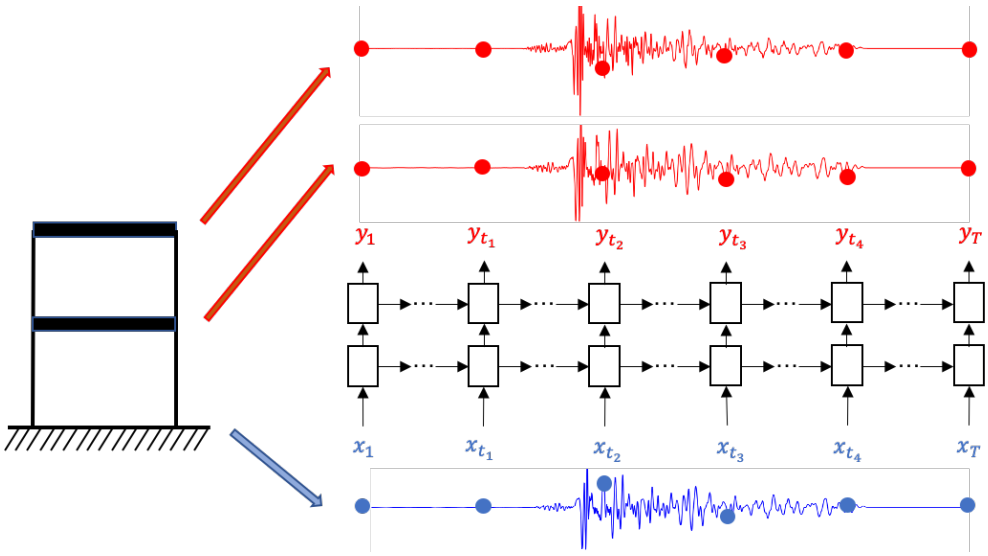


Figure 6.2: Two-layer LSTM model.

6.2.3 LSTM with Attention Mechanism

In the two models above, the final responses are predicted using the hidden state vectors of a single step (i.e, h_t of time step t predicts the response of this time step). The prediction results can be improved by using an attention mechanism, i.e., the response at each step is predicted using a weighted average of the hidden state vectors of the current and previous steps. Therefore, the model could "reinforce" the prediction by recalling the responses in the previous cycles. Concretely, the weighted hidden state vector at time step t is expressed as follows:

$$h_t^* = \sum_{t'=t-N}^{t'=t} \alpha_{tt'} h_{t'}, \tag{6.1}$$

$$\alpha_{tt'} = \text{softmax}_{t'} \left(f(h_t, h_{t'}) \right), \tag{6.2}$$

where $0 \leqslant \alpha_{tt'} \leqslant 1$ is the score (weight) for y_t to pay attention to the hidden state vector at time step t', and $f(h_t, h_t')$ is a function that needs to be determined. Two functions are explored in this study, which are *self-attention mechanism* [118] and *fixed-attention scores model*. The self-attention mechanism computes separate *key vector* and *query vector* of the hidden state vector (through matrix multiplications between the hidden state vector and the corresponding two weight matrices, where the weights are back-propagated in the training process) for each time step. Subsequently, the score function $f(h_t, h_t')$ is calculated as the of the inner product of the key vector of time step t and the query vector of time step t'. The fixed-attention score model is not a trained model. Instead, it uses a fixed score between two steps. The responses are cyclic in the Fast Fourier Transform (FFT) sense, and the periods are dependent on the natural periods of the structure and the periods of the applied GM. Therefore, several most dominant periods of the GM are obtained using FFT analyses, and weights are applied on these periods. For other periods, the weight is zero. The model only looks backward rather than forward by using a weighted average among the current step and the previous steps. In addition, to limit the computational effort and to reduce unnecessary averages, the time step ranges are constrained for the self-attention model, i.e., specify N in Eq. 6.1. This is a model design parameter that needs to be tuned in the experiments. The black triangles in Fig. 6.3 represent the above attention mechanism. It should be noted that the above fixed-attention scores model is not common in literature and practice, and it is designed here to consider the cyclic responses in the FFT sense.

6.2.4 LSTM with Convolution Filters

Instead of using the GM as the input for the LSTM cells, this model uses convolution filters to pre-process the input GM. These filters are able to scan the local patterns of the GM and observe the trends of variation. Therefore, the input for each LSTM cell is a vector, whose dimension is the number of local filters. The mechanism of the convolution filters is similar to the lower layers of the Convolutional Neural Network (CNN) in Computer Vision (CV), where the edges of an image are detected using local convolution filters. Section 2.2,

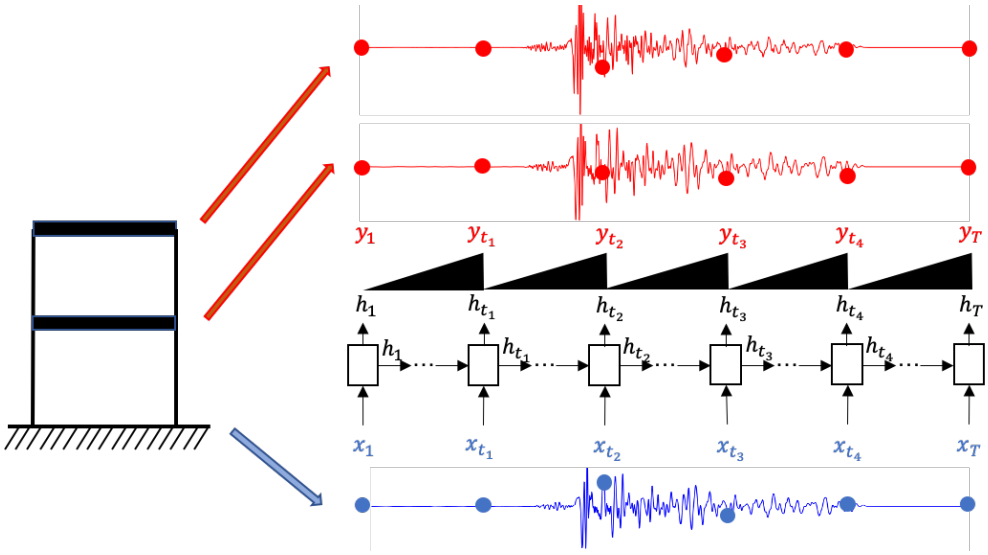


Figure 6.3: LSTM with attention mechanism.

in particular Fig. 2.8, describes the convolution operation. Typically, multilayers are used (e.g., 3 filters are used in Fig. 6.4, and in this case, the cardinality (size) of x_t is 3), where different filters in these layers learn different local patterns. The number of filters for each layer is another model design parameter.

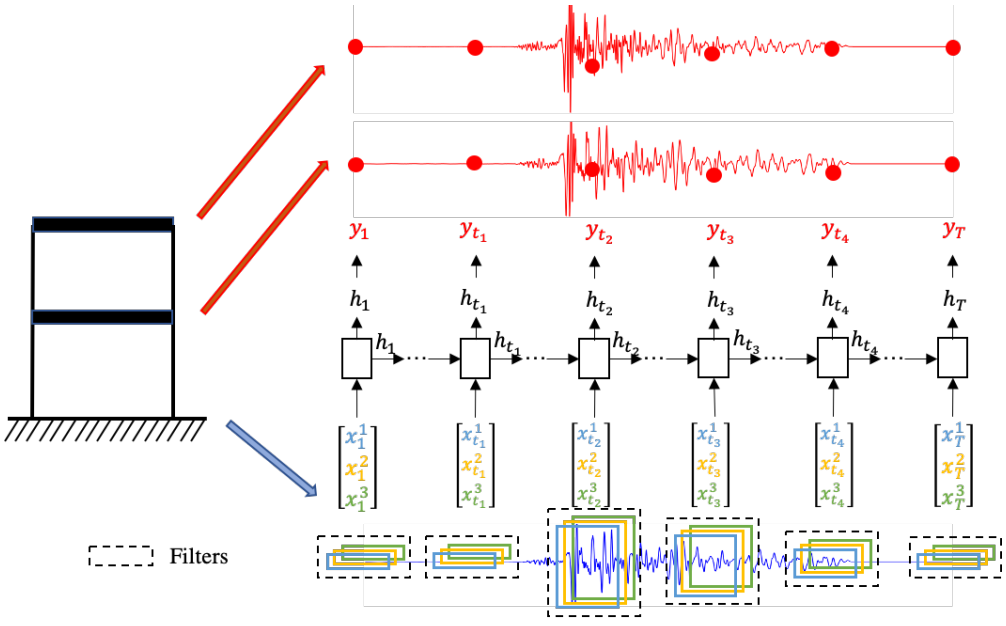


Figure 6.4: LSTM with convolution filters.

6.2.5 Temporal Convolutional Network

The mechanism of the TCN [60][28] is different from the previous four models. In comparison to these models, this model does not use LSTM cells. Instead, it relies totally on the convolution layers. It uses convolution filters to scan the input at different scopes by controlling a *dilation factor*. The formulation of the TCN is expressed below. It is noted that the input x and the output y are the input of the first hidden layer (denoted as $h^{[0]}$) and the output of the last hidden layer (denoted as $h^{[n]}$), respectively.

$$h_t^{[i+1]} = \sum_{l=0}^{k-1} w_l^{[i]} h_{t-d\times l}^{[i]}, \tag{6.3}$$

where d is the dilation factor and k is the kernel size of the convolution layers. The dilation factor determines how far the model can look backward for each layer. In Fig. 6.5, the bottom layers with smaller dilations observe the "local" patterns of the response, while the top layers with large dilations have large scope and observe the "global" response. Therefore, similar to the LSTM models, the TCN model is able to capture the long-term dependencies. In practice, d in a layer is taken as twice the value in the previous layer. The model is demonstrated in Fig. 6.5, where the black lines represent the $h_t^{[i]}$ (which are represented as the dots in the lower layer) that are used to calculate $h_t^{[i+1]}$ (which are represented as the dots in the upper layer). Note that in Fig. 6.5, only one filter is represented for each layer, where in actual implementation, similar to LSTM with convolution filters model, multiple filters are usually used to learn different local patterns. One hypothesis for the TCN model is that its training and prediction processes are faster than models with LSTM layers because the latter require step-by-step propagation, which is a slow process as the total number of time steps are typically large. In contrast, the TCN model could process steps in parallel, and later steps do not need to wait for the results (as for h_t in LSTM cells) from previous steps for processing.

6.3 Evaluating Metrics

Three evaluating metrics are used to quantitatively compare the performances of the five models described above, which are correlations between true & predicted responses, error distribution (in %), and training time. The correlations and error distributions are adopted here because they are also used in the study by Zhang et al. [130] to serve as a useful comparison between the "vanilla" models from literature and the more complex models developed in the present study. The training times are used throughout the study to quantify the time consumption and computational efficiency for the training process preceding the real predictions.

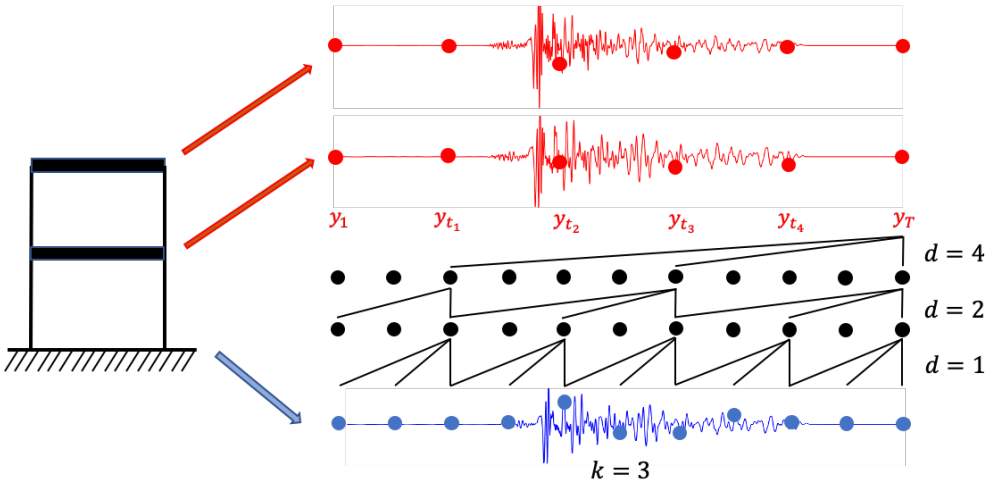


Figure 6.5: Temporal Convolutional Network.

6.3.1 Correlation

Correlation in the context of this dissertation is the correlation between the true response and the predicted one. These responses from all considered cases, time steps, and locations of one model are collected, and separate correlation coefficients are calculated for the training set and the test set. Therefore, a higher correlation value corresponds to a better model performance.

6.3.2 Error Distribution

The considered error herein is the relative error between the real response and the predicted one. Mathematically,

$$err = \frac{y_i^{\text{true}} - y_i^{\text{predicted}}}{\max_t(y_i^{\text{true}})}, \tag{6.4}$$

where $\max_t$ is the maximization over the time steps for each response case. The error is comparable among response cases and locations, since it is normalized with the maximization operation in the denominator. The error from all response cases and time steps are collected, and the distribution could be plotted. Therefore, the more the distribution is centered around 0.0, the better the model is.

6.3.3 Training Time

Equal number of time steps are set for different models. Therefore, the training time is expected to be comparable among the models. As stated above, the TCN model is expected to have the least training time.

6.4 Application 1

The models are tested on the real recorded measurements from a 6-story hotel building in San Bernardino, CA, USA. The sensors are installed, and the responses from past earthquakes are stored in the CSMIP publicly-available database[3] (Station No. 23287, Fig. 6.6). In addition to the sensors at the ground level, the building has sensors to measure the response at the third floor and roof levels, and their recordings for the North-South horizontal direction are used in this application. The number of records are the same as in the study by Zhang et al. [130], where 11 & 8 cases are used for training and testing, respectively.

For each model described in Section 6.2, the model complexity is controlled by the number of model parameters, e.g., the number of units in the LSTM cells and the number of filters in the TCN network. The model parameters and their investigated values are summarized in Table 6.1. Simpler models tend to have the problem of under-fitting, while complex models tend to have the problem of over-fitting. Some models have several parameters, and varying combinations of these parameters are evaluated and compared. In order to limit the total number of these combinations, a baseline set of model parameters for each model is specified for comparison. For other model parameter sets, only one parameter is changed from the baseline model. This is helpful for evaluating the relative importance of different model parameters. In Table 6.1, the baseline model parameter values are underlined.

Table 6.1: Model design parameters.

Model	**Parameter (Notation)**	**Values**
1-layer LSTM	# Units (U)	<u>30</u>, 50, 100, 500
2-layer LSTM	# Units (U)	<u>30</u>, 50
LSTM w/ Self-attention	# Units (U)	<u>30</u>, 50
	Look Back Range (R)	90, <u>360</u>, 720
LSTM w/ Fixed-attention	# Units (U)	<u>30</u>, 50
LSTM w/ Convolution	# Units (U)	10, <u>30</u>, 100
	Filter Size (S)	90, <u>360</u>, 720
	# Filter (F)	5, <u>20</u>, 50
TCN	Maximum Dilation (D)	128, <u>512</u>, 1,024, 4,096
	Filter Size (S)	10, <u>50</u>, 100
	# Filter (F)	2, 5, 10, <u>20</u>

In the training process, Adam optimizer [53] is used to minimize the training error. The training process started with a learning rate of 0.001, and the rate decay is adopted to reduce the learning rate when the loss stopped decreasing for 500 epochs. The number of epochs is set to 10,000.

[3] `https://www.conservation.ca.gov/cgs/smip`

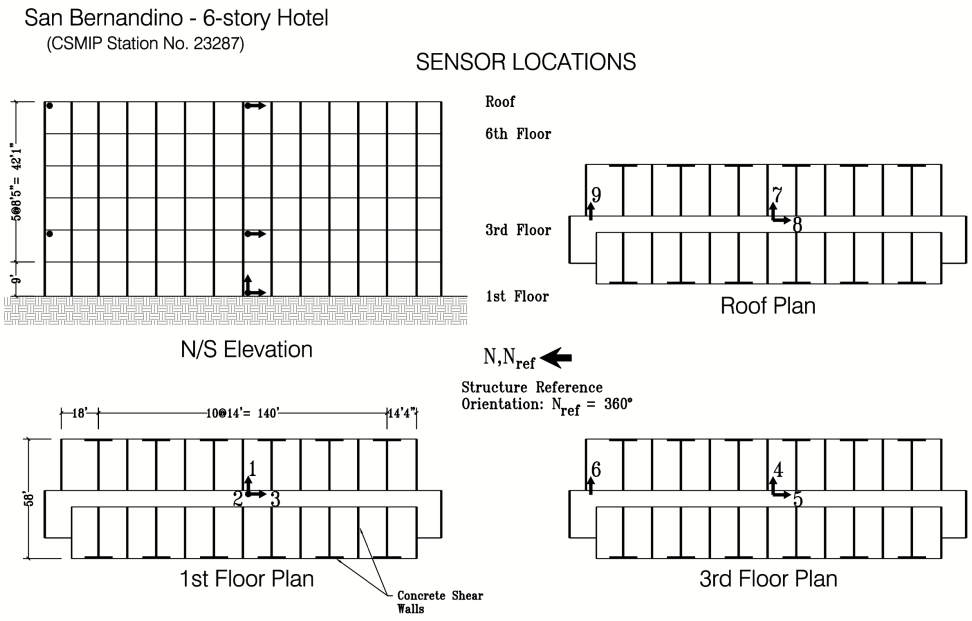


Figure 6.6: San Bernardino 6-story hotel building (CSMIP station No. 23287). Sources: Google Earth (top) and CSMIP (bottom).

6.4.1 Results and Discussion

The results for application 1 are discussed in this section. These include correlation, error distribution, training time, and visualization of key results.

Correlation: The results are shown in Table 6.2. Models other than those marked as

"Baseline" in the column "Parameter," correspond to changing only one model parameter, and the column "Parameter" specifies the parameters and the corresponding values. Bold and underlined numbers correspond to the maximum values of training and test data sets for each model and the maximum values among all models, respectively. Several observations from the results could be drawn as listed below.

- The results are excellent where the training and test sets achieve correlations of 0.99 and 0.97, respectively, for most models. As discussed later, the predicted responses match the true ones pretty well. The results for the training set is better than the test set, as expected. Alternating the model parameters inside each model changes the model complexities, and subsequently affects the results. However, the effect of changing the model architectures (i.e., testing different models) is more significant.

- The LSTM with convolution layers achieves excellent results in both the training and test sets. The model with kernel size of 720 produced the best test set correlation among all models. Therefore, convolution filters help the model learn local trends of the input GM before passing into the LSTM model.

- Vanilla LSTM and TCN also achieve more than satisfactory results. Vanilla LSTM (1-layer or 2-layer) models produce balanced results between the training and test sets, i.e., the difference of results from the training and test sets is small. TCN achieves a high training set correlation (model filter size of 100 produces the best training set correlation). However, in comparison to other models, the test set correlation is not as satisfactory. Therefore, the convolution layers of TCN have enough expressiveness on the seen data. However, in comparison to the LSTM models, TCN lacks the ability to learn the intrinsic temporal correlations of data.

- The LSTM with attention mechanisms has the worst performance. Instead of improving the results over the vanilla LSTM, it worsens the results for both explored attention mechanisms. This result is explained in Fig. 6.7 for the self-attention mechanism, which plots the attention scores for the model with # units = 50 and look back range = 360. The vertical axis is the actual time step t to calculate the final h^* (Eq. 6.1), and the horizontal axis is the time steps t' to calculate the scores. Note that this heat map is a lower triangular because $t' < t$ (recall that h^* is calculated only from the previous steps). It is clearly shown that the scores for different t are the same for a fixed t', i.e., h^* values are identical for different steps, especially for values of $t' \geqslant 60$. Therefore, instead of "sharpening" the final prediction results, it "smoothes" and averages the local results leading to large errors. For the fixed-attention mechanism, only a small set of hand-selected periods are used, and the weights for each of these periods are equal. Therefore, the model is not flexible enough to incorporate all natural periods with different values. In conclusion, the models with attention mechanisms are incapable of improving the long-term memories of the LSTM models.

Table 6.2: Model correlation for application 1.

