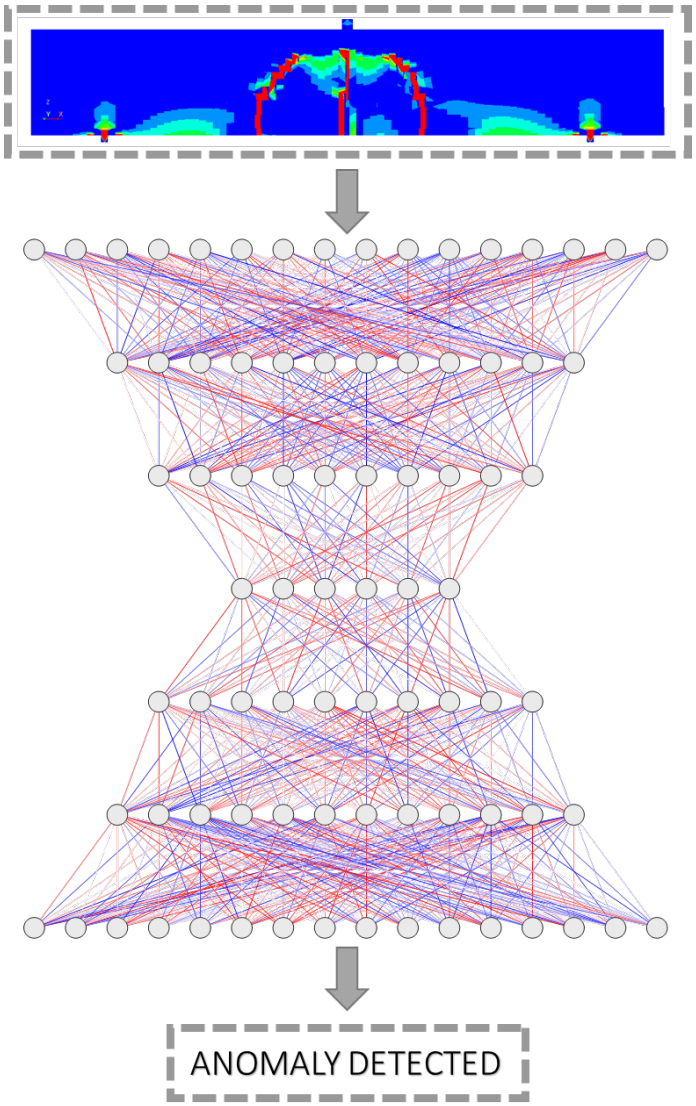
CHALMERS
UNIVERSITY OF TECHNOLOGY



Structural Health Monitoring of Concrete Elements Using Deep Machine Learning

Master's thesis in Complex Adaptive System

Dimitrios Karypidis

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

This page is intentionally left blank

Master's thesis 2019:NN

Structural Health Monitoring of Concrete Elements Using Deep Machine Learning

KARYPIDIS DIMITRIOS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
SensIT project
Chalmers University of Technology
Gothenburg, Sweden 2019

Structural Health Monitoring of Concrete Elements Using Deep Machine Learning
DIMITRIOS KARYPIDIS

Supervisor(s):
Carlos Gil Berrocal, Department of Architecture and Civil Engineering
Rasmus Rempling, Department of Architecture and Civil Engineering
Mats Granath, Department of Physics

Examiner: Mats Granath, Department of Physics

Master's Thesis 2019:NN
Department of Physics
SensIT Project
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Strain profile of a 2D beam constructed in DIANA Finite Element Analysis.