Model	Parameter	Training	Testing
1-layer LSTM	Baseline	0.9984	0.9849
	$U = 50$	0.9985	0.9862
	$U = 100$	**0.9989**	**0.9876**
	$U = 500$	0.9988	0.9821
2-layer LSTM	Baseline	0.9985	**0.9865**
	$U = 50$	**0.9988**	0.9843
LSTM w/ Self-attention	Baseline	0.9925	0.9349
	U = 50	0.9899	0.8854
	$R = 90$	**0.9950**	0.9197
	$R = 720$	0.9937	**0.9425**
LSTM w/ Fixed-attention	Baseline	0.9949	0.9768
	$U = 50$	**0.9952**	**0.9773**
LSTM w/ Convolution	Baseline	0.9991	0.9874
	$U = 10$	0.9989	0.9843
	$U = 100$	0.9992	0.9868
	$S = 90$	0.9986	0.9858
	$S = 720$	**0.9993**	**<u>0.9881</u>**
	$F = 5$	0.9990	0.9860
	$F = 50$	0.9992	0.9869
TCN	Baseline	0.9998	0.9748
	$D = 128$	0.9992	0.9796
	$D = 1,024$	0.9998	0.9777
	$D = 4,096$	0.9998	0.9765
	$S = 10$	0.9987	**0.9814**
	$S = 100$	**<u>0.9999</u>**	0.9778
	$F = 2$	0.9966	0.9642
	$F = 5$	0.9991	0.9785
	$F = 10$	0.9997	0.9760

Error distribution: Since the error is presented as a distribution in terms of the Probability Density Function (PDF), it is not possible to numerically compare its values. Therefore, only distributions for the two best models identified using the correlations above, namely, TCN with $S = 100$ (which had the largest correlation for the training set) and LSTM with convolution filters with $S = 720$ (which had the largest correlation for the test set), are shown. The results are plotted in Fig. 6.8 and it is observed that the error values are centered around 0.0 and remain within $\pm$ 2%.

Training time: The training time is shown in Table 6.3. As expected, TCN has the

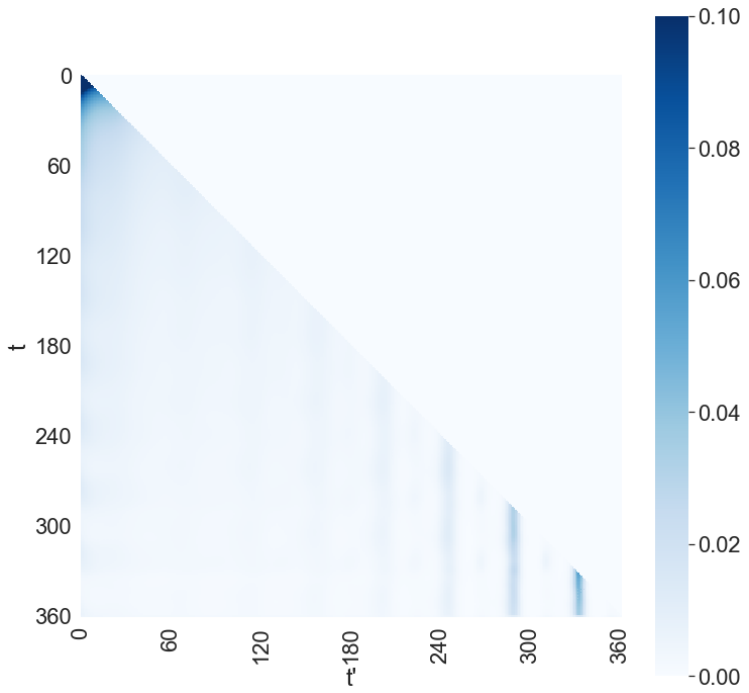


Figure 6.7: Attention scores heat map for the self-attention mechanism for application 1.

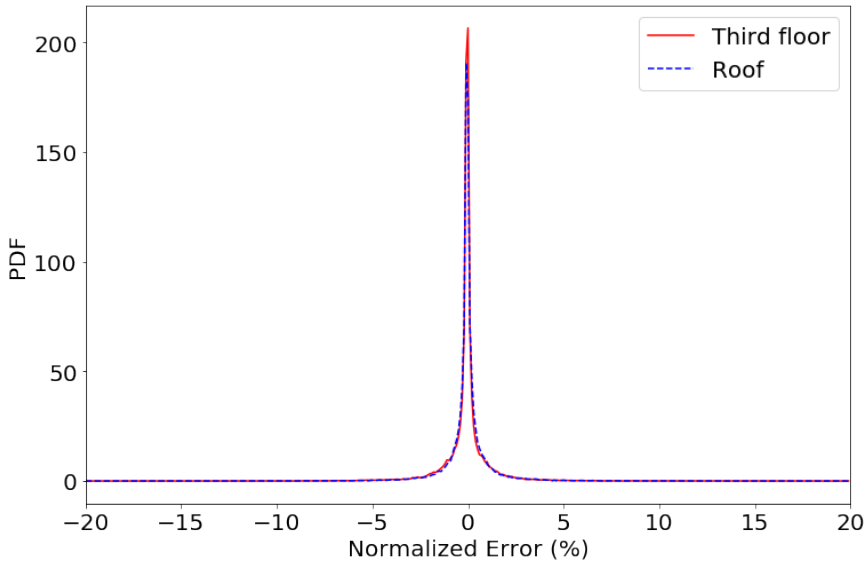
lowest training time (∼1/3 of the training time of other models with baseline parameters). For each model, the training time is dependent on the complexity, e.g., LSTM model with larger # of units will have higher training time. One exception is the LSTM with self-attention mechanism, where models with larger look back ranges have smaller training time. This phenomenon is related to how the mechanism is implemented. Briefly, models with larger look back ranges need less `for` loops over the time ranges. It is noted that the `for` loops are not fully utilizing the parallelized computing resources and therefore, the training time is longer when there are more of such loops.

Visualization of key results: Finally, a visual comparison of the real and predicted responses in the TS is shown in Fig. 6.9 for one test data[4]. The results of the predicted response are obtained from the LSTM with convolution filters ($S = 720$), which produces the highest correlation. It is observed that the real and predicted acceleration responses (in units of g's, where g is the acceleration of gravity) are very close to each other.

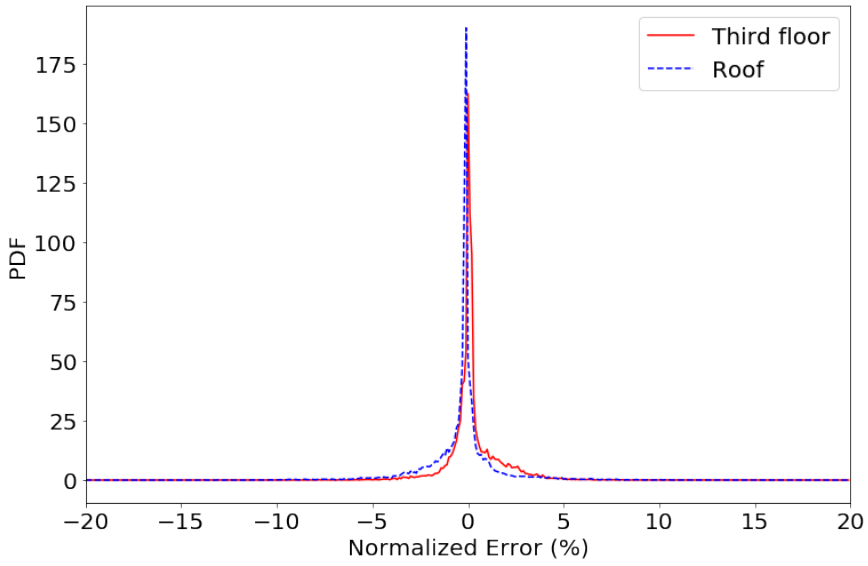
6.5 Application 2

The models described in this chapter are also applied to the FEM model of the Concentrically Braced Steel (CBS) frame described in Section 3.2. The input is obtained from the simulated

[4]Chino Hills Earthquake, July 29, 2008, moment magnitude $M_w = 5.4$.



(a) Training set for TCN ($S = 100$)



(b) Test set for LSTM w/ convolution filters ($S = 720$)

Figure 6.8: Error distributions for two models used in application 1.

GM TS, and the output is obtained from the simulated structural TS at the center of the floors. The correlation results are shown in Table 6.4. Note that, as previously observed in application 1, the attention mechanisms do not produce satisfactory results. For the sake of completeness, only one result for the model with attention mechanism is shown for the baseline model of LSTM with self-attention mechanism. Similar observations as in application

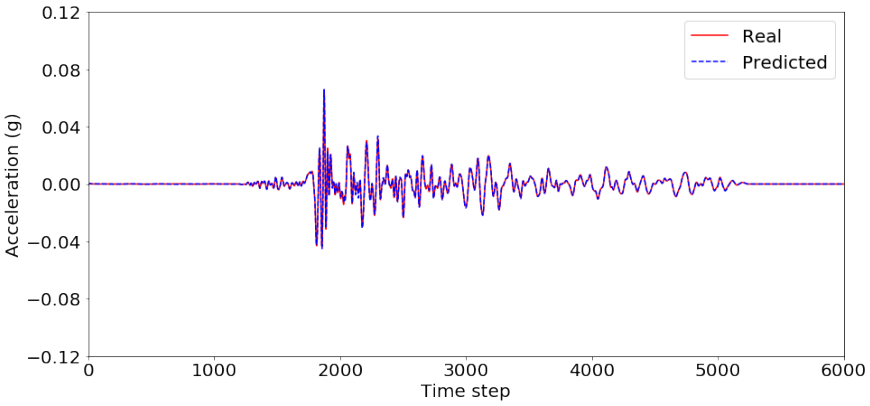
Table 6.3: Model training time for application 1.

Model	Parameter	Time (min.)
1-layer LSTM	Baseline	46.14
	$U = 50$	56.34
	$U = 100$	48.88
	$U = 500$	83.81
2-layer LSTM	Baseline	87.07
	$U = 50$	108.39
LSTM w/ self-attention	Baseline	64.33
	$U = 50$	75.75
	$R = 90$	75.16
	$R = 720$	66.35
LSTM w/ fixed-attention	Baseline	49.51
	$U = 50$	59.92
LSTM w/ convolution	Baseline	44.67
	$U = 10$	38.48
	$U = 100$	47.27
	$S = 90$	46.47
	$S = 720$	44.47
	$F = 5$	46.06
	$F = 50$	46.45
TCN	Baseline	16.91
	$D = 128$	11.78
	$D = 1{,}024$	19.10
	$D = 4{,}096$	23.37
	$S = 10$	14.38
	$S = 100$	18.49
	$F = 2$	10.86
	$F = 5$	11.62
	$F = 10$	15.22

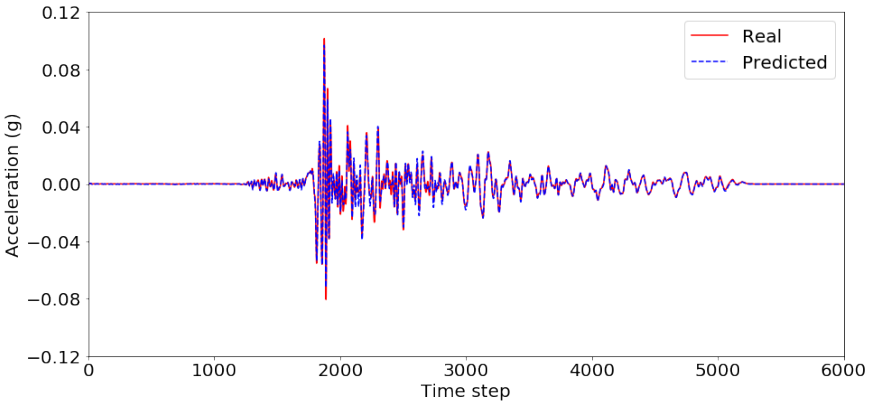
1 are drawn, except that the highest training correlation corresponds to the LSTM model with convolution filters rather than TCN. Since the findings from both applications 1 and 2 are general consistent, the main conclusion of this chapter is that the convolution operation improves the prediction performance of the structural response in the form of acceleration TS.

A visual comparison of the real and predicted responses in the form of the TS is shown in Fig. 6.10 for one test data[5]. The results of the predicted response are obtained from the

[5]Kobe Earthquake, January 16, 1995, moment magnitude $M_w = 6.9$.



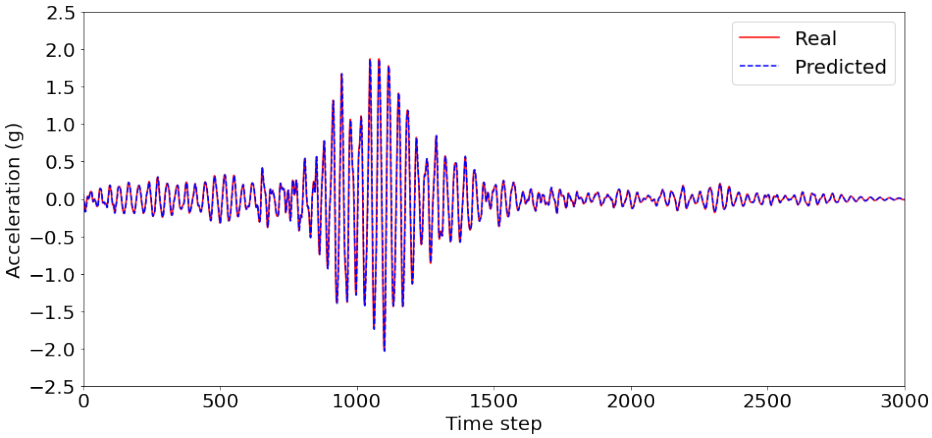
(a) Third floor



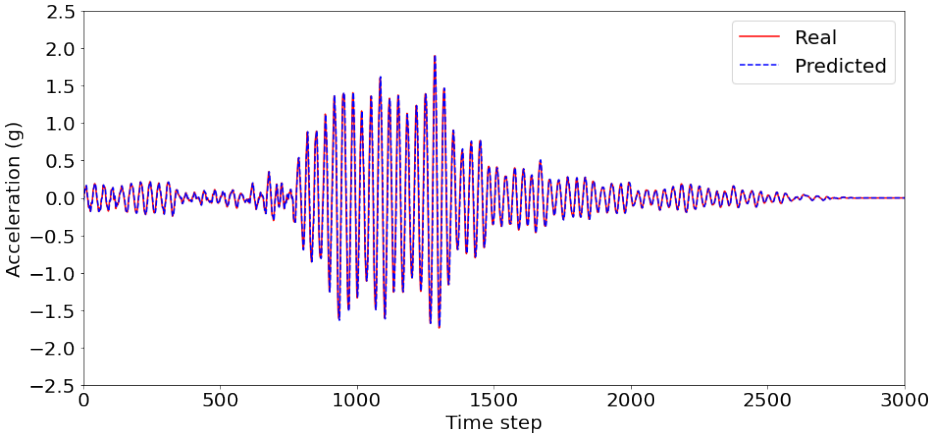
(b) Roof

Figure 6.9: Real vs. predicted responses for test data using the LSTM with convolution filters ($S = 720$) for application 1.

LSTM with convolution filters ($U = 100$). The test case is selected, because of obtained highly nonlinear responses (as seen in Fig. 6.10, the maximum acceleration approaches $2.0g$, in both X and Y directions. This is also seen from the true damage labels as described in Section 5.2). A close match between the real and predicted responses is clearly observed, even though high inelastic response occurs. This justifies the applicability of the model for the nonlinear structural response predictions.



(a) Roof (X-direction)



(b) Roof (Y-direction)

Figure 6.10: Real vs. predicted responses for test data using the LSTM with convolution filters ($U = 100$) for application 2.

Table 6.4: Model correlation for application 2.

Model	**Parameter**	**Training**	**Testing**
1-layer LSTM	Baseline	0.9928	0.9934
	$U = 50$	0.9947	0.9949
	$U = 100$	**0.9961**	**0.9956**
2-layer LSTM	Baseline	0.9963	0.9963
	$U = 50$	**0.9976**	**0.9972**
LSTM w/ Self-attention	Baseline	**0.9470**	**0.9256**
LSTM w/ Convolution	Baseline	0.9989	0.9981
	$U = 10$	0.9961	0.9935
	$U = 100$	**0.9991**	**0.9984**
	$S = 90$	0.9982	0.9979
	$S = 720$	**0.9991**	0.9980
	$F = 5$	0.9982	0.9977
	$F = 50$	0.9987	0.9969
TCN	Baseline	0.9977	0.9857
	$D = 128$	0.9984	0.9893
	$D = 1{,}024$	0.9976	0.9842
	$D = 2{,}048$	0.9973	0.9836
	$S = 10$	**0.9986**	**0.9947**
	$S = 100$	0.9953	0.9653
	$F = 2$	0.9717	0.9477
	$F = 5$	0.9957	0.9806
	$F = 10$	0.9978	0.9868

Chapter 7

Optimal Sensor Placement Algorithm Using Directed Information

7.1 Introduction & Background

In Chapters 4 and 5, a Long Short-Term Memory (LSTM) based Structural Health Monitoring (SHM) framework is proposed, where the health states of example structures are successfully identified. One hidden assumption for such framework is that the Time Series (TS) data acquired by the sensors reflect the real health conditions of the structure. Therefore, the Optimal Sensor Placement (OSP) problem is put forward aiming towards finding a sensor placement plan (i.e., the set of locations to position the sensors). This problem is related to *optimization* and *information theory* and is widely considered in system/model identification where there is a growing interest in its solution for SHM/damage detection. The sensor optimization procedure is vital for smart structural systems, as it allows for the reduction of the cost of sensors (including initial, installation, operational, and maintenance costs), while at the same time, ensuring sufficient level of reliability of the monitoring results.

In this chapter, a novel OSP algorithm under earthquake loading, which is based on the concept of Directed Information (DI), discussed in Chapter 2, is proposed. Apart from such an automatic OSP algorithm, traditional engineering judgement is the usual foundation for designing a sensor placement plan. This engineering judgement and the OSP algorithm are not strictly substitutes of one another. Instead, fusion of expert opinions and algorithms are possible and is typically helpful for improving the performance and efficiency. For example, engineering judgement is helpful for selecting a set of possible locations where sensors could be installed (the set of locations selected by a human expert is usually in a grid pattern, i.e., intersections of important structural components). Therefore, such engineering judgement is helpful to limit the number of available choices for the application of the OSP algorithm and subsequently improves the efficiency and accuracy of the algorithm. In this case, the role of the OSP algorithms is to choose a subset of such locations. Given the number of possible locations n and the total number of sensors to be placed $m \leqslant n$, the total possible number

of combinations is $C_n^m = n!/(m!(n-m)!)$. Even though the possible locations are already prescribed by engineering judgement, an exhaustive search over all possible sensor placement combinations is computationally demanding. The optimization algorithm is needed to find a sub-optimal plan close to the optimal one found through an exhaustive search.

The earliest work on OSP dealt with the System Identification (SI) problem with implementation of OSP focused on modal identification. Numerous studies addressed this topic where the main differences among these studies are the choices of objective functions. According to the best of the author's knowledge, the earliest work is attributed to Kammer et al. [49] where the basic idea is to place sensors on locations such that different structural modes are less intertwined. Given a set of dynamic modes to be identified, sensors should provide measurements such that the extracted mode shapes and frequencies are linearly independent. A *Fisher Information Matrix* (FIM [27]) is used, and the sensors that did not contribute substantially to the linear independence are progressively removed. Similar studies on utilizing FIM is also found in [50][123]. It should be noted that in [123], the authors used Genetic Algorithm (GA), where the function to be optimized is taken as the determinant of FIM. Details of GA can be found in Chapter 3. The trace of FIM is also used as the target function [116], with the idea to place sensors at locations whose displacement/velocity/acceleration are most sensitive to changes of the mode shapes. In the study by Yi et al. [124], the sensors are selected such that the measured modal vectors are as orthogonal as possible. This is achieved by formulating a Modal Assurance Criterion (MAC) matrix, and forcing the matrix to be diagonal. Carne & Dohrmann [13] also used the MAC matrix, and the value to be optimized (minimized) is the maximum off-diagonal term. Papadimitriou et al. [89] and Yuen et al. [126] used an *information entropy*, which quantifies the structural parameter uncertainties, and the sensors are placed on locations where such uncertainties are minimized. In the study by Heo et al. [41], the sensors are placed on locations such that the *modal kinetic energies* picked up by the sensor are optimized. Li et al. [63] adopted a similar approach, and ranked all candidate sensor positions by their modal kinetic energies. Apart from the objective function mentioned above, there also exist other functions. Hac & Liu [40] proposed a *performance index*, which is expressed in terms of the eigenvalues of an *Observability Gramian* (a term that is proportional to the output energy released by the system). Penny et al. [92] used what is called *average driving point residue* as the objective function. In terms of the optimization algorithms, apart from the GA mentioned above, other algorithms are also adopted, including Monkey Algorithm [125], Particle Swarm Optimization [64], Firefly Algorithm [132], and Artificial Bee Colony algorithm (with modifications for discrete optimization) [111].

The number of studies of OSP for damage detection is far less than that for SI. The most common sign of damage is the change of the natural frequency and mode shape. Therefore, in the work by Cobb & Liebst [25], the locations of the sensors that are most sensitive to the change of modal parameters are chosen. In the work by Worden & Burrows [120], the damage is simulated, and Neural Network (NN) is trained to output the level of damage. The optimal sensor locations are chosen such that the errors from the NN are minimized. In that case, an optimization algorithm (e.g., GA) is adopted. A similar approach is adopted in the work

by Field & Grigoriu [30], where the sensor locations are optimized such that several criteria (e.g., the false positive rate) are below specified thresholds. Flynn & Todd [31] adopted a similar approach, except that the criteria are quantified in a Bayesian probabilistic setting. Azarbayejani et al. [8] adopted a similar probabilistic approach, and the method allocated sensors to places that have the highest probability of detecting the damage.

For most of the studies discussed above, models based on the Finite Element Method (FEM) are required to provide input data to the proposed OSP algorithms. This demonstrates the need for a *digital twin* of the real structure (the concept of digital twin is discussed in Chapter 4) when dealing with the OSP problems. For example, in the work by Sun & Büyüköztürk [111], the proposed algorithm is conducted on the Canton Tower, Guangzhou, China, and a full scale FEM model that is created using the software ANSYS [5] is used. In the work by Worden & Burrows [120], different damage states are simulated by making adjustments to the FEM model. In the trained NN for damage identification, the simulated responses are used as the data, and the prescribed damage states are used as the labels.

The study of DI is gaining attention in the scientific community, where DI is used as a method for feature selection. Rao et al. [96] used the DI for selecting features in classifying tissue-specific genomic regions from those that are not tissue-specific. There also exist studies in structural engineering, e.g., Zheng et al. [131] used the DI to identify the causal relationship of free vibration from sensors. In their work, the environmental and nearby effects, e.g., passing of nearby trains, on the causal dependencies are also studied. However, the literature on SHM in conjunction with OSP is limited. Observing the *causal relationship* between two locations is not only helpful for identifying the optimal layout of sensor arrays, but is also helpful for understanding the structural system itself, e.g., understanding the displacement constraint relationships among different structural elements and joints. This will be more obvious using an illustrative example in the next section.

7.2 The OSP Algorithm

In this section, several key ideas of the proposed algorithm is described, and the final algorithm is shown at the end. In this study, the different directions (i.e., horizontal-1 (X), horizontal-2 (Y), and vertical (Z)) at one possible sensor location are treated as different "locations." In other words, each direction at one location is treated separately to determine if a sensor needs to be installed to record the acceleration in this direction. Unless otherwise specified, the locations referred to in this study are the "locations" in the broad sense as described above (locations + directions). Moreover, each sensor is assumed to only measure the acceleration in one direction. Therefore, the number of "locations" is equal to the number of sensors. In practice, if accelerations of several directions need to be measured at one location, a multi-directional accelerometer is used.

7.2.1 Calculating DI in TS Analyses

In the proposed algorithm, the DI between two possible locations should be calculated using the acceleration TS from two locations. However, as discussed in the example in Chapter 2, obtaining the analytical solution of DI is impractical. There are two main reasons: (1) the Probability Density Function (PDF) of the given TS is hard to obtain, and (2) given the PDF, the DI is difficult to compute using the definitions in Chapter 2, Eq. 2.26. Therefore, an approximate method to estimate the DI is proposed by Jiao et al. [46] and also adopted in this study. Two approximations are adopted in their work, corresponding to the two reasons above: (1) The PDF is estimated through Context Tree Weighting [119] for the TS; and (2) The approximate DI (that is close to the value computed using the definition of DI in Chapter 2, Eq. 2.26) is computed using the approximate DI based on the above estimated PDF.

The setting of the approximate method to calculate the DI requires that the TS values should be discrete, rather than continuous, in terms of the possible values (i.e., the TS values could be any real number[1]). Therefore, in this study, the TS values should be heuristically discretized through n bins. The maximum and minimum values of each TS are obtained, and bins with equal lengths of ranges are selected (i.e., for each bin, the difference between the upper and lower boundary values is equal). The bins are numbered from 1 to n. In the TS, each value is assigned to a bin where this value is within the range of the upper and lower boundary values of the bin. Therefore, the TS of the original values are converted into the TS of the bin numbers. The process is shown in Fig. 7.1. As an example, suppose for a TS, the maximum and minimum values are p_{max} and p_{min}, respectively, with $R_p = p_{max} - p_{min}$. Then, the i^{th} bin corresponds to the range $\left[p_{min} + \frac{i-1}{n}R_p, p_{min} + \frac{i}{n}R_p\right]$, and the TS values within this range are converted into the same scalar value i.

7.2.2 Example of Calculating the DI

Below is a basic example of the calculation of DI for a highway overcrossing bridge to identify the causal relationship and displacement constraint relationships of different locations. The bridge of interest is Jack Tone Road On-Ramp Overcrossing (Fig. 7.2), which is built in 2001 and located in Ripon City, San Joaquin County, CA, at the intersection of Route 99 & Jack Tone Road. The FEM model of the bridge is developed using OpenSees[2] [51], Fig. 7.2. The column-bents, superstructure (which consists of the bridge deck and the cap-beam), and seat-type abutments are explicitly modeled. The Ground Motion (GM) input for this simulation corresponds to the earthquake that is occurred at Iwate, Japan, in 2008, with moment magnitude $M_w = 6.9$, recorded at station AKT013. The accelerations of nodes are

[1]In the setting of this study, the original TS is not strictly "continuous," because the values in the TS are the sampled ones over the time. The TS values are "discrete" in terms of the sampling procedure, but "continuous" in terms of the possible values. See Fig. 7.1 for an explanation.

[2]Open System for Earthquake Engineering Simulation, `https://opensees.berkeley.edu/`, is the computational platform developed at the Pacific Earthquake Engineering Research (PEER) Center.

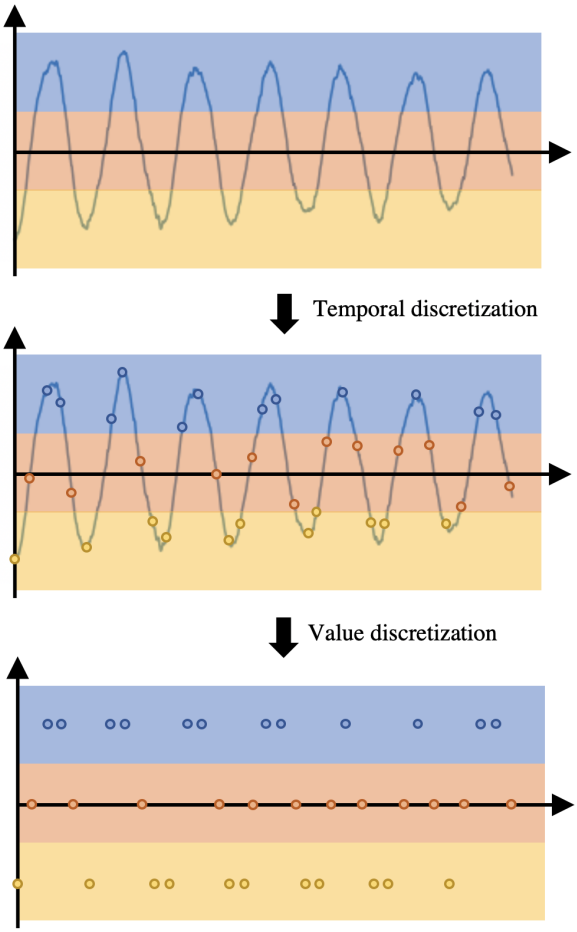


Figure 7.1: Discretization of the TS data.

recorded as TS in the two horizontal directions (X: longitudinal, and Y: transversal, refer to Fig. 7.2).

For simplicity and because of symmetry along both the X and Y directions, only the acceleration responses of nodes 100, 105, and 110 (see Fig. 7.2) are studied. Both the X and Y direction accelerations are used at node 110 with corresponding TS denoted as $110x$ and $110y$, respectively, while only the X-direction accelerations are used at nodes 100 and 105 with corresponding TS denoted as $100x$ and $105x$, respectively. Considering that the bridge is mostly symmetric about the X and Y axes leading to minimum torsional response, it is intuitive that the accelerations in the X and Y directions are approximately independent of each other. Moreover, the accelerations in the X-direction for the nodes along the girder are close to each other because the girder is rigid along the longitudinal direction, i.e., the girder is not elongating or shortening along the axis of the girder. Therefore, the causal relationship between $110x$ and $110y$ is expected to be small, while those between pairs among $100x$, $105x$,

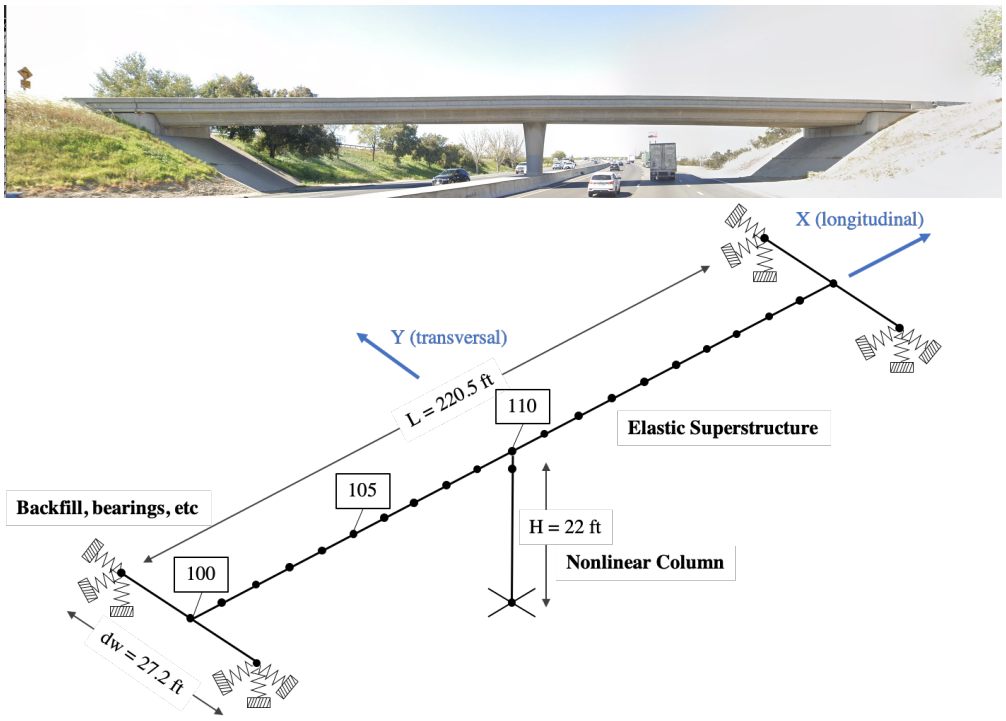


Figure 7.2: Jack Tone Road Overcrossing Bridge (top, from Google Map) and the corresponding FEM model (bottom).

and $110x$ are expected to be large.

The results for $110x \to 110y$, $110x \to 100x$, and $110x \to 105x$ are shown in Table 7.1. Recall that, as mentioned in Chapter 2, the DI highly depends on the length of the TS, i.e., the DI is in general larger if the TS is longer. Therefore, in Table 7.1, the results for the averaged DI are shown. As expected, the DI of $110x \to 110y$ is small (close to zero), while the DI of $110x \to 100x$ and $110x \to 105x$ are large. Moreover, the DI of $110x \to 100x$ is (slightly) smaller than that of $110x \to 105x$ because node 105 is closer to node 110. It is intuitive to understand that closer nodes along the girder have more "similar" responses. Therefore, the DI between two closer nodes is higher.

Table 7.1: The DI values for Jack Tone Road Overcrossing Bridge.

Causal Relationship	**Value**
$110x \to 110y$	0.0890
$110x \to 100x$	0.3596
$110x \to 105x$	0.3603

7.2.3 Ideas

In this subsection, several ideas for constructing the final OSP algorithm are described. These ideas are related to the feed-forward location selection, the DI related to sensor locations, and the DI related to sets of GM TS.

Idea 1. **Feed-forward location selection**: The basic idea of the proposed OSP is similar to the forward step-wise feature selection method, described in Chapter 2. The new "feature" to be selected at each iteration is the location of the next sensor to install. In the algorithm, the first location could be selected randomly, or preferably selected by the input of human experts. Starting from the second location, the candidate locations are the set of locations that are not previously selected. In other words, suppose the selected locations formed a subset R of the whole set of locations S, then the subset of candidate locations is the complement of the selected locations in the whole set, i.e., $S - R$. The new location is selected in this subset such that the DI for the sensor locations, Eq. 7.1, is minimized.

Idea 2. **DI related to sensor locations**: In this study, the DI quantifies the amount of new information available by observing the new sensor recording. To simplify this concept in this study, the total *relative* amount of new information available given a subset of existing sensors is quantified as the summation over the DI from existing sensors to the new sensor.

Idea 3. **DI related to sets of GM TS**: The proposed OSP algorithm applies in particular to the structures under seismic loads. However, there are large variances of obtaining the DI from a single GM because of the randomness of the GM, and using only the DI from one GM is biased. Therefore, the TS results from a set of GM should be collected, and a summed[3] DI value over the set of GM is obtained. Summing over a set of GM when calculating the DI could reduce the randomness and variance from the GM and obtain a more accurate result to be used for selecting the sensor locations.

Based on ideas 2 & 3 above, the DI for each candidate sensor location i is quantified as:

$$DI(i) = \sum_{j \in R} \sum_{GM} DI(j, i), \tag{7.1}$$

where the first summation corresponds to the summation of locations j in the current subset R of locations already selected, and the second summation corresponds to the summation of the GM cases. The next selected location is based on minimizing the value calculated using Eq. 7.1. If the DI is the smallest, then the amount of new information available thorough adding the new sensor is the highest. In other words, the information duplication from the new sensor compared to the already selected sensors is the lowest. Such heuristic pushes the wide distribution of sensors to cover distinct locations and directions, which is helpful to comprehensively monitor the health state of the structure at hand.

[3]The DI can also be an averaged value where indeed the relative value, rather than the absolute value, matters, as explained later.

7.2.4 The OSP Algorithm

Details of the OSP algorithm are discussed in this subsection. Moreover, the implementation of the algorithm in MATLAB [74] is also presented in Listing 7.1.

- **Input**: (1) Simulations from multiple GM into TS responses; (2) Set of all possible locations, S, for sensor placement; (3) Number of expected sensors, n; and (4) Selection of the first sensor location, a, from the set of possible locations and treat as the current subset, R, of selected locations, i.e., $R = \{a\}$.
- **Output**: Subset of locations, R, selected where the sensors are to be placed.
- **Algorithm**: (1) For each location i not in the selected subset (i.e., $i \in S-R$), calculate $DI(i) = \sum_{j \in R} \sum_{GM} DI(j,i)$. (2) Find the location k with the lowest $DI(k)$ among the locations in $S-R$, i.e., $k = \arg \max_{i \in S-R} DI(i)$. (3) Append the location k to the current subset, i.e., $R \leftarrow R \bigcup \{k\}$. (4) Iterate the above process, until the number of elements in the subset R reaches n.

```
% Initialize an array to store the computed averaged DI between
% two locations to avoid repetitive calculations
dis = zeros(num_loc, num_loc);
% Initialize the list of selected locations;
% location 15 is selected manually to start the algorithm
curr_list = [15];
% Iterate over all possible locations
for i = 1: 32
    di_gain_max = Inf;
    % Initialize the new location to be added
    loc_add = 0;
    % Iterate over locations not selected
    for j = 1: num_loc
        % Skip the locations already selected
        if (any(curr_list == j))
            continue;
        end
        % Initialize the DI(j), which will be calculated through
        % summation over locations
        di_gain = 0;
        % Compute the DI between potential and selected locations
        for k = curr_list
            if (dis(k, j) == 0)
                % loc{k} and loc{j} are the TS for all GM cases at
                % locations k and j, respectively;
                % num_GM is the number of GMs
                % average_directed_info is defined below
                dis(k, j) = average_directed_info(loc{k}, loc{j}, num_GM);
            end
            % Sum over the locations to calculate DI(j)
```

```
            di_gain = di_gain + dis(k, j);
        end
        % Select the new location with minimum DI(j)
        if (di_gain < di_gain)
            loc_add = j;
            di_gain_max = di_gain;
        end
    end
    % Append the new selected location to the list of selected locations
    curr_list(end + 1) = loc_add;
end

% Function to compute averaged (summed) DI over GMs
function avg_di = average_directed_info(loc1, loc2, num_GM)
    % loc1: List of TS for GMS for the first location
    % loc2: List of TS for GMS for the second location
    % num_GM: number of GMs
    % Store the DIs for GMs as an array
    di = zeros(num_GM, 1);
    % Iterate over GMs
    for i = 1: num_GM
        % Compute DI between two locations
        B_DI = compute_DI(loc1{i}.', loc2{i}.');
        data_length = length(B_DI);
        % The DI needs to be divided by the length of TS;
        % refer to the final remark in Section 2.3 following Eq. (2.29)
        di(i) = B_DI(data_length) / data_length;
    end
    % Average over DIs computed for GM cases
    avg_di = mean(di);
end
```

Listing 7.1: Listing of the OSP algorithm as implemented in MATLAB.

7.3 Application

In this section, the OSP algorithm described above is applied on the model using FEM of the Concentrically Braced Steel (CBS) frame described in Section 3.2. First, the sensor layout plan is demonstrated and justified qualitatively. Then, the results are validated quantitatively through the results from the three SHM classification tasks identified in Chapter 5 (i.e., overall structural damage diagnosis, damage localization, and local damage pattern/severity identification). The quantification is computed using a two-step approach as follows: (1) A trained simple Machine Learning (ML) algorithm, which uses the Cumulative Absolute Velocity (CAV) [80] of the locations where sensors are to be placed as the features; and (2) A trained LSTM Encoder-Decoder model, which uses the TS of selected sensor locations.

7.3.1 Problem Setting

Given the CBS frame, the sensors are to be placed on locations such that the damage from all floors 1, 2 & 3 and both directions X & Y are detectable. The total number of GM cases where inelastic responses are recorded is 266 out of all applied cases of 596. The set of all possible locations of sensors are shown in Fig. 7.3, where these locations are marked with red dots, and the directions are indicated with red arrows. In the X-Z elevation view, the circles that enclose the red dots indicates the Y-direction. The locations selected are beam-column joints and column bases in the two X and Y directions. Moreover, the sensor locations in the same floor are placed on different bays of the frame where two diagonally opposite exterior and interior columns are candidates for instrumentation. Note that, X and Y directions are considered as separate "locations" in Fig. 7.3, i.e., $n = 4$ floors (including base) $\times$ 4 joints/floor $\times$ 2 directions $= 32$ possible locations are selected.

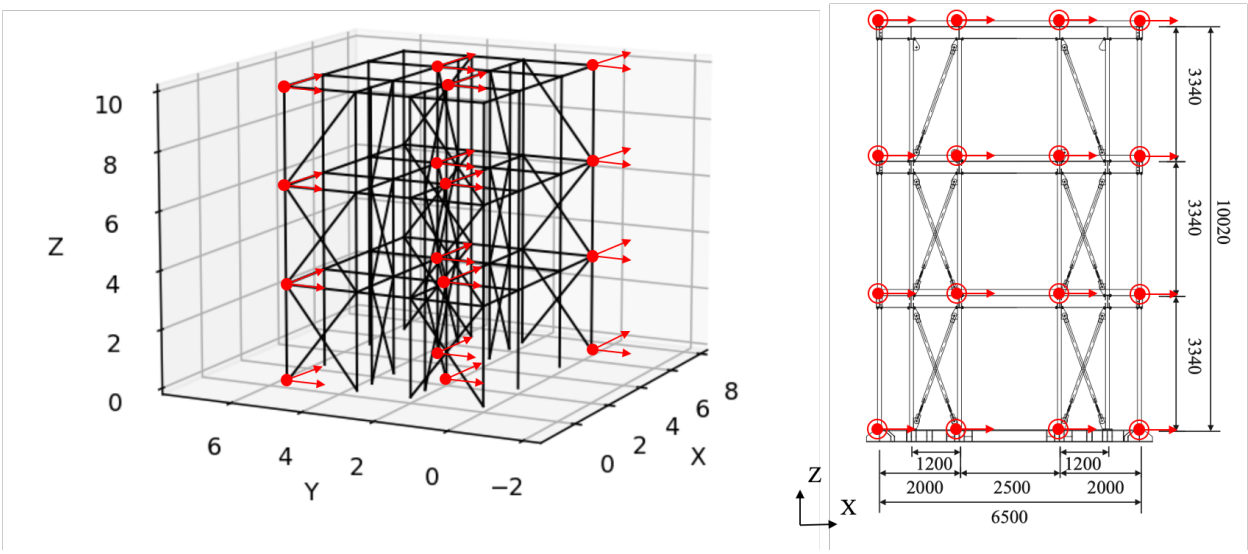


Figure 7.3: Locations of possible sensors: spatial view (left) and $X - Z$ elevation (right).

7.3.2 Result of Selected Locations

The first 9 locations (including directions) selected by the algorithm are shown in Fig. 7.4, where they are marked with red dots and arrows, as in Fig. 7.3. It is noted that the identification of the number of sensors (9 in this case) is part of the selection process, where this number is not a constraint input to the OSP algorithm. From Fig. 7.4, one observes that the joints at the left corner of the spatial view are favored because when the DI of several nodes are equal, the node with a smaller node number is selected. In the FEM model, the nodes are numbered by increasing coordinates first in the X-direction and then

in the Y-direction. Since the structure is symmetric, the DI of joints on the left and right corners are equal, making the algorithm selects the joints in the left corner.

Qualitatively, the results is satisfactory and follows intuitive expectations. The algorithm selects locations from all floors (including the base). This is helpful for identifying damage from all floors. Moreover, the algorithm selects both directions, which is effective for identifying damage in these two directions.

The sequence of the selected sensor locations is numbered in Fig. 7.4 (recall that the OSP algorithm selects these locations progressively and the first (#1) sensor, i.e., the location at the top of the structure, is manually selected to initialize the algorithm). This sequence is important because, if fewer sensors are to be selected, the algorithm picks the first few locations in the sequence. In addition, the sequence is used to observe the relationship between the model SHM performance and the number of sensors in the initial validation process, described below.

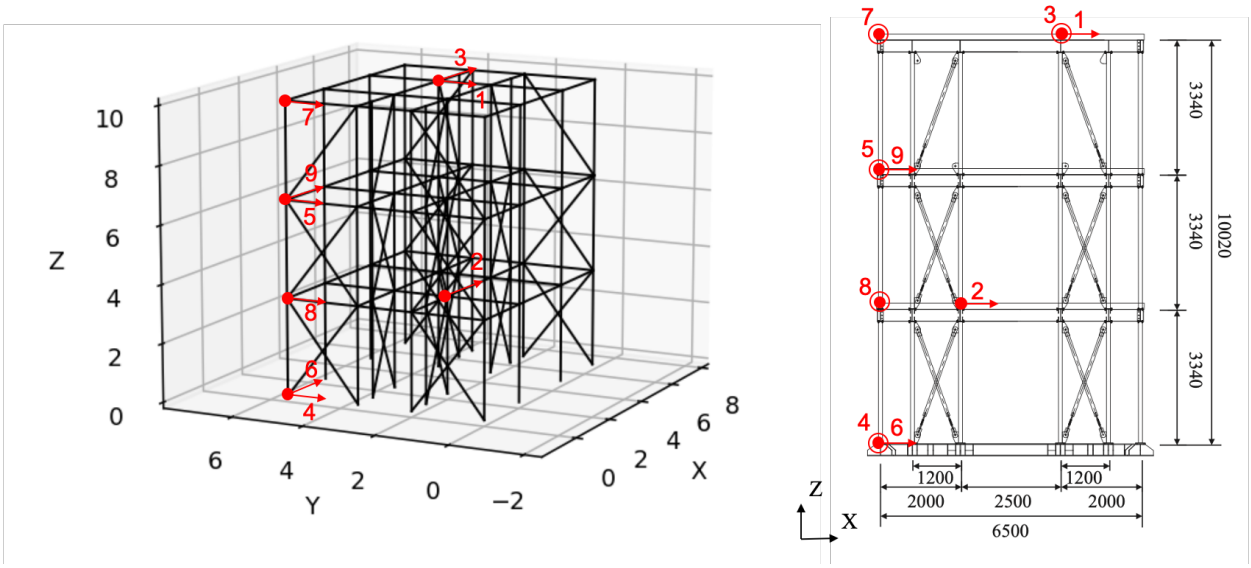


Figure 7.4: Sequence of selected sensor locations: spatial view (left) and $X - Z$ elevation (right).

7.3.3 Initial Quantitative Validation Using CAV in ML

A simple computer experiment to validate the OSP algorithm using simple features from the sensor locations to train ML models is developed. The classification results of the three SHM tasks identified in Chapter 5 (i.e., overall structural diagnosis, damage localization, and local damage pattern/severity identification) are obtained. The adopted features here

are based on the Cumulative Absolute Velocity (CAV) [80][97], defined as follows:

$$CAV = \int_0^{t_{\max}} |a(t)|dt, \tag{7.2}$$

where $a(t)$ is the acceleration TS, and $t_{\max}$ is the duration of the record. In early studies [97], the CAV is defined on the GM TS. More recent studies (e.g., [80][66]) including this dissertation, the CAV definition is extended for recordings from accelerometers used for SHM. The CAV at locations, where sensors are placed according to the OSP algorithm, are calculated and used as features for initial validation of the algorithm.

This validation is not only helpful for initial estimation of the three classification results identified in Chapter 5, but is also helpful for observing the relationship between classification accuracy and the number of sensors. Recall that for each location where sensor is to be installed, one CAV result as a feature is calculated. Therefore, the number of selected sensors is equal to the number of features. Gradually increasing the number of sensors is equivalent to gradually increasing the number of features in the ML model. Therefore, the improvement of accuracy of the ML models due to increasing the number of sensors could be observed. The accuracy from cross-validation is also discussed. Cross-validation is widely used in ML to assess the results of the ML models. In the cross-validation, the data is randomly (and preferably equally) split into a fixed number of folds (denoted as k, and selected as $k = 5$ in this dissertation) and k ML models are trained, where $k - 1$ folds of data are used as training sets, and the remaining one fold of data is used as a test set, refer to Fig. 7.5. The accuracy is obtained for each of the k models, and the final accuracy is the average of those from the k models. The Support Vector Machine (SVM) algorithm is used to train the ML models.

The results for the three SHM classification tasks when 9 sensors are installed (i.e., 9 features) are listed in Table 7.2. Moreover, the corresponding results when sensors are installed at all the possible locations (i.e., 32 features) are listed for comparison. As shown in Table 7.2, the results are satisfactory where those with 9 features are, in general, similar or slightly lower than those with 32 features. The relationship between the accuracy and the number of sensors is shown in Fig. 7.6. This plot is obtained by adding locations sequentially (following the sequence described in the OSP algorithm results in Fig. 7.4), and iteratively training the ML models to obtain accuracy results through cross-validation for Task 1. In Fig. 7.6, the dashed line corresponds to the case when the sensors are installed at all possible locations. As expected, as the number of sensor increases, the accuracy increases. Moreover, for 9 sensors, the accuracy is almost equal to the corresponding result when the sensors are installed at all 32 possible locations. This observation demonstrates that, in the latter case of 32 sensors, there is information redundancy in the sensors for ML classifications. The results are also compared against the cases when random 9 locations are selected. Cross-validation results for 10 such cases are obtained with statistics listed in Table 7.3, where superior performance is observed for the OSP algorithm in comparison to the statistics of these 10 random cases.

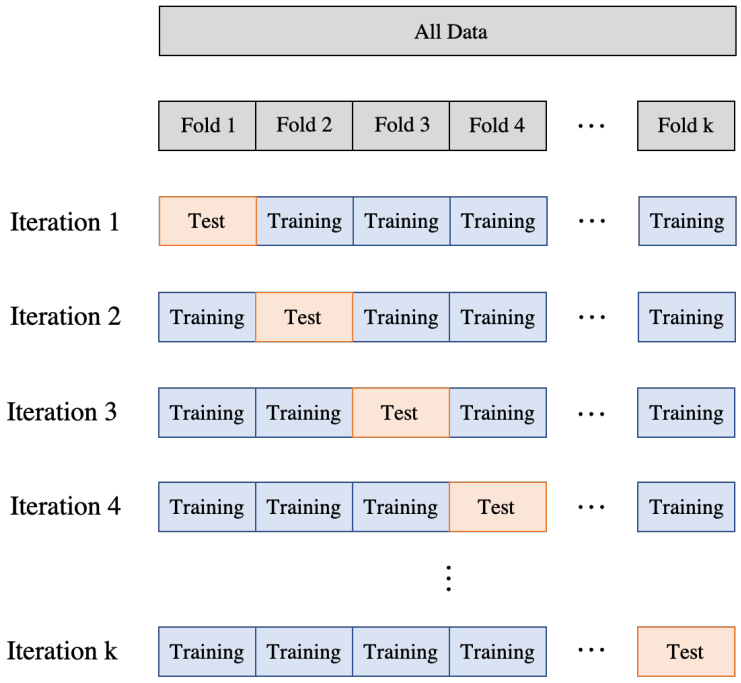


Figure 7.5: Demonstration of the cross-validation implementation.

Table 7.2: Accuracy (%) for the initial validation of the OSP algorithm using CAV.

Tasks		9 features	32 features
Task 1		80.6	82.9
Task 2	Floor 1	81.9	84.6
	Floor 2	84.3	84.3
	Floor 3	81.6	83.6
Task 3	Floor 1	77.2	76.1
	Floor 2	74.9	74.8
	Floor 3	76.2	79.2

7.3.4 Final Quantitative Validation Using LSTM

In this subsection, the OSP algorithm is finally validated using the 1-layer LSTM network proposed in Chapter 4. The TS from the first 9 sensors selected above are used to train the model. The training process of the LSTM network and the corresponding hyper-parameters are exactly the same as those in Chapters 4 and 5. Therefore, they are not stated here for brevity. The results from the LSTM with # of units = 50 is listed in Table 7.4, where also the results from the initial validation using CAV in ML are listed for comparison. It is observed that, in general, the final validation results are better than those from the initial

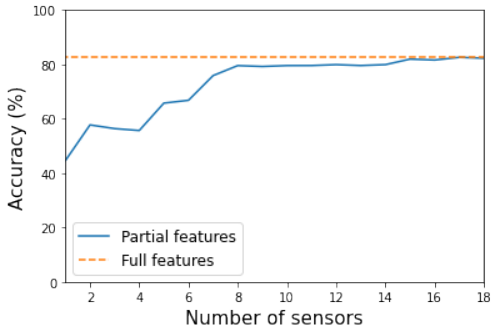


Figure 7.6: Variation of the initial validation accuracy vs. number of sensors (Task 1).

Table 7.3: Statistics of initial validation accuracy (%) using CAV in ML of 10 randomly selected cases of 9 sensor locations (Task 1).

Statistics	**Values**
Mean	75.5
Standard Deviation	2.9
Maximum	81.2
Minimum	71.5
OSP algorithm	**80.6**

validation.

Table 7.4: Accuracy (%) for the final validation of the OSP algorithm using LSTM compared to that of the initial validation using CAV in ML (9 sensors).

Tasks		**LSTM**	**CAV in ML**
Task 1		89.1	80.6
Task 2	Floor 1	90.3	81.9
	Floor 2	89.4	84.3
	Floor 3	90.0	81.6
Task 3	Floor 1	87.0	77.2
	Floor 2	85.6	74.9
	Floor 3	84.8	76.2

Chapter 8

Regional Post-Earthquake Reconnaissance

8.1 Introduction

As mentioned in Chapter 1, regional post-earthquake reconnaissance is important to enhance the understanding of the performance of the built environment, speed up the recovery, and make informed decisions related to current and future hazards. Recently, the use of Artificial Intelligence (AI) methods in natural hazards reconnaissance is gaining attention. For example, Mangalathu & Burton [72] trained a Long Short-Term Memory (LSTM) model for classifying building damage using text-based natural language damage descriptions according to the green, yellow, and red tagging categories of ATC-20 [26]. Moreover, social media data such as tweets from Twitter [115] have been used in a few studies to train Machine Learning (ML) algorithms for direct eyewitness messages in disasters [127] and to identify themes in social media behavior during hazards [110]. However, according to the best of the author's knowledge, the study on automatic reconnaissance by incorporating ML methods and public information is very limited. Considering that this is an almost untapped application field of ML, the preliminary study conducted herein is expected to lead to the initiation of future advances in this area.

In this chapter, the data that are highly related to the earthquake reconnaissance, is collected. Based on the collected data, two studies are conducted in the context of earthquake reconnaissance. The first study is the automatic generation of reconnaissance briefings, which is an essential part of field reconnaissance. Automatic briefing generation aims at decreasing the time to generate a briefing and increase the accuracy and abundance of information by facilitating access to many identified resources that can be otherwise missed. The second study is the use of social media to extract information related to earthquake consequences and resilience, such as *recovery time*, which is difficult to be obtained using other methods.

8.2 Automatic Data Collection

Data collection is the preliminary process for automatic report generation and quantification of resilience. When a new earthquake is reported, the official information is downloaded and stored on the server of the Pacific Earthquake Engineering Research (PEER) Center. There are two categories of the collected data. These are discussed in the following two subsections.

8.2.1 Category 1 of Collected Data

The first category of the collected data is the official information about the earthquakes. After an earthquake occurs, the United States Geological Survey (USGS) records the key information, including, but not limited to, the date and time, the magnitude, and the epicentral location (in longitude and latitude), Fig. 8.1. Moreover, USGS provides other descriptions of the earthquake, such as the tectonic summary of the site, and a preliminary estimations of the economic losses and fatalities using Prompt Assessment of Global Earthquake for Response (PAGER). The PAGER estimations are expressed as the probabilistic distributions of corresponding values within ranges, Fig. 8.1. Such objective information is important, and it is placed at the beginning of the earthquake briefings.

USGS provides an earthquake hazard Application Programming Interface (API)[1]. A Python script is developed to communicate with this API where only earthquakes that have moment magnitude $M_w \geqslant 5$, and a USGS tagged PAGER alert level in either yellow, orange, or red, are recorded. The PAGER alert level is a summary of the PAGER estimations, and it provides a general measure of the earthquake severity to the public. If the PAGER alert level is closer to red, the earthquake is more destructive. The script is scheduled to run daily on the PEER server, and query new earthquakes from USGS API.

Information of new earthquakes about the date and time, the magnitude, and the epicentral location, etc., is returned in the format of Comma Separated Values (`csv`) files. The intensity measure (which includes Peak Ground Acceleration (PGA), Peak Ground Velocity (PGV), and quantitative intensity, are displayed as the intensity map shown in Figure 8.2). Moreover, the tectonic information, and the PAGER estimations of fatalities and economic losses (Fig. 8.1), are stored on the USGS website for the specific earthquake, and such information is obtained through *web scraping*[2]. The specific website to execute the scraping is obtained from the `csv` file described above. For example, the link for the intensity map shown in Fig. 8.2 is obtained from the `csv` file, and the image is downloaded and stored as files in the corresponding format (e.g., the intensity map is stored as image in `jpg` format, and the tectonic information is stored as text in `txt` format).

[1] `https://earthquake.usgs.gov/earthquakes/feed/`

[2] Web scraping is the process of using a bot (i.e., a software application that runs automated tasks over the Internet) to extract content and data from a website.

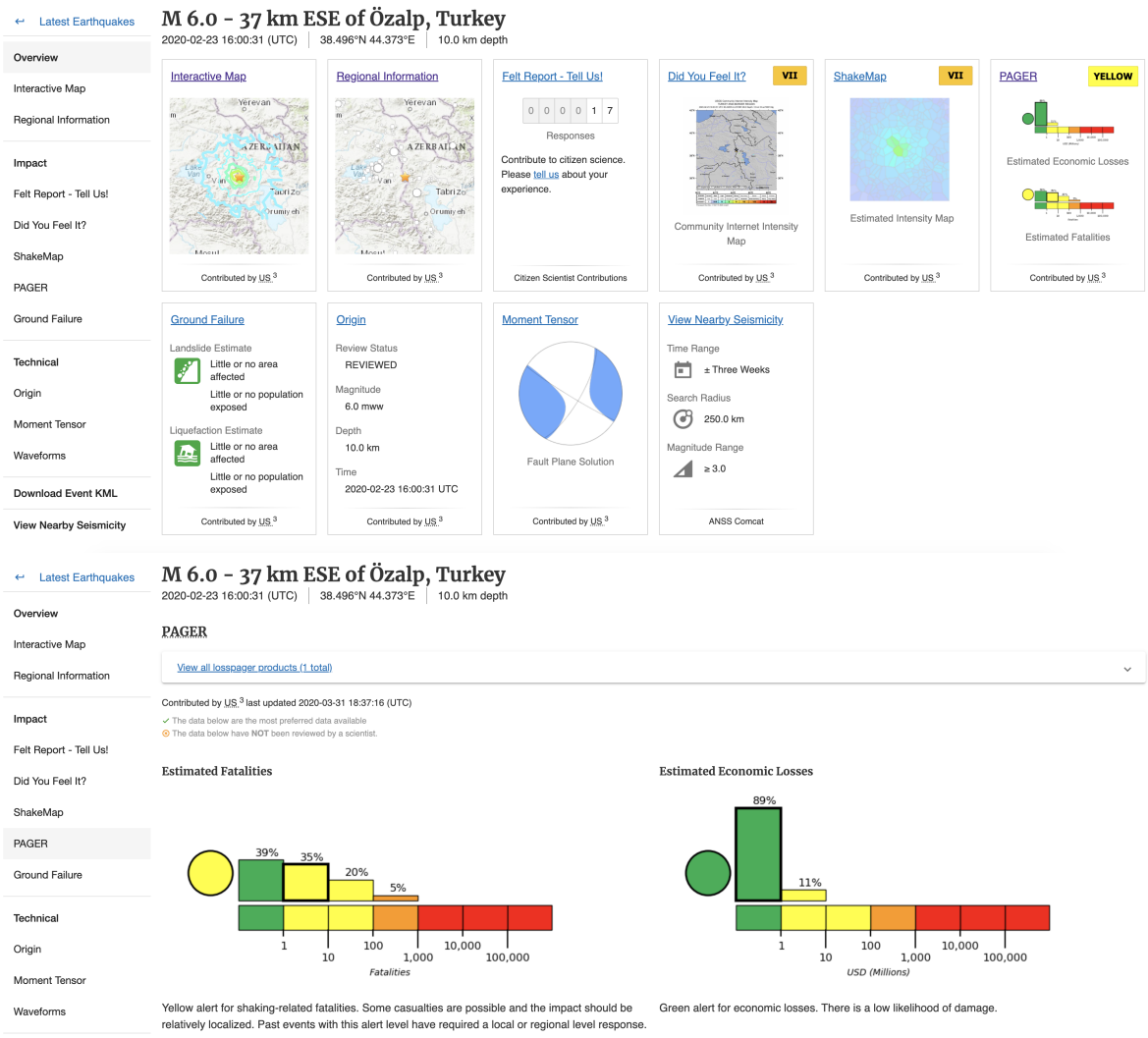


Figure 8.1: Example of earthquake information from USGS website (top) for the PAGER estimated economic losses and fatalities (bottom).

8.2.2 Category 2 of Collected Data

The second category of the collected data is the social media information about the earthquakes. After an earthquake is recorded from the USGS API, a scipt starts collecting related social media data from Twitter and related news articles from News API[3]. Tweets are collected over a period of three months using the keyword "earthquake" and the earthquake location. Tweets are also collected in the local language to capture local effects more precisely. News articles related to the earthquake are collected for a duration of a week. The

[3] `https://newsapi.org/`

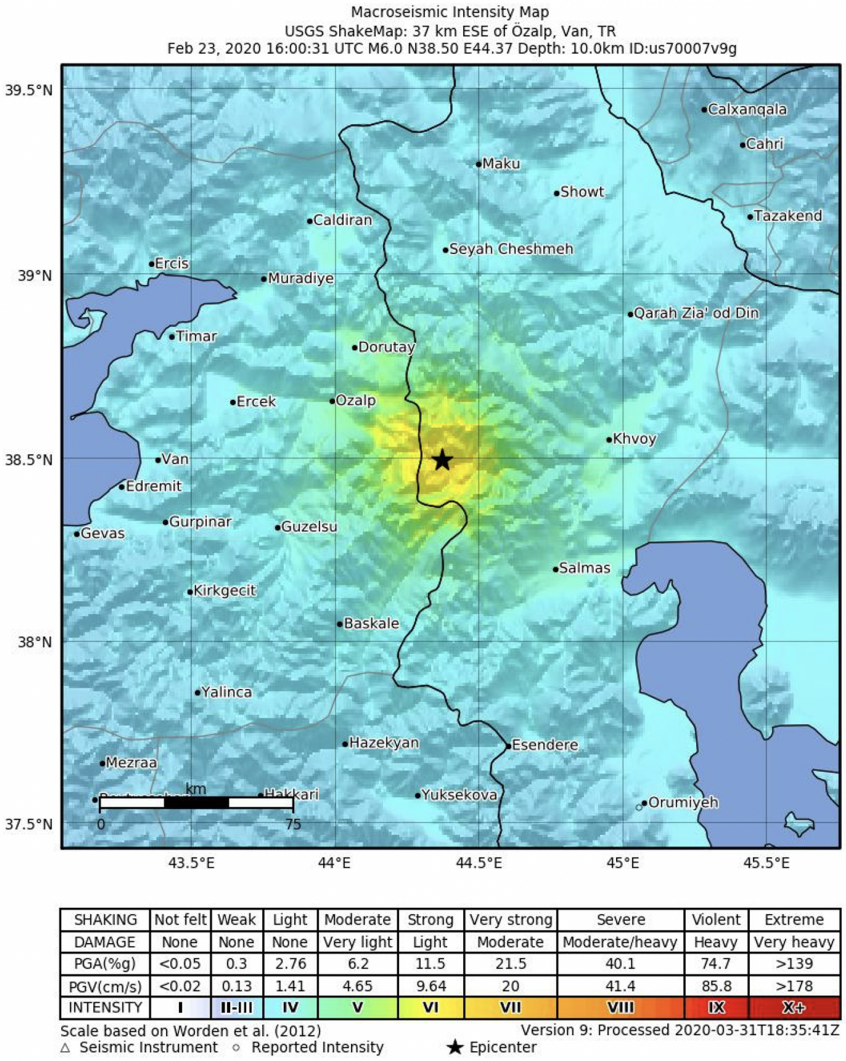


Figure 8.2: Example of the intensity map from USGS for an earthquake.

news articles data and social media data are respectively used in automatic report generation and in resilience analysis, as detailed in the next two sections. The workflow for automatic data collection after an earthquake is shown in Fig. 8.3.

8.3 Automatic Generation of Earthquake Briefings

The Structural Extreme Events Reconnaissance (StEER) Network[4] aims at building societal resilience by generating new knowledge on the performance of the built environment through

[4]https://www.steer.network/

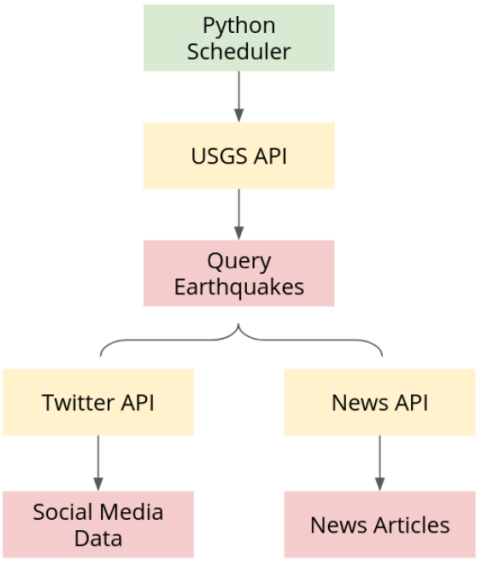


Figure 8.3: Automatic data collection workflow [114].

impactful post-disaster reconnaissance efforts disseminated to affected communities. StEER deepens the structural Natural Hazards Engineering (NHE) community's capacity for reliable post-event reconnaissance through: (1) promoting community-driven standards, best practices, and training for field reconnaissance, (2) coordinating early, efficient and impactful event responses, and (3) broadly engaging communities of research, practice, and policy to accelerate learning from disasters. One of the activities related to the coordination of early, efficient, and impactful event responses is to assemble a Virtual Assessment Structural Team (VAST) and to generate a reconnaissance report or briefing within few days after an extreme event such as an earthquake. The reconnaissance briefings are an essential component of each natural hazard field reconnaissance as they report all findings, observations, and conclusions from the event. These briefings can be in the form of reports from the detailed field assessments or preliminary reports based on virtual resources. In this section, the automated tool for generating earthquake briefings is described. It should be noted that the briefing generated here is not the final version, but rather an intermediate document that helps the domain experts create the final document in an accurate and efficient way. The provided tool not only decreases the time to generate a briefing, but also increases the abundance of information by facilitating systematic access to many identified resources that can be missed in conventional manual briefing preparation.

8.3.1 Overview

A typical earthquake briefing consists of standard sections of "Introduction," "Hazard Description," "Damage to Buildings," "Damage to Other Infrastructure," and "Resilience Aspects and Effects on Community". The "Introduction" and "Hazard Description" sections include standard contents with only a few items related to the specifics of the event (e.g., the date and time, the magnitude, the location, and the intensity measure). A script is developed that directly fills out the relevant information automatically. The remaining three sections ("Damage to Buildings," "Damage to Other Infrastructure," and "Resilience Aspects and Effects on Community") are generated using information collected from the new articles. In order to generate contents for these three sections, a classification task is performed to classify each sentence in the article into one of the four categories, which are "building," "infrastructure," "resilience," and "other". Sentences that are classified into the first three classes correspond to those to be summarized and added to the corresponding section. The fourth class corresponds to the sentences that do not fit into any of these three sections. The classification task is then followed by a document summarization task. In this study, an *extractive summarization technique* [4] is employed to condense and summarize all the sentences in each section.

8.3.2 Hazard Description

As mentioned above, information of new earthquakes queried from USGS is stored as files in the corresponding formats. For the `csv` files, a Python script is developed to parse the information and generate sentences that describe the basic information of the earthquakes. Information in other formats are pasted directly into the briefings. One exception is that for the PAGER estimations of fatalities and economic losses, textual explanations need to be generated by "reading" the probabilistic distributions values in Fig. 8.1. To generate this text, the numerical values are identified and located in the image using Computer Vision (CV) methods through the Python package `OpenCV` [86] and `pytesseract` [95]. Images are pre-processed (convert RGB/BGR[5] images to grayscale), and the elements (e.g., shapes, and strings) in the images are located (through bounding boxes, where the coordinates of four corners are identified, Figure 8.4) using `OpenCV`. Then, the strings in the string elements are recognized using Optical Character Recognition (OCR) method using `pytesseract`. Finally, the strings are converted to integers (if they are indeed numerical values rather than words), and parsed into the textual explanations.

8.3.3 Sentences Classification

Before the sentences obtained from the news articles are summarized within the separate briefing sections, the sentences need to be classified into the four categories described above.

[5]RGB and BGR images have different arrangements of the subpixels for Red, Green, and Blue for RGB and essentially in reverse for BGR.

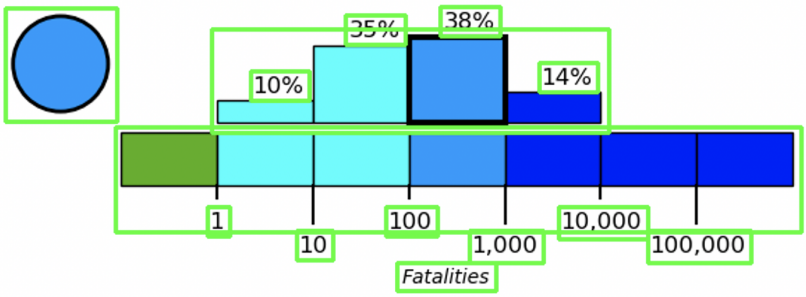


Figure 8.4: Example of identifying and locating numbers from USGS estimated distributions of fatalities and economic losses for an earthquake. The green boxes correspond to the elements identified and located in the figures. The OCR is conducted on these elements to identify if numerical values are present.

Currently, there is no available labeled dataset for this study. Therefore, a training dataset, containing around 200 sentences, is generated using available past StEER earthquake briefings and reports. The paragraphs are downloaded, and split into sentences, which are manually labeled based on the section that they belong to. Four classification methods are considered herein, from simple to complex ones, these are: (1) keywords matching[6], (2) Logistic Regression (LR), (3) Support Vector Machine (SVM), and (4) Convolutional Neural Network (CNN). The architecture from the study by Kim [52] is adopted for the CNN as a classifier of sentences, refer to Fig. 8.5. The words in the sentences are first embedded (i.e., convert each word in the sentence into a 1-D vector) by randomly-initialized word vectors of size k. Word embedding vectors are updated through back-propagation during training. Therefore, each sentence is converted into a 2-D matrix with size $n \times k$, where n is the number of words in the sentence. Three Convolutional filters (width of k) are applied on the matrix, with heights of 3, 4, and 5. After the Convolutional operation, Maxpooling is used to select the three maximum values from the results from the three Convolutional operations. Finally, a Fully Connected (FC) layer with softmax activation is applied, and the output is a 1-D vector of size 4, where each value corresponds to the probability that the sentence belongs to one of the 4 classes.

The above classification method is tested using an earthquake briefing for moment magnitude $M_w = 6.4$ Albania earthquake (2019), which caused significant damage and disruptions to the local community. For this event, around 130 sentences are collected from different

[6]For each of the three categories, a set of keywords are identified that are relevant to the category. For example, for the category of "buildings," the keywords are "building," "house," "apartment," "hotel," etc. In the keywords matching classifier, each sentence belongs to the category that matches the maximum number of keywords within the set of keywords for this category. If no keyword matches to any word of the sets of the three categories, the sentence would be classified as "others."

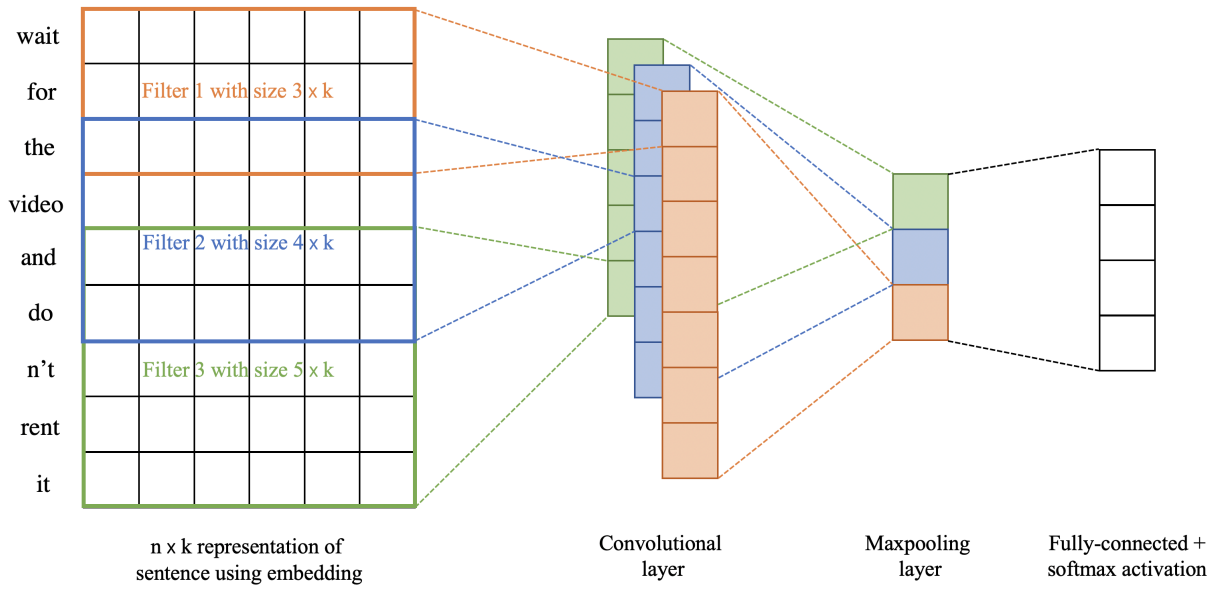


Figure 8.5: Convolutional classifier of sentences adopted from [52].

news websites. Table 8.1 summarizes the performance of the classification algorithms on the training data collected from the past briefings and the data from the news related to this test case of the 2019 Albania earthquake. The results show that keywords matching is quite limited as it is difficult to exhaustively list all of the possible keywords in each section. The other methods have relatively high performance. The *majority-voting*[7] is adopted to output the final result. The accuracy is considered sufficient, because as described above, the automatically-generated briefing is not regarded as the final one. Regardless, this "not so high" accuracy (especially for the test case) is attributed to the small size of the used training dataset and it is expected to increase with larger number of labeled data in the future.

8.3.4 Document Summarization

After the classification task is completed, the second step for the automatic generation of the earthquake briefing is to condense and synthesize sentences in each section. This is accomplished using techniques from *automatic document summarization*, which is the process of condensing a set of data computationally in order to create a summary that best represents the information of the original content. There are two general approaches to document summarization, which are *extractive summarization* and *abstractive summariza-*

[7]The *majority-voting* method is an ensemble learning method that predicts the label with the most votes from the set of classification algorithms.

Table 8.1: Sentence classification accuracy (%) for training and test data.

Algorithm	**Training**	**Test Case**
Keywords matching	61%	35%
LR	100%	67%
SVM	100%	67%
CNN	93%	75%
Majority-voting	96%	67%

tion [4]. Extractive summarization techniques extract the summary from the original data without modifying the sentences or phrases. In contrast, abstractive summarization may paraphrase the summary. In this study, the unsupervised extractive summarization technique, which aims at finding a minimal set of representative sentences of the original articles that effectively summarize these articles, is adopted, since the briefing generated is used as a first step to quickly provide useful and relevant information to researchers. Extractive summarization methods generally have more stable performance compared to the abstractive ones. The unsupervised extractive summarization method called *TextRank* [76] is used. It is a graph-based ranking model for text processing, and several use cases have demonstrated its success in benefiting automatic text summarization. The TextRank algorithm decides on a score of each sentence, based on text information drawn from the entire document, and selects the sentences that have the highest scores.

First, the TextRank model builds a *graph* that represents the document and the relationships between words. In the case of the extreme events (e.g., earthquakes) news summaries, each *vertex* in the graph represents a sentence in the text, and each *edge* in the graph represents the relationship between two vertices (i.e., two sentences). An *edge weight* is assigned to each edge and in this study, these relationships (represented by the edge weights) are expressed in terms of the "similarity" between sentences, where this similarity is measured as a function of their content overlap. Sentences that address similar concepts usually have high content overlap. The overlap of two sentences can be simply calculated as the number of common words in the two sentences. A *normalization factor* is also added to avoid favoring long sentences. Formally, given two sentences S_i and S_j, with the S_i sentence being represented by the set of N_i words $w_k^i, k = 1, 2, \ldots, N_i$ that appear in the sentence, i.e., $S_i = \{w_1^i, w_2^i, \ldots, w_{N_i}^i\}$, the similarity relationship between S_i and S_j is defined as follows:

$$\text{Similarity}(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \,\&\, w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)}, \tag{8.1}$$

where w_k is the common element in both sets S_i and S_j, therefore it has no superscript, and $|\bullet|$ indicates the cadinality (size) of the set $\bullet$.

Second, the TextRank model employs a graph-based ranking algorithm that takes into account the above edge weights when computing the score associated with each sentence in the graph. Let $G(V, E)$ be a *directed graph* with the set of vertices V and the set of edges

E. For a given vertex V_i, let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex V_i points to (successors). In the current application, the document graph is an *undirected graph*, in which the out-degree of a vertex (i.e., the number of connections that originate at a vertex and point outward to other vertices) is equal to the in-degree of the vertex (i.e., the number of connections that point inward at a vertex). The weighted score of a given sentence S_i, $WS(S_i)$, is defined as follows:

$$WS(S_i) = (1-d) + d \sum_{S_j \in In(S_i)} \frac{\text{Similarity}(S_j, S_i)}{\sum_{S_k \in Out(S_j)} \text{Similarity}(S_j, S_k)} WS(S_j), \tag{8.2}$$

where d is a damping factor that can take a value between 0.0 and 1.0, indicating the probability of jumping from a given vertex to another random vertex in the graph. After the ranking algorithm is run on the graph, the sentences are sorted in reversed order of their scores, ranking from the highest to the lowest, and the top ranked sentences are selected as the summary, refer to Fig. 8.6 for an example.

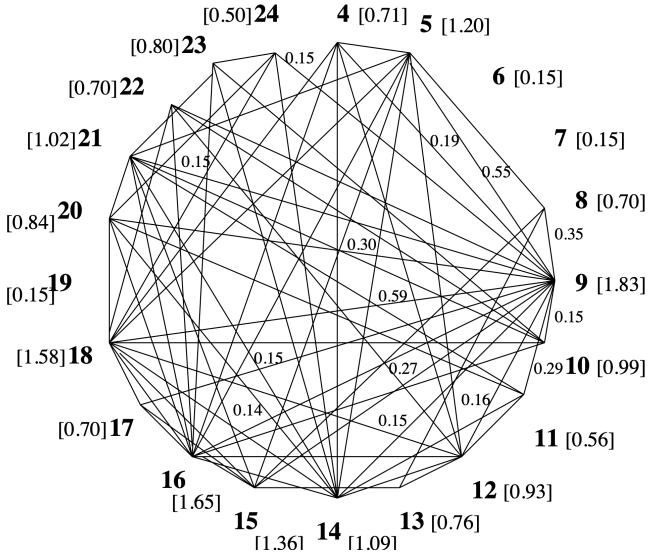


Figure 8.6: Example graph developed for extraction of sentences [76]: bold numbers are vertices representing sentences; numbers in square brackets are weighted scores WS; numbers inside the graph are computed similarities (i.e., edge weights).

8.3.5 Final Briefing Generation

The information obtained above (e.g., generated texts and any downloaded images) are gathered, and a draft briefing is generated into a Microsoft Office Word document file (`docx`) through the Python package `python-docx` [94]. A Portable Document Format (`pdf`) file

version is also created. Both versions are finally stored on the PEER server for experts to view, edit, and distribute.

8.4 Social Media for Resilience Analysis

In this section, social media information is used for resilience analysis. In particular, a *recovery time* is quantified in this study. In the context of extreme events, recovery time, conceptually illustrated in Fig. 1.1 as $t_1 - t_0$, is the time needed after the extreme event to restore the functionality of a structure, an infrastructure system (e.g. water supply, power grid, or transportation network), or a community, to a desired level that can operate or function the same, close to, or better than the condition before the extreme event [101]. The determination of the recovery time using information from the social media is based on the assumption that certain keywords related to recovery (e.g., school, office, transportation, or power outage) appear more frequently on the shared posts, tweets, etc., right after the occurrence of an earthquake and the frequency of these keywords reduces as time passes. Using this assumption, the time between the occurrence of the earthquake and when these frequencies reduce to pre-earthquake levels is used as a measure of the recovery time. The final recovery time is evaluated as the weighted average from each recover time (for different facilities) [48]. The steps to determine the recovery time are as follows:

1. Determine keywords for different facilities and assign user-defined weights to them (in this study: school: 20%, road: 20%, house: 20%, office: 20%, and collapse: 20%);

2. Determine the variation of the number of posts containing these keywords with time;

3. Determine the recovery time t_r for each factor, where $t_r = t_1 - t_0$, t_0 is the earthquake occurrence time, and t_1 is the time when the number of posts for the considered keyword falls below a pre-determined threshold (e.g., 15% of the maximum frequency) and becomes steady; and

4. Determine the final weighted recovery time from the keywords.

A case study is conducted to compute the recovery time by using Sina Weibo[8] posts collected for the moment magnitude $M_w = 6.6$ Ya'an, China earthquake (2013). Fig. 8.7 shows the frequency of keywords (school, road, house, office, and collapse) as a function of time (in hours) for this earthquake, where t_1 and t_0 are also specified on each plot. The final estimation of the recovery time is approximately 4 days (96 hours). In this case, the recovery time is not satisfactory, as it exceeds the 3 days (72 hours), which is the *golden relief time* after an event [85]. Therefore, this preliminary evaluation indicates the need to improve different aspects of the resiliency of the community to reduce this recovery time, as discussed in Chapter 1 in relation to Fig. 1.1.

[8]Sina Weibo, `https://weibo.com/us`, is a Chinese micro-blogging website launched in 2009. It is one of the biggest social media platforms in China.

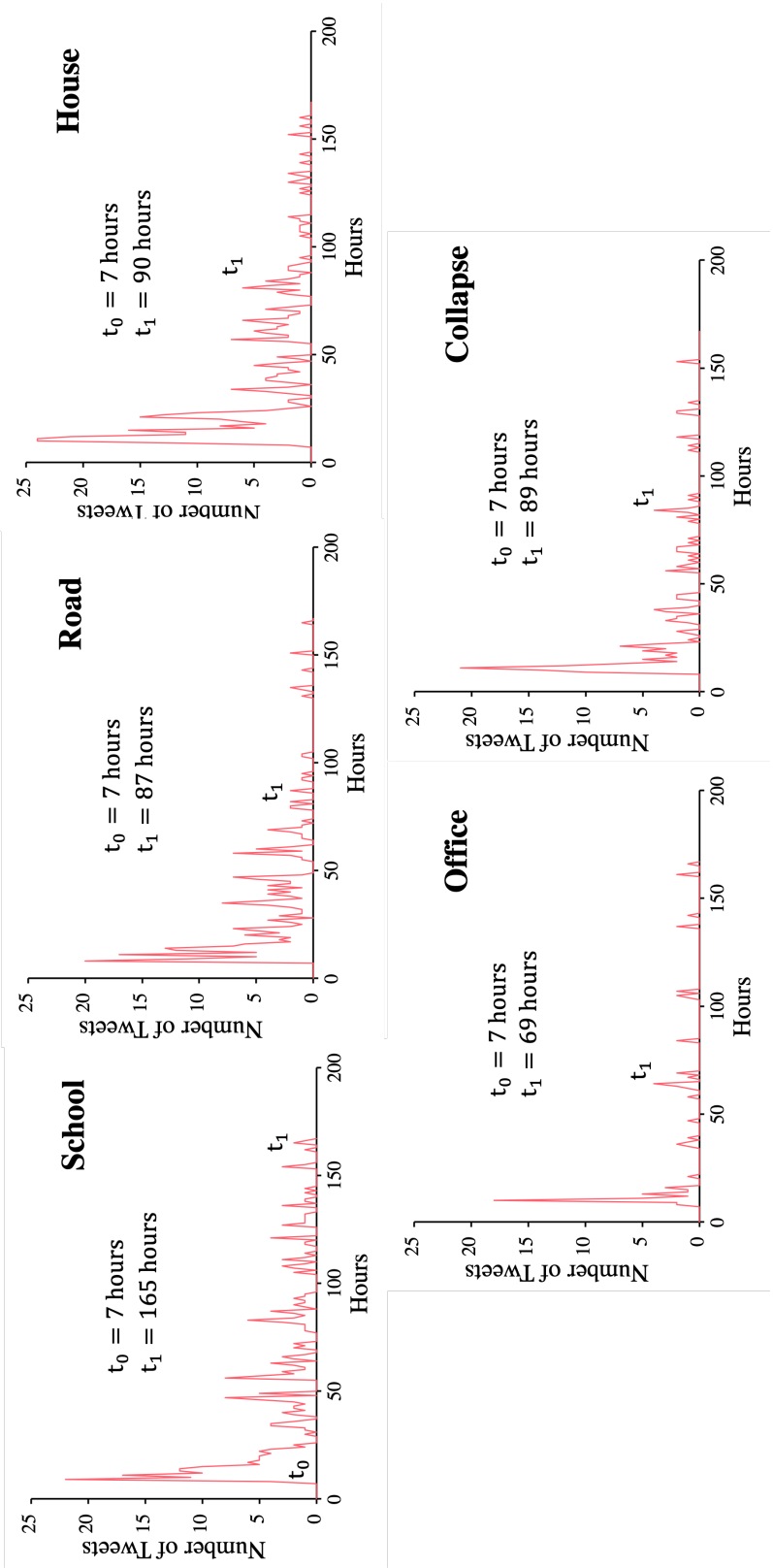


Figure 8.7: Variation of frequency of different keywords over time for an example earthquake.

Chapter 9

Summary, Conclusions and Future Extensions

9.1 Summary

In this dissertation, methods to improve the automatic infrastructure monitoring and reconnaissance process are proposed. Methods are introduced to improve the data collection process, where information that are highly relevant to the health states of structures as well as the whole infrastructure system and buildings can be collected. Algorithms are introduced to improve the data processing, where the "health" states of infrastructure systems and buildings can be obtained. Such results are helpful for the decision-making process to increase resiliency of the built environment and related communities. The most important technique to automate the above processes is the adoption of Artificial Intelligence (AI) tools, in particular, Machine Learning (ML) methods.

In this dissertation, the following three major topics are discussed:

- Structural Health Monitoring (SHM) using a Long Short-Term Memory (LSTM) Encoder-Decoder network for health diagnosis of a single structure (Chapters 4 and 5), and variants of Deep Learning (DL) Time Series (TS) models for the structural response prediction of a single structure (Chapter 6);
- Optimal Sensor Placement (OSP) using Directed Information (DI) for structural TS data collection (Chapter 7); and
- Regional post-earthquake reconnaissance and recovery time analysis of the built environment using automated tools and Natural Language Processing (NLP) techniques (Chapter 8).

Apart from the above three major topics, other topics are also discussed. For example, in Chapter 2, a brief theoretical background is provided for reference. In Chapter 3, the Finite Element Method (FEM) modeling of the used example structures is presented in detail. Such

theoretical foundation and the FEM models are used throughout this dissertation. The main contributions and conclusions for the three major topics, in addition to directions for future extensions, are presented in the next two sections.

9.2 Contributions & Conclusions

9.2.1 Structural Health Monitoring

A novel method of SHM, which utilizes the LSTM Encoder-Decoder network structure, is proposed, where the model shows promising results in the identification of structural health state from TS data obtained in two computer experiments. Two of the most important contributions are: (1) Model development for using the LSTM network as a SHM algorithm by making use of the TS obtained physically (through field or laboratory measurements) or computationally (through simulations using the FEM) from the structure; and (2) Model development by adopting the encoder-decoder architecture to improve the robustness and efficiency of the SHM algorithm. Several key conclusions are drawn from the study, which are listed below:

- The encoder part of the model maps the high dimensional multivariate TS data into a Latent Space Vector (LSV), i.e., compress the original TS into a compact vector representation. The LSTM network is chosen in order to capture the response of a structure for at least one cycle (the number of steps depends on the natural period and the sampling frequency of the adopted sensors or simulated data, but in general, it takes tens to hundreds of steps to represent the response for a full cycle), because of its capability to capture long-term dependencies in sequences of data in a structured manner. Under the belief that the compressed LSV represents the response and the health state of the structure, it is connected to an ordinary Neural Network (NN) to complete the health monitoring tasks. On the other hand, the decoder network works as a supplemental one to ensure the lossless compression of the encoder network. Both encoder and decoder networks are trained concurrently, and they learn an "approximate" identity mapping from the original TS (the input of the encoder) to the reconstructed TS (the output of the decoder). By this "approximate mapping," the noise from the input can be removed in the reconstruction process.

- The tasks proposed for the SHM are multi-level classification tasks, which are overall damage diagnosis (damage/no damage), damage localization (location of damage), and local damage pattern/severity identification (e.g., no damage, slight damage, and noticeable damage). The model should be trained on a set of Ground Motion (GM) responses, and the performance should be validated on another set of GM responses. This ensures robustness to unseen responses of the GM. The whole model is pretrained in the sense that it is trained on previous earthquakes. After the occurrence of an earthquake, the TS is transmitted to the model, and the trained model outputs the

predictions of the damage states from the identified tasks in real time. This ensures rapid and automated performance evaluation following major earthquakes, which facilitates the decision-making processes used for the post-earthquake community recovery.

- For the performance of the identified three tasks, damage diagnosis and localization achieve higher overall accuracy ($> 90\%$) compared to the local identification task accuracy ($\approx 80\%$). For the number of units in each LSTM cell (i.e., the dimension of the LSV's), there is a trade-off among model performance, complexity, and training time. When the model is more complex, the training time is generally longer (with some exceptions, e.g., deploying strategies for faster training possibly makes more complex models converge faster with shorter training time). There is no clear relationship between the model complexity and the model performance. Less complex models tend to under-fit the data, while more complex models tend to over-fit the data. There is an intermediate model complexity that does not over-fit or under-fit the data, and most probably produces the smallest validation loss, i.e., leading to the best classification performance. Therefore, longer training time does not necessarily indicate better performance. A few models with different number of units are trained for each application, and their performance is evaluated to select the best one. From the computer experiments, it is seen that cells with 50 to 100 units produced the best performance. In terms of the model structure, considering the trade-off between model training time, complexity, and accuracy, one-layer LSTM models adequately capture the response of multivariate TS inputs, and stacked (e.g., two-layer) LSTM networks are not generally needed for higher accuracy of the considered applications.

- The true labels have strong influence on the evaluation of the models. For the Concentrically Braced Steel (CBS) frame application in Section 5.2 (a FEM model experiment but yet similar to real practices), the real damage cases are continuous in nature, while in order to solve the identification problems as classification ones, strict discrete labels should be associated with the data points. Therefore, it is difficult to classify the data points near the boundaries of the classes into the "true" class they belong to. This observation significantly affects the classification performance of the different tasks investigated in this study.

- The performance of the LSTM model is found to be qualitatively consistent with the expected human expert evaluations. In the considered applications, loosening braces of the CBS frame application involving shaking table tests (Section 5.2) is a more significant and noticeable damage pattern than reducing the stiffness of the columns of the Reinforced Concrete (RC) planar frame application (Section 5.1). This is reflected in the overall comparison of the accuracy scores for these two applications. The localization task is generally easier with higher accuracy than the local damage severity/pattern identification task. Moreover, damage in the lower floors of buildings causes more observable change in the structural performance. This is more easily identified in the RC frame application. For the local damage severity/pattern identification, the

error (misclassification) is mainly one class away, i.e., it is easier for the trained model to misclassify two nearby classes, while it is harder for the model to make such misclassification between two classes that are more distinct (i.e., far from each other in an ordered set of classes representing the task at hand).

A novel method for a rapid structural response prediction under GM, which uses deep NN rather than traditional time history analyses, is proposed. Several models are explored with varying parameters. The models include: (1) vanilla LSTM (with one or two layers of cells), (2) LSTM with attention mechanism, (3) LSTM with convolutional filters, and (4) the Temporal Convolutional Network (TCN). Such models can accurately predict the responses and several key conclusions are drawn from the study, which are listed below:

- All the proposed models, except the LSTM model with attention mechanism, are able to successfully predict the responses of structures with reference to the time history analysis outputs. The LSTM model with the convolutional filters has the most satisfactory performance. The convolutional layer is helpful for pre-processing the random GM inputs. The original LSTM models (with one or two layers of cells) have slightly worse performance. However, the most important advantage of such original models is their simplicity, where the number of model parameters (e.g., # of units for the LSTM cells) is less than that in other models. Therefore, such simple models are still applicable irrespective of their sub-optimal performance because of their simple design. For the TCN model, which does not have the LSTM hidden state vector that passes through the time steps, lacks the "global view" of the TS data that the family of LSTM models possesses. Therefore, the performance of the TCN model is slightly lower than the LSTM-based models. The attention mechanism failed in this study because it is not effective in focusing on more important steps by averaging the response within nearby time steps.

- Similar to the LSTM Encoder-Decoder network, there is a trade-off among model training time, complexity, and accuracy of the proposed models. In this study, more complex models tend to have better performances (albeit this does not always hold), and longer training time is expected in general. For example, the two-layer LSTM network has better performance than the one-layer LSTM network. The TCN, due to its architecture that facilitates parallel computation among time steps, has the lowest training time. However, as described above, its performance is in general lower than that of LSTM models (e.g., LSTM with convolutional filters). For each model, several parameters are identified, e.g., the number of units for LSTM cells. Adding complexity (e.g., increasing the number of LSTM cell units) increases the training time, but it does not necessarily improve the performance. Therefore, in order to achieve more satisfactory performance, the selected models should be trained several times with varying parameters, and select the one that has the best performance. It should be noted that, to some extent, careful design of models can overcome such trade-off. For example, the complexity of adding convolutional layers is less than that of adding

a full LSTM layer. However, the performance of the former is better for the considered applications.

9.2.2 Optimal Sensor Placement

An algorithm is proposed to automate the process of selecting sensor layout plans to collect TS data that are highly indicative of the health states of structures. For this purpose, DI is used to quantify the causal relationship of TS data between two locations in the structure. This algorithm is based on the quantified causal relationship can select a subset of possible locations where sensors to be installed to monitor the health state of a structure. The algorithm shows promising results, both qualitatively and quantitatively, for the studied application. Several key conclusions are drawn from the study, which are listed below:

- The proposed algorithm is successful in selecting a qualitatively and quantitatively satisfactory sensor layout plan. The (estimated) DI is able to successfully quantify the causal relationship of sensor recordings in the TS of two locations. The results aligned well with the physical models (e.g., the geometric constraints of the structural elements). Therefore, such quantification is not only useful for the developed OSP algorithm, but is also helpful as a supplemental tool for physical model designs to plan and analyze the structural responses, e.g., instrument and analyze the load path of the structural earthquake resisting systems. The OSP algorithm is able to select a representative subset of possible locations using DI. The problem of OSP is transformed into a feature selection problem in ML, and uses related ML techniques in the implemented solution (e.g., using feed-forward feature selection process). Such transformation is not only helpful for the implementation, but is also beneficial for the interpretation of the results (e.g., observing the relationship between SHM performance and the number of sensors). The new features are selected such that the causal dependence between the new features and the existing ones is low, and the duplicate information among sensors is minimal. Therefore, the algorithm pushes for the wide distribution of sensors to cover distinct locations and directions, which would be helpful to comprehensively and cost-effectively monitor the health state of large structures.

- There are two directions to improve the performance of ML-based SHM models. The first direction is to collect more data, e.g., installing more sensors. This direction is particularly helpful when the dataset size (i.e., the number of data points and the number of features of these data points) is inadequate. In the initial validation of the OSP model in Section 7.3.3, as the number of features increases, the performance of the ML models also increases. However, collecting more data would result in more timely and costly effort for the collection, transmission, and processing. If more sensors are installed, the costs from installation, maintenance, etc., would be higher where these costs are proportional to the number of sensors. There is a classical trade-off between the size of data and the performance of the ML models. The second direction is to

improve the ML model. In this study, two ML models are adopted. The first uses a simple ML model, i.e., Support Vector Machine (SVM), and only the Cumulative Absolute Velocity (CAV) at selected sensor locations as features. The second uses the more complicated LSTM encoder-decoder model. It is observed that the second model has superior performance. Even by comparing the performance of the first model with a complete set of features (i.e., calculating the CAV from all the possible locations as features), and the second model with only the subset of features (i.e., training the LSTM model with only the TS from the selected sensor locations), the latter one had better performance. This illustrates the importance of careful design of the adopted ML model where sometimes designing an appropriate ML model is more important than collecting more data.

9.2.3 Regional Reconnaissance & Recovery Time

A method for automatic regional post-earthquake reconnaissance and recovery time analysis is proposed. First, public information (including information from United States Geological Survey (USGS) and social media) is gathered automatically after earthquake events. Second, the collected information is used to automatically generate earthquake briefings. Third, the collected information is used to estimate the recovery time, which is the time needed after the extreme event to restore the functionality of the built environment in the affected community. Several key conclusions are drawn from the study, which are listed below:

- In terms of the data collection, the provided tools are able to successfully extract information about the earthquakes. These tools not only decrease the time to generate a briefing, but also increase the abundance of information by facilitating systematic access to many identified resources that can be missed in conventional manual briefing preparation, e.g., search the keywords and browse the websites. Moreover, these tool can not only successfully extract official and objective information (such as numerical data from USGS and news articles and images from reporters), but also extract first-hand (albeit subjective) information from the general public (such as Tweets and Sina Weibo posts). Therefore, the collected information is considered to be more comprehensive than the information obtained through human preparation. This is similar to the idea of web scraping, where labor work is replaced by the automated scripts in the process of information collection. Web scraping of one web page requires very little time, allowing many web pages to be effectively scraped where the number of pages can be beyond the amount a human can extract information from in short time. Moreover, the collected information is more precise, because there is almost no mistake in the collected information through the developed scripts (this is particularly true for the information from USGS website, where many pieces of information (numerical data, text, and images) need to be collected).

- In terms of the data processing, for earthquake briefings, the sentences obtained from new articles need to be classified into three sections of briefings ("Damage to Build-

ings," "Damage to Other Infrastructure," and "Resilience Aspects and Effects on Community") using the trained classifiers using ML. The accuracy of these classifiers is shown to be sufficient, as the automatically generated reports are not regarded as final reports, but rather intermediate documents to help the domain experts to create the final documents in an accurate and efficient manner where the drafts are generated within short times following the events. Depending on the urgency of the decision-making, these drafts cab be first sent to personnel in charge of decisions as immediate references. Subsequently, these drafts can be edited by experts for creating the final versions to be archived.

9.3 Future Extensions

There are several future extensions that deserve further investigation. These extensions are listed below:

- The goal of the proposed SHM models (structural damage diagnosis and response prediction) is to expedite decision making. The SHM models could be linked to the Performance-Based Earthquake Engineering (PBEE) quantitative analysis and design framework or more generally the Performance-Based Engineering (PBE) and its broader extension to Resilient Design for Extreme Events (RDEE). In the PBEE, the response of a structure is quantified as the Engineering Demand Parameters (EDP) from the structural analysis based on the estimated Intensity Measures (IM) from the hazard analysis, and the EDP are linked to the Damage Measures (DM) by the damage analysis, which are finally converted to the Decision Variables (DV), e.g., casualties, economic loss, and downtime, from the loss analysis [122]. Therefore, it is possible to link the DM identified in the damage diagnosis model, as well as the Engineering Demand Parameters obtained from the response prediction model, to the DV, refer to Fig. 2.3. Currently, the PBEE methodology has been standardized [1], and available software programs support this methodology. These programs rely on statistical sampling [103][93] and are available for further expansion and applications.

- In the OSP algorithm proposed in Chapter 7, the number of sensors to be installed and all possible (not optimal) locations for their installation is pre-selected by experts. Subsequently, the OSP algorithm is mostly automated to determine the optimal sensor locations. As a future extension, the OSP algorithm can be fully automated, where further study to design an algorithm that selects the number of sensors will be needed. The algorithm should be based on the structural size (e.g., the number of floors and the number of bays in two horizontal directions), the structural complexity (e.g., whether the building plan is rectangular or has vertical or horizontal irregularities), and the available resources (e.g., price and availability of sensors) for instrumentation, communication, data storage, and maintenance.

- In the resilience analysis in Chapter 8, only a recovery time is determined, which is quantified as the average recovery time (based on the keyword occurrence frequency) of several keywords representing different components/facilities of the built environment, e.g., schools, hospitals. There are two main problems: (1) the quantification of recovery time is too simplified; and (2) it only quantifies the recovery time, rather than the quality of the infrastructure system $Q(t)$ (as shown in Fig. 1.1). As seen in Eq. 1.1, obtaining $Q(t)$ is more important to quantify the resiliency of the community including the infrastructure systems. Moreover, the recovery time is automatically obtained based on $Q(t)$. Therefore, quantification of $Q(t)$ by establishing a more comprehensive quantitative model is important. Based on that, processing a wider range of obtained numerical or text information (through NLP methods) is required. A comprehensive city-scale infrastructure information (e.g., the real time data of electricity consumption and water usage from the electric grid and the water distribution system, respectively) may also be helpful in that regard.

A closure note about the dissertation is that it presents a few tools for rapid and more efficient monitoring and reconnaissance of infrastructure systems and buildings using AI technology. However, such tools are not a replacement of the work of structural engineers. On the one side, human expertise is still required for making decisions, e.g., the determination of reparation plans of structures based on the SHM diagnosis output, and the determination of the possible locations of sensors in the OSP algorithm. On the other side, in comparison to traditional physics-based SHM, the proposed AI models incorporate statistical analyses. As the complexity of the monitored infrastructure systems and buildings increases, the required amount of data for such data-driven SHM would also increase, which increases the cost related to the data collection, storage, and processing. Indeed, that is the inherent problem of statistical learning models. Therefore, it is natural to think of a balanced point between physics-based models and statistical-based or data-driven models, i.e., a mixture of these two classes of models, namely, a digital twin. The creation of physics-based models still requires expertise of engineering professionals. In the current era where AI is gaining popularity, structural engineers are expected to play an essential role in the field of data-driven monitoring and reconnaissance and its future development and growth of applications.

Bibliography

[1] Federal Emergency Management Agency (FEMA). *Seismic Performance Assessment of Buildings*. 2018.

[2] *2021 Report Card for America's Infrastructure*. `https://infrastructurereportcard.org/`.

[3] Chengbo Ai et al. "A Nonballasted Rail Track Slab Crack Identification Method Using a Level-Set-Based Active Contour Model". In: *Computer-Aided Civil and Infrastructure Engineering* 33.7 (2018), pp. 571–584.

[4] Mehdi Allahyari et al. *Text Summarization Techniques: A Brief Survey*. 2017. arXiv: `1707.02268`.

[5] *ANSYS*. `https://www.ansys.com/`.

[6] Yalin Arici and Khalid M. Mosalam. "Statistical significance of modal parameters of bridge systems identified from strong motion data". In: *Earthquake Engineering & Structural Dynamics* 34.10 (2005), pp. 1323–1341.

[7] Yalin Arici and Khalid M. Mosalam. "System identification of instrumented bridge systems". In: *Earthquake Engineering & Structural Dynamics* 32.7 (2003), pp. 999–1020.

[8] M. Azarbayejani et al. "A probabilistic approach for optimal sensor allocation in structural health monitoring". In: *Smart Materials and Structures* 17.5 (2008), p. 055019.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. arXiv: `1409.0473`.

[10] Seongdeok Bang et al. "Encoder–decoder network for pixel-level road crack detection in black-box images". In: *Computer-Aided Civil and Infrastructure Engineering* 34.8 (2019), pp. 713–727.

[11] Tanvi Bhandarkar et al. "Earthquake trend prediction using long short-term memory RNN". In: *International Journal of Electrical and Computer Engineering (IJECE)* 9 (2019), p. 1304.

[12] Michel Bruneau et al. "A Framework to Quantitatively Assess and Enhance the Seismic Resilience of Communities". In: *Earthquake Spectra* 19 (2003).

[13] T. G. Carne and C. R. Dohrmann. "A modal test design strategy for model correlation". In: (Dec. 1994).

[14] P. Cawley and R. Adams. "The location of defects in structures from measurements of natural frequencies". In: *The Journal of Strain Analysis for Engineering Design* 14.2 (1979), pp. 49–57.

[15] Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk. "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks". In: *Computer-Aided Civil and Infrastructure Engineering* 32.5 (2017), pp. 361–378.

[16] Young-Jin Cha et al. "Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types". In: *Computer-Aided Civil and Infrastructure Engineering* 33.9 (2018), pp. 731–747.

[17] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[18] Zhengping Che et al. "Recurrent Neural Networks for Multivariate Time Series with Missing Values". In: *Scientific Reports* 8.6085 (2018).

[19] Yueshi Chen, Wei Wang, and Yiyi Chen. "Full-scale shake table tests of the tension-only concentrically braced steel beam-through frame". In: *Journal of Constructional Steel Research* 148 (2018), pp. 611–626.

[20] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: `1406.1078`.

[21] Kyunghyun Cho et al. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. 2014. arXiv: `1409.1259`.

[22] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[23] Anil K. Chopra. *Dynamics of Structures, Theory and Applications to Earthquake Engineering*. 2017.

[24] American Society of Civil Engineers. *ASCE/SEI 7-16, Minimum Design Loads and Associated Criteria for Buildings and Other Structures*. 2017.

[25] Richard G. Cobb and Brad S. Liebst. "Sensor placement and structural damage identification from minimal sensor information". In: *AIAA journal* 35.2 (1997), pp. 369–374.

[26] Applied Technology Council. *Field Manual: Postearthquake Safety Evaluation of Buildings*. 2005.

[27] H. Cramer, H. Cramér, and Karreman Mathematics Research Collection. *Mathematical Methods of Statistics*. Goldstine Printed Materials. Princeton University Press, 1946.

[28] Shumin Deng et al. "Knowledge-driven stock trend prediction and explanation via temporal convolutional network". In: *Companion Proceedings of The 2019 World Wide Web Conference*. 2019, pp. 678–685.

[29] DIANA FEA. *User's Manual – Release 10.3*. 2019.

[30] R. V. Field Jr and M. Grigoriu. "Optimal design of sensor networks for vehicle detection, classification, and monitoring". In: *Probabilistic engineering mechanics* 21.4 (2006), pp. 305–316.

[31] Eric B. Flynn and Michael D. Todd. "A Bayesian approach to optimal sensor placement for structural health monitoring with application to active sensing". In: *Mechanical Systems and Signal Processing* 24.4 (2010), pp. 891–903.

[32] Yuqing Gao, Boyuan Kong, and Khalid M. Mosalam. "Deep leaf-bootstrapping generative adversarial network for structural image data augmentation". In: *Computer-Aided Civil and Infrastructure Engineering* 34.9 (2019), pp. 755–773.

[33] Yuqing Gao and Khalid M. Mosalam. "Deep Transfer Learning for Image-Based Structural Damage Recognition". In: *Computer-Aided Civil and Infrastructure Engineering* 33.9 (2018), pp. 748–768.

[34] Yuqing Gao and Khalid M. Mosalam. "PEER Hub ImageNet (Φ-Net): A Large-Scale Multi-Attribute Benchmark Dataset of Structural Images". In: *Pacific Earthquake Engineering Research (PPER) Center Report* 7 (Nov. 2019).

[35] Yuqing Gao et al. "Auto-Regressive Integrated Moving-Average Machine Learning for Damage Identification of Steel Frames". In: *Applied Sciences* 11.13 (2021).

[36] D. Asir Antony Gnana, S. Appavu Alias Balamurugan, and E. Jebamalar Leavline. "Literature review on feature selection methods for high-dimensional data". In: *International Journal of Computer Applications* 136.1 (2016), pp. 9–17.

[37] Alex Graves, Abdelrahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6645–6649.

[38] Michael Grieves. "Virtually Intelligent Product Systems: Digital and Physical Twins". In: *Complex Systems Engineering: Theory and Practice*. American Institute of Aeronautics and Astronautics, July 2019, pp. 175–200. ISBN: 978-1624105647.

[39] Aiping Guo et al. "Data mining algorithms for bridge health monitoring: Kohonen clustering and LSTM prediction approaches". In: *The Journal of Supercomputing* (2019).

[40] A. Hać and L. Liu. "Sensor And Actuator Location In Motion Control Of Flexible Structures". In: *Journal of Sound and Vibration* 167.2 (1993), pp. 239–261.

[41] G. Heo, M. L. Wang, and D. Satpathi. "Optimal transducer placement for health monitoring of long span bridge". In: *Soil Dynamics and Earthquake Engineering* 16.7 (1997), pp. 495–502.

[42] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (1997), pp. 1735–80.

[43] Ministry of Housing and China Urban-Rural Development of China Beijing. *Code for Seismic Design of Buildings (GB50011-2010, in Chinese)*. 2010.

[44] Chuanshuang Hu and Muhammad T. Afzal. "A statistical algorithm for comparing mode shapes of vibration testing before and after damage in timbers". In: *Journal of Wood Science* 52 (2006), pp. 348–352.

[45] *Image Question Answering using CNN with Dynamic Parameter Prediction*. `https://wiki.math.uwaterloo.ca/statwiki/index.php?title=stat441w18/Image_Question_Answering_using_CNN_with_Dynamic_Parameter_Prediction`.

[46] Jiantao Jiao et al. "Universal estimation of directed information". In: *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6220–6242.

[47] Joe Yue-Hei Ng et al. "Beyond short snippets: Deep networks for video classification". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 4694–4702.

[48] Elizabeth Jordan and Amy Javernick-Will. "Indicators of Community Recovery: Content Analysis and Delphi Approach". In: *Natural Hazards Review* 14 (2013), pp. 21–28.

[49] Daniel C. Kammer. "Sensor placement for on-orbit modal identification and correlation of large space structures". In: *Journal of Guidance, Control, and Dynamics* 14.2 (1991), pp. 251–259.

[50] Daniel C. Kammer and Michael L. Tinkerb. "Optimal placement of triaxial accelerometers for modal vibration tests". In: *Mechanical Systems and Signal Processing* 18.1 (2004), pp. 29–41.

[51] Peyman Kaviani, Farzin Zareian, and Ertugrul Taciroglu. "Seismic behavior of reinforced concrete bridges with skew-angled seat-type abutments". In: *Engineering Structures* 45 (2012), pp. 137–150.

[52] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 1746–1751.

[53] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: `1412.6980`.

[54] Xiangxiong Kong and Jian Li. "Non-contact fatigue crack detection in civil infrastructure through image overlapping and crack breathing sensing". In: *Automation in Construction* 99 (2019), pp. 125–139.

[55] Kesavan Krishnan Nair and Anne S. Kiremidjian. "Time Series Based Structural Damage Detection Algorithm Using Gaussian Mixtures Modeling". In: *Journal of Dynamic Systems, Measurement, and Control* 129.3 (2006), pp. 285–293.

[56] Solomon Kullback and Richard A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[57] Vipin Kumar and Sonajharia Minz. "Feature selection: a literature review". In: *SmartCR* 4.3 (2014), pp. 211–229.

[58] Huseyin Kuyuk and Ohno Susumu. "Real-Time Classification of Earthquake using Deep Learning". In: *Procedia Computer Science* 140 (2018), pp. 298–305.

[59] Heung-Fai Lam, Ka-Veng Yuen, and James L. Beck. "Structural Health Monitoring via Measured Ritz Vectors Utilizing Artificial Neural Networks". In: *Computer-Aided Civil and Infrastructure Engineering* 21.4 (2006), pp. 232–241.

[60] Colin Lea et al. "Temporal convolutional networks: A unified approach to action segmentation". In: *European Conference on Computer Vision*. Springer. 2016, pp. 47–54.

[61] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521 (2015), pp. 436–44.

[62] Kanghyeok Lee et al. "A Novelty Detection Approach for Tendons of Prestressed Concrete Bridges Based on a Convolutional Autoencoder and Acceleration Data". In: *Sensors* 19.7 (2019), p. 1633.

[63] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. *A Hierarchical Neural Autoencoder for Paragraphs and Documents*. 2015. arXiv: `1506.01057`.

[64] Juelong Li et al. "Optimal sensor placement for long-span cable-stayed bridge using a novel particle swarm optimization algorithm". In: *Journal of Civil Structural Health Monitoring* 5.5 (2015), pp. 677–685.

[65] Xiao Liang and Khalid M. Mosalam. "Ground motion selection and modification evaluation for highway bridges subjected to Bi-directional horizontal excitation". In: *Soil Dynamics and Earthquake Engineering* 130 (2020), p. 105994.

[66] Xiao Liang, Khalid M. Mosalam, and Sifat Muin. "Simulation-Based Data-Driven Damage Detection for Highway Bridge Systems". In: *11th National Conference on Earthquake Engineering*. 2018.

[67] Tung Yen Lin and Sidney D. Stotesbury. *Structural Concepts and Systems for Architects and Engineers*. Van Nostrand Reinhold Company, 1988.

[68] Yipeng Liu et al. "Short-term traffic flow prediction with Conv-LSTM". In: *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE. 2017, pp. 1–6.

[69] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9 (2008), pp. 2579–2605.

[70] Spyros Makridakis, Evangelos Spiliotis, and Vassilis Assimakopoulos. "The M4 Competition: Results, findings, conclusion and way forward". In: *International Journal of Forecasting* (2018).

[71] Pankaj Malhotra et al. *LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection*. 2016. arXiv: `1607.00148`.

[72] Sujith Mangalathu and Henry V. Burton. "Deep learning-based classification of earthquake-impacted buildings using textual damage descriptions". In: *International Journal of Disaster Risk Reduction* 36 (2019), p. 101111.

[73] James L. Massey. "Causality, feedback and directed information". In: *International Symposium on Information Theory and Its Applications (ISITA-90)*. Citeseer. 1990, pp. 303–305.

[74] *MATLAB*. `https://www.mathworks.com/products/matlab.html`.

[75] Liu Mei, Akira Mita, and Jin Zhou. "An improved substructural damage detection approach of shear structure based on ARMAX model residual". In: *Structural Control and Health Monitoring* 23.2 (2016), pp. 218–236.

[76] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Text". In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 404–411.

[77] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[78] Jack Moehle and Gregory Deierlein. "A framework methodology for performance-based earthquake engineering". In: *13th World Conference on Earthquake Engineering* 1 (2004).

[79] Khalid M. Mosalam and Yalin Arici. "Health monitoring of a bridge system using strong motion data". In: *Smart Structures and Systems* 5 (July 2009).

[80] Sifat Muin and Khalid M. Mosalam. "Cumulative Absolute Velocity as a Local Damage Indicator of Instrumented Structures". In: *Earthquake Spectra* 33 (2017), pp. 641–664.

[81] Sifat Muin and Khalid M. Mosalam. "Human-machine Collaboration Framework for Structural Health Monitoring and Resiliency". In: *Engineering Structures* 235 (2021), p. 112084.

[82] Sifat Muin and Khalid M. Mosalam. "Localized Damage Detection of CSMIP Instrumented Buildings using Cumulative Absolute Velocity: A Machine Learning Approach". In: *SMIP18 Seminar on Utilization of Strong-Motion Data* (Sacramento, California). 2018.

[83] *NAE Grand Challenges for Engineering: 14 Grand Challenges for Engineering in the 21st Century*. `http://www.engineeringchallenges.org/`.

[84] Kesavan Krishnan Nair, Anne S. Kiremidjian, and Kincho H. Law. "Time series-based damage detection and localization algorithm with application to the ASCE benchmark structure". In: *Journal of Sound and Vibration* 291.1 (2006), pp. 349–368.

[85] Sergio F. Ochoa and Rodrigo Santos. "Human-centric Wireless Sensor Networks to Improve Information Availability During Urban Search and Rescue Activities". In: *Information Fusion* 22 (2015), pp. 71–84.

[86] *OpenCV*. https://opencv.org/.

[87] *OpenSeesPy*. https://openseespydoc.readthedocs.io/en/latest/index.html.

[88] Hong Pan et al. "Time-Frequency-Based Data-Driven Structural Diagnosis and Damage Detection for Cable-Stayed Bridges". In: *Journal of Bridge Engineering* 23.6 (2018), p. 04018033.

[89] Costas Papadimitriou, James L. Beck, and Siu-Kui Au. "Entropy-Based Optimal Sensor Location for Structural Model Updating". In: *Journal of Vibration and Control* 6.5 (2000), pp. 781–800.

[90] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[91] *PEER Ground Motion Database*. https://ngawest2.berkeley.edu/.

[92] J. E. T. Penny, M. I. Friswell, and S. D. Garvey. "Automatic choice of measurement locations for dynamic testing". In: *AIAA Journal* 32.2 (1994), pp. 407–414.

[93] *Performance Based Engineering Application (PBE)*. https://simcenter.designsafe-ci.org/research-tools/pbe-application/.

[94] *python-docx*. https://python-docx.readthedocs.io/en/latest/.

[95] *Python-tesseract*. https://pypi.org/project/pytesseract/.

[96] Arvind Rao et al. "Directed-Information Based Feature-Selection for Tissue-Specific Sequences". In: *2007 IEEE/SP 14th Workshop on Statistical Signal Processing*. IEEE. 2007, pp. 210–214.

[97] J. W. Reed and R. P. Kassawara. "A Criterion for Determining Exceedance of the Operating Basis Earthquake". In: *Nuclear Engineering and Design* 123.2-3 (1990), pp. 387–396.

[98] *RNN, Talking about Gated Recurrent Unit*. https://technopremium.com/blog/rnn-talking-about-gated-recurrent-unit/.

[99] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. eprint: 1609.04747.

[100] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2020.

[101] H. Russello. "Framework for analytical quantification of disaster resilience". In: *Engineering structures* 32 (2010), pp. 3639–3649.

[102] Anders Rytter. "Vibrational Based Inspection of Civil Engineering Structures". PhD thesis. Denmark, 1993.

[103] *Seismic Performance Prediction Program (SP3)*. https://www.hbrisk.com/.

[104] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[105] B. H. Shekar and Guesh Dagnew. "Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data". In: *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*. 2019, pp. 1–8.

[106] Weixing Shi, Jiazeng Shan, and Xilin Lu. "Modal identification of Shanghai World Financial Center both from free and ambient vibration response". In: *Engineering Structures* 36 (2012), pp. 14–26.

[107] Xingjian Shi et al. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Advances in Neural Information Processing Systems 28*. 2015, pp. 802–810.

[108] Shun-Yao Shih, Fan-Keng Sun, and Hung-Yi Lee. "Temporal Pattern Attention for Multivariate Time Series Forecasting". In: *Machine Learning* 108 (2019), pp. 1421–1441.

[109] D. A. Sofge. "Structural health monitoring using neural network based vibrational system identification". In: *Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference*. 1994, pp. 91–94.

[110] Kevin Stowe et al. "Developing and Evaluating Annotation Procedures for Twitter Data during Hazard Events". In: *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 133–143.

[111] Hao Sun and Oral Büyüköztürk. "Optimal sensor placement in structural health monitoring using discrete optimization". In: *Smart Materials and Structures* 24.12 (2015), p. 125034.

[112] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27*. 2014, pp. 3104–3112.

[113] Yujin Tang et al. "Sequence-to-Sequence Model with Attention for Time Series Classification". In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. 2016, pp. 503–510.

[114] Alicia Y. Tsai et al. *Text Analytics for Resilience-Enabled Extreme Events Reconnaissance*. 2020. arXiv: 2011.13087.

[115] *Twitter*. https://twitter.com/.

[116] Firdaus Udwadia. "Methodology for Optimum Sensor Locations for Parameter Identification in Dynamic Systems". In: *Journal of Engineering Mechanics* 120 (Feb. 1994).

[117] *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[118] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762.

[119] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. "The context-tree weighting method: Basic properties". In: *IEEE transactions on information theory* 41.3 (1995), pp. 653–664.

[120] K. Worden and A. P. Burrows. "Optimal sensor placement for fault detection". In: *Engineering Structures* 23.8 (2001), pp. 885–901.

[121] Yongjia Xu et al. "Real-time regional seismic damage assessment framework based on long short-term memory neural network". In: *Computer-Aided Civil and Infrastructure Engineering* 36.4 (2021), pp. 504–521.

[122] T. Y. Yang et al. "Seismic Performance Evaluation of Facilities: Methodology and Implementation". In: *Journal of Structural Engineering* 135.10 (2009), pp. 1146–1154.

[123] Leehter Yao, William A. Sethares, and Daniel C. Kammer. "Sensor placement for on-orbit modal identification via a genetic algorithm". In: *AIAA Journal* 31.10 (1993), pp. 1922–1928.

[124] Ting-Hua Yi, Hong-Nan Li, and Ming Gu. "Optimal sensor placement for structural health monitoring based on multiple optimization strategies". In: *The Structural Design of Tall and Special Buildings* 20.7 (2011), pp. 881–900.

[125] Ting-Hua Yi, Hong-Nan Li, and Xu-Dong Zhang. "A modified monkey algorithm for optimal sensor placement in structural health monitoring". In: *Smart Materials and Structures* 21.10 (Aug. 2012), p. 105033.

[126] Ka-Veng Yuen et al. "Optimal Sensor Placement Methodology for Identification with Unmeasured Excitation". In: *Journal of Dynamic Systems Measurement and Control-transactions* 123 (Dec. 2001).

[127] Kiran Zahra, Muhammad Imran, and Frank O. Ostermann. "Automatic identification of eyewitness messages on twitter during disasters". In: *Information processing & management* 57.1 (2020), p. 102107.

[128] A. Żak, M. Krawczuk, and W. Ostachowicz. "Vibration of a Laminated Composite Plate with Closing Delamination". In: *Journal of Intelligent Material Systems and Structures* 12.8 (2001), pp. 545–551.

[129] Jun Zhang et al. "An Improved Long Short-Term Memory Model for Dam Displacement Prediction". In: *Mathematical Problems in Engineering* 2019 (2019), pp. 1–14.

[130] Ruiyang Zhang et al. "Deep long short-term memory networks for nonlinear structural seismic response prediction". In: *Computers & Structures* 220 (2019), pp. 55–68.

[131] Yixin Zheng et al. "Understanding Environmental Effect on Building Vibration for Structural Health Monitoring Using Event Detection and Causal Analysis". In: 2013.

[132] Guang-Dong Zhou, Ting-Hua Yi, and Hong-Nan Li. "Sensor Placement Optimization in Structural Health Monitoring Using Cluster-In-Cluster Firefly Algorithm". In: *Advances in Structural Engineering* 17.8 (2014), pp. 1103–1115.

Appendix A

Details of the Planar Reinforced Concrete Frame Model

In this appendix, details of the Reinforced Concrete (RC) frame model using Finite Element Method (FEM) in Section 3.1 are described. The model is developed using concrete (the main element) and steel reinforcement (which includes the longitudinal reinforcing bars and the transverse stirrups). The used material properties of the concrete and steel in the model are listed in Tables A.1 and A.2, respectively. Note that for steel material, the plastic stress-strain relationship is defined by a bilinear model, and Table A.2 shows the two controlling pairs of strain & stress (unit: N/mm^2) points. The compressive strength defined in Table A.2 is taken as the peak value defining a parabolic stress-strain relationship for the nonlinear behavior of concrete in compression.

Table A.1: Material properties of concrete used in the RC frame model.

Property	**Value**
Young's modulus	34,300 N/mm^2
Poisson's ratio	0.2
Mass density	2,400 kg/m^3
Smeared crack model	Rotating
Tensile strength	3.13 N/mm^2
Compressive strength	43.5 N/mm^2

Since the model for the RC framed structure is planar, the thickness of the elements in the out-plane direction is specified. Therefore, the reduction of stiffness (as conducted in Chapter 5) is modeled as reduction of the thickness of the targeted elements (e.g., for 10% reduction of the stiffness of the first story columns, the thicknesses of these targeted columns are reduced by 10%). Moreover, since the elements are square or rectangular in cross-section, the four-node quadrilateral elements are used as the type of the used finite elements.

Table A.2: Material properties of steel used in the RC frame model.

Property	**Value**
Young's modulus	200,000 N/mm^2
Poisson's ratio	0.3
Mass density	8,050 kg/m^3
Plastic strain-yield stress	{(0, 460), (0.1, 660)}

The model is supported at the base, i.e., at the bottom of the columns of the first story. The 3-D translations are constrained at all nodes. Therefore, the 3-D rotations are also implicitly constrained. The uniform excitation is applied at the base. For simplicity, only the excitation in the X direction (see Fig. 3.1 for the coordinate system specifications) is applied. The applied excitation (101 Ground Motion (GM) runs) corresponds to real earthquake recorded GM signals, which are listed in Appendix B.

Nonlinear transient analyses are conducted with Newton-Raphson method used as the iterative solver, with maximum number of iterations set to 20. The convergence norm is set for force and displacement, with limiting values of 0.01 for both. The time integrator is selected as Newmark β method with the default values $\gamma = 0.5$ and $\beta = 0.25$ as specified in the used commercial software DIANA manual [29].

Appendix B

Ground Motion Inputs Used in the Finite Element Method Simulations

In this appendix, the Ground Motion (GM) applied to the two numerical models in Chapter 3, i.e., the Reinforced Concrete (RC) frame and the Concentrically Braced Steel (CBS) frame, are listed in Tables B.1 and B.2, respectively. These motions are downloaded from the Pacific Earthquake Engineering Research (PEER) Center GM Next Generation Attenuation (NGA)-West2 database [91]. In Tables B.1 and B.2, M_w is the moment magnitude , V_{S30} is the time-averaged shear-wave velocity for the top 30 m depth of the soil from the ground surface, which is an important parameter for evaluating the dynamic behavior of soils, and R_{jb}, i.e., the "Joyner-Boore" distance, is the closest horizontal distance to the vertical projection of the rupture.

Table B.1: GM applied to the RC frame model in Section 3.1.

Location Name	**Year**	**Station Name**	$\boldsymbol{M_w}$	$\boldsymbol{R_{jb}}$	$\boldsymbol{V_{S30}}$
Northwest Calif-01	1938	Ferndale City Hall	5.50	52.73	219.31
Northern Calif-05	1967	Ferndale City Hall	5.60	27.36	219.31
Lytle Creek	1970	Wrightwood - 6074 Park Dr	5.33	10.70	486.00
Managua Nicaragua-02	1972	Managua ESSO	5.20	4.33	288.77
Point Mugu	1973	Port Hueneme	5.65	15.48	248.98
Northern Calif-07	1975	Petrolia General Store	5.20	28.48	368.72
Friuli Italy-02	1976	Buia	5.91	10.99	310.68
		Forgaria Cornino	5.91	14.65	412.37
Izmir Turkey	1977	Izmir	5.30	0.74	535.24
Dursunbey Turkey	1979	Dursunbey	5.34	5.57	585.04
Coyote Lake	1979	Coyote Lake Dam	5.74	5.30	561.43
		Gilroy Array #2	5.74	8.47	270.84
		Gilroy Array #3	5.74	6.75	349.85

		Gilroy Array #4	5.74	4.79	221.78
		Gilroy Array #6	5.74	0.42	663.31
		San Juan Bautista 24 Polk St	5.74	19.46	335.50
Norcia Italy	1979	Cascia	5.90	1.41	585.04
Imperial Valley-07	1979	El Centro Array #2	5.01	17.32	188.78
		El Centro Array #3	5.01	14.54	162.94
		El Centro Array #4	5.01	9.69	208.91
		El Centro Array #5	5.01	8.56	205.63
		El Centro Array #6	5.01	7.40	203.22
		El Centro Differential Array	5.01	7.87	202.26
		Holtville Post Office	5.01	7.69	202.89
Imperial Valley-08	1979	Westmorland Fire Sta	5.62	9.39	193.67
Livermore-01	1980	San Ramon - Eastman Kodak	5.80	15.19	377.51
Livermore-02	1980	Livermore - Fagundas Ranch	5.42	0.79	387.04
		Livermore - Morgan Terr Park	5.42	7.94	550.88
		San Ramon - Eastman Kodak	5.42	14.31	377.51
Imperial Valley-02	1940	El Centro Array #9	6.95	6.09	213.44
Northern Calif-01	1941	Ferndale City Hall	6.40	44.52	219.31
Northern Calif-03	1954	Ferndale City Hall	6.50	26.72	219.31
Parkfield	1966	Cholame - Shandon Array #5	6.19	9.58	289.56
		Cholame - Shandon Array #8	6.19	12.90	256.82
		Temblor pre-1969	6.19	15.96	527.92
San Fernando	1971	Castaic - Old Ridge Route	6.61	19.33	450.28
		LA - Hollywood Stor FF	6.61	22.77	316.46
		Lake Hughes #1	6.61	22.23	425.34
		Lake Hughes #12	6.61	13.99	602.10
		Lake Hughes #4	6.61	19.45	600.06
		Palmdale Fire Station	6.61	24.16	452.86
		Pasadena - CIT Athenaeum	6.61	25.47	415.13
		Santa Felita Dam (Outlet)	6.61	24.69	389.00
Managua Nicaragua-01	1972	Managua ESSO	6.24	3.51	288.77
Friuli Italy-01	1976	Tolmezzo	6.50	14.97	505.23
Imperial Valley-06	1979	Aeropuerto Mexicali	6.53	0.00	259.86
		Agrarias	6.53	0.00	242.05
		Brawley Airport	6.53	8.54	208.71
		Calexico Fire Station	6.53	10.45	231.23
		Cerro Prieto	6.53	15.19	471.53
		Chihuahua	6.53	7.29	242.05
		Coachella Canal #4	6.53	49.10	336.49
		Compuertas	6.53	13.52	259.86
		Delta	6.53	22.03	242.05

		EC County Center FF	6.53	7.31	192.05
		El Centro - Meloland Geot. Array	6.53	0.07	264.57
		El Centro Array #1	6.53	19.76	237.33
		El Centro Array #10	6.53	8.60	202.85
		El Centro Array #11	6.53	12.56	196.25
		El Centro Array #12	6.53	17.94	196.88
		El Centro Array #13	6.53	21.98	249.92
		El Centro Array #3	6.53	10.79	162.94
		El Centro Array #6	6.53	0.00	203.22
		El Centro Array #7	6.53	0.56	210.51
		El Centro Differential Array	6.53	5.09	202.26
		Holtville Post Office	6.53	5.35	202.89
		Niland Fire Station	6.53	35.64	212.00
		Parachute Test Site	6.53	12.69	348.69
Kern County	1952	Taft Lincoln School	7.36	38.42	385.43
Tabas Iran	1978	Boshrooyeh	7.35	24.07	324.57
		Dayhook	7.35	0.00	471.53
Trinidad	1980	Rio Dell Overpass E Ground	7.20	76.06	311.75
		Rio Dell Overpass W Ground	7.20	76.06	311.75
Taiwan SMART1(45)	1986	SMART1 C00	7.30	56.01	309.41
		SMART1 E01	7.30	53.31	308.39
		SMART1 E02	7.30	51.35	671.52
		SMART1 I01	7.30	56.18	275.82
		SMART1 I07	7.30	55.82	309.41
		SMART1 M01	7.30	56.87	268.37
		SMART1 M07	7.30	55.11	327.61
		SMART1 O01	7.30	57.90	267.67
		SMART1 O02	7.30	57.13	285.09
		SMART1 O04	7.30	55.18	288.24
		SMART1 O06	7.30	53.99	293.46
		SMART1 O07	7.30	54.17	314.33
		SMART1 O08	7.30	54.80	357.43
		SMART1 O10	7.30	56.94	320.11
		SMART1 O12	7.30	58.00	303.36
Cape Mendocino	1992	Eureka - Myrtle West	7.01	40.23	337.46
		Fortuna - Fortuna Blvd	7.01	15.97	457.06
Landers	1992	Amboy	7.28	69.21	382.93
		Barstow	7.28	34.86	370.08
		Boron Fire Station	7.28	89.69	291.03
		Coolwater	7.28	19.74	352.98
		Desert Hot Springs	7.28	21.78	359.00

		Fort Irwin	7.28	62.98	367.43
		Indio - Coachella Canal	7.28	54.25	339.02
		Joshua Tree	7.28	11.03	379.32
		Mission Creek Fault	7.28	26.96	355.42
		Morongo Valley Fire Station	7.28	17.36	396.41
		North Palm Springs	7.28	26.84	344.67

Table B.2: GM applied to the CBS frame model in Section 3.2 (AS: After Shock).

Location Name	**Year**	**Station Name**	M_w	R_{jb}	V_{S30}
Anza-02	2001	Borrego Springs - Scripps Clinic	4.92	38.05	357.64
		El Centro Array #10	4.92	119.04	202.85
Big Bear City	2003	Devore - Devore Water Company	4.92	51.08	526.24
Chi-Chi	1999	CHY019	7.62	49.98	497.53
		CHY025	7.62	19.07	277.5
		CHY041	7.62	19.37	492.26
		CHY059	7.62	73.26	191.09
		CHY059	7.62	73.26	191.09
		CHY076	7.62	42.15	169.84
		CHY093	7.62	49.82	190.49
		HWA011	7.62	49.29	355.76
		HWA017	7.62	47.04	578.11
		HWA020	7.62	39.8	626.43
		HWA049	7.62	46.65	508.61
		ILA004	7.62	86.61	124.27
		TAP095	7.62	107.8	206.24
		TAP097	7.62	97.26	237.23
		TCU029	7.62	28.04	406.53
		TCU036	7.62	19.83	478.07
		TCU036	7.62	19.83	478.07
		TCU048	7.62	13.53	551.21
		TCU052	7.62	0.0	579.1
		TCU064	7.62	16.59	645.72
		TCU065	7.62	0.57	305.85
		TCU067	7.62	0.62	433.63
		TCU081	7.62	55.48	430.47
		TCU101	7.62	2.11	389.41
		TCU103	7.62	6.08	494.1
		TCU110	7.62	11.58	212.72
		TCU111	7.62	22.12	237.53

		TCU113	7.62	31.05	230.3
		TCU113	7.62	31.05	230.3
		TCU117	7.62	25.42	198.58
		TCU117	7.62	25.42	198.58
		TCU120	7.62	7.4	459.34
		TCU128	7.62	13.13	599.64
		TCU141	7.62	24.19	223.04
		TCU147	7.62	70.61	537.92
Chi-Chi-02	1999	CHY019	5.9	92.47	497.53
		HWA029	5.9	45.17	614.05
		ILA063	5.9	80.14	996.51
		TCU115	5.9	49.99	215.34
Chi-Chi-03	1999	TCU063	6.2	33.59	476.14
		TCU075	6.2	18.47	573.02
		TCU106	6.2	35.3	451.37
		TCU113	6.2	41.07	230.3
		TCU116	6.2	21.09	493.09
Chi-Chi-04	1999	CHY002	6.2	37.21	235.13
		CHY016	6.2	79.77	200.86
		CHY025	6.2	29.2	277.5
		CHY088	6.2	48.38	318.52
		CHY092	6.2	33.02	253.72
		CHY115	6.2	90.38	259.43
		TCU140	6.2	53.03	223.6
Chi-Chi-05	1999	CHY006	6.2	52.99	438.19
		CHY029	6.2	54.31	544.74
		CHY035	6.2	52.61	573.04
		CHY055	6.2	94.31	225.77
		CHY088	6.2	76.09	318.52
		HWA017	6.2	48.32	578.11
Chi-Chi-06	1999	TCU048	6.3	38.5	551.21
		TCU141	6.3	44.62	223.04
Christchurch	2011	Christchurch Botanical Gardens	6.2	5.52	187.0
		LINC	6.2	18.47	263.2
		ROLC	6.2	24.25	295.74
		Riccarton High School	6.2	9.43	293.0
		Styx Mill Transfer Station	6.2	11.24	247.5
Chuetsu-oki	2007	Joetsu Ogataku	6.8	16.77	414.23
		Joetsu Uragawaraku Kamabucchi	6.8	18.6	655.45
		Kariwa	6.8	0.0	282.57
		Mitsuke Kazuiti Arita Town	6.8	11.35	274.23

		NIGH07	6.8	46.84	528.19
		Nagaoka	6.8	3.97	514.3
		TCG002	6.8	109.84	616.18
Darfield	2010	Canterbury Aero Club	7.0	14.48	280.26
		Christchurch Resthaven	7.0	19.48	141.0
		DORC	7.0	29.96	280.26
		GDLC	7.0	1.22	344.02
		Papanui High School	7.0	18.73	263.2
		ROLC	7.0	0.0	295.74
		SPFS	7.0	29.86	389.54
		Styx Mill Transfer Station	7.0	20.86	247.5
Denali	2002	Carlo (temp)	7.9	49.94	399.35
		R109 (temp)	7.9	42.99	341.56
		TAPS Pump Station #08	7.9	104.17	424.9
		TAPS Pump Station #10	7.9	0.18	329.4
		TAPS Pump Station #11	7.9	126.39	376.1
Duzce	1999	Lamont 1058	7.14	0.21	529.18
El Mayor-Cucapah	2010	Cerro Prieto Geothermal	7.2	8.88	242.05
		Chihuahua	7.2	18.21	242.05
		Glamis - Black Mountain Rd	7.2	89.69	743.0
		RIITO	7.2	13.7	242.05
		San Diego - Hwy 15 & Ocean	7.2	127.37	371.81
		Temecula - 6th & Mercedes	7.2	159.89	416.15
Gilroy	2002	Santa Clara - Hwy 237/Alviso	4.9	58.82	188.87
Gulf of California	2001	El Centro Array #7	5.7	100.31	210.51
Hector Mine	1999	Bombay Beach Fire Station	7.13	120.69	257.03
		Desert Hot Springs	7.13	56.4	359.0
		Fort Irwin	7.13	65.04	367.43
		Little Rock Post Office	7.13	146.51	442.02
		San Bernardino - Del Rosa Sta	7.13	96.91	642.83
		San Bernardino - E & Hospitality	7.13	105.2	296.97
		San Bernardino - Fire Sta. #10	7.13	103.62	279.46
		San Bernardino - Fire Sta. #10	7.13	103.62	279.46
Iwate	2008	AKT003	6.9	131.65	506.26
		AKT009	6.9	118.96	514.86
		AKT013	6.9	67.75	636.67
		AKT015	6.9	74.75	135.4
		IWT014	6.9	36.75	314.6
		MYG013	6.9	63.53	252.68
		MYG017	6.9	95.3	122.07
Kobe	1995	Port Island (0 m)	6.9	3.31	198.0

		Shin-Osaka	6.9	19.14	256.0
Kocaeli	1999	Arcelik	7.51	10.56	523.0
		Atakoy	7.51	56.49	310.01
		Goynuk	7.51	31.74	347.62
L'Aquila (AS1)	2009	Celano	5.6	19.95	612.78
		Lab.Gran Sasso	5.6	17.86	547.0
L'Aquila	2009	Avezzano	6.3	23.67	199.0
Landers	1992	Amboy	7.28	69.21	382.93
		Baker Fire Station	7.28	87.94	324.62
		Barstow	7.28	34.86	370.08
		Barstow	7.28	34.86	370.08
		Downey - Co Maint Bldg	7.28	157.46	271.9
		Forest Falls Post Office	7.28	45.34	436.14
		Indio - Coachella Canal	7.28	54.25	339.02
		LA - N Westmoreland	7.28	159.13	315.06
		LA - Obregon Park	7.28	151.7	349.43
		La Habra - Briarcliff	7.28	143.12	338.27
		Mission Creek Fault	7.28	26.96	355.42
		Pomona - 4th & Locust FF	7.28	117.5	384.44
		Puerta La Cruz	7.28	94.48	442.7
		Twentynine Palms	7.28	41.43	635.01
Little Skull Mtn	1992	Station #4-Pahrump 2	5.65	61.04	352.05
Mohawk Val	2001	Sparks - Fire Station #2	5.17	79.18	359.21
Niigata	2004	FKS022	6.63	64.25	211.76
		ISKH02	6.63	150.77	720.76
		NIG014	6.63	25.14	128.12
Northridge-01	1994	LA - Baldwin Hills	6.69	23.5	297.07
		LA - Brentwood VA Hospital	6.69	12.92	416.58
		LA - City Terrace	6.69	35.03	365.22
		LA - Pico & Sentous	6.69	27.82	304.68
Parkfield-02	2004	PARKFIELD - UPSAR 02	6.0	9.49	416.82
		Parkfield - Fault Zone 15	6.0	0.8	307.59
Tottori	2000	HRS014	6.61	142.65	218.06
		TTRH02	6.61	0.83	310.21
Umbria Marche (AS16)	1998	Sellano Ovest	5.4	37.19	509.0
Yorba Linda	2002	Los Angeles - Acosta Residence	4.27	40.61	330.03

Acronyms

AI Artificial Intelligence. 1, 3, 9, 97, 109, 116

ANN Artificial Neural Network. 11, 13

API Application Programming Interface. 53, 98, 99

ASCE American Society of Civil Engineers. 1, 34

CAV Cumulative Absolute Velocity. vi, 10, 91, 93–96, 114

CBS Concentrically Braced Steel. i, iii, iv, vi, vii, 36–38, 54, 59–65, 67, 77, 91, 92, 111, 128, 131

CNN Convolutional Neural Network. iii, 13, 16–18, 25, 69, 103, 105

CPU Central Processing Units. 53

CSMIP California Strong Motion Instrumentation Program. iv, 67, 73, 74

CV Computer Vision. 12, 16, 25, 69, 102

DI Directed Information. ii, vi, 2, 3, 5, 6, 26, 28–30, 83, 85, 86, 88, 89, 92, 93, 109, 113

DL Deep Learning. i, 1, 4–6, 12, 16, 18, 20, 53, 109

DM Damage Measures. iii, 11, 115

DOF Degree of Freedom. 34, 50, 51, 55

DV Decision Variables. iii, 11, 115

EDP Engineering Demand Parameters. 5, 11, 66, 115

FC Fully Connected. iii, 16–18, 24, 48, 50, 67, 68, 103

FEM Finite Element Method. ii–iv, vi, 5, 6, 10, 31, 32, 34–36, 38–40, 44, 47, 54, 56, 59, 60, 66, 77, 85, 86, 88, 91, 92, 109–111, 126, 128

FFT Fast Fourier Transform. 69

FIM Fisher Information Matrix. 84

GA Genetic Algorithm. iv, 39, 41, 43, 44, 84

GM Ground Motion. ii, vii, 6, 34, 36, 38–42, 46, 47, 52, 53, 56, 57, 59, 67, 69, 75, 78, 86, 89, 90, 92, 94, 110, 112, 127, 128, 131

GPU Graphic Processing Units. 16, 53

GRU Gated Recurrent Unit. iii, 21, 22

IM Intensity Measures. iii, 11, 115

IoT Internet of Things. 9

KL-divergence Kullback-Leibler divergence. 64

LR Logistic Regression. 13, 103, 105

LSTM Long Short-Term Memory. i, iii, iv, vi, 1, 2, 4, 5, 9, 16, 21–25, 32, 46–49, 54, 56–58, 61–71, 73, 75–83, 91, 95–97, 109–114

LSV Latent Space Vector. 1, 4, 5, 24, 47–50, 56, 57, 61, 64, 65, 110, 111

MA Moving Average. 11, 52

MAC Modal Assurance Criterion. 84

MDOF Multi Degree of Freedom. 50, 51

MI Mutual Information. 27–29

ML Machine Learning. i, iii, vi, 1, 3–5, 9–16, 20, 44, 49, 53, 91, 93–97, 109, 113–115

NGA Next Generation Attenuation. 34, 38, 128

NHE Natural Hazards Engineering. 101

NLP Natural Language Processing. 2, 4, 24, 109, 116

NN Neural Network. ii, iii, 13, 16–20, 24, 48, 50, 65–68, 84, 85, 110, 112

OSP Optimal Sensor Placement. ii, vi, viii, 2, 3, 6, 15, 26, 30, 47, 83–85, 89–96, 109, 113, 115, 116

PAGER Prompt Assessment of Global Earthquake for Response. v, 98, 99, 102

PBE Performance-Based Engineering. 115

PBEE Performance-Based Earthquake Engineering. iii, 5, 11, 12, 66, 115

PDF Probability Density Function. 13, 55, 76, 86

PEER Pacific Earthquake Engineering Research. iii, ix, 11, 34, 38, 86, 98, 107, 128

PGA Peak Ground Acceleration. iii, 11, 40, 41, 98

PGV Peak Ground Velocity. 98

RC Reinforced Concrete. i–iv, vi, vii, 6, 31–35, 38, 54–61, 111, 126–128

RDEE Resilient Design for Extreme Events. 115

RNN Recurrent Neural Network. iii, 1, 4, 13, 16, 18–21, 24–26, 46, 48

SHM Structural Health Monitoring. i, iii, iv, 1–5, 7–13, 17, 18, 46, 47, 49, 50, 54–56, 67, 83, 85, 91, 93, 94, 109, 110, 113, 115, 116

SI System Identification. 10, 84

StEER Structural Extreme Events Reconnaissance. 100, 101, 103

SVM Support Vector Machine. 12, 13, 94, 103, 105, 114

t-SNE t-distributed Stochastic Neighbor Embedding. iv, 61, 64, 65

TCN Temporal Convolutional Network. iv, 2, 5, 67, 71–73, 75, 76, 78, 79, 82, 112

TS Time Series. iv, 1, 4–6, 10, 12, 18, 24, 25, 46–49, 51–55, 57–59, 66, 67, 77–79, 83, 86–91, 94, 95, 109–114

USGS United States Geological Survey. v, 2, 4, 98–100, 102, 103, 114

VAST Virtual Assessment Structural Team. 101

WN White Noise. 40

Symbols

$a(t)$ acceleration time history.

b bias parameter.

c damping matrix.

c_t cell state vector.

$\tilde{c_t}$ cell input activation vector.

C_n^m combination.

d dilation factor.

f_t forget gate activation vector.

h_t hidden state vector.

i_t input gate activation vector.

k stiffness matrix; or kernel size.

l log-likelihood function.

L loss function.

m mass matrix.

M_w moment magnitude.

$\mathcal{N}(\mu, \sigma^2)$ normal distribution with mean μ and standard deviation σ.

o_t output gate activation vector.

$p(t)$ time-varying external force vector.

q scalar modal coordinate.

$Q(t)$ quality of system (%).

r_t reset gate activation vector.

t time or time step.

t_0 time of occurrence of earthquake.

t_1 time of recovery to full functionality.

t_{max} duration of ground motion record.

t_r recovery time.

t^* maximum acceptable recovery time.

u displacement; or structural response parameter.

$\dot{u}$ velocity.

$\ddot{u}$ acceleration.

W weight parameter.

x_t input vector.

y_t output vector.

z_t update gate activation vector.

α learning rate.

$\alpha_{tt'}$ attention weight between two steps.

β parameter in Newmark's time-stepping method.

γ parameter in Newmark's time-stepping method.

γ_φ resiliency of system.

θ structural properties.

σ sigmoid activation function.

Φ mode shape.

$|\bullet|$ cardinality of $\bullet$.

$\|\bullet\|$ ℓ_2 norm of $\bullet$.

$\odot$ Hadamard product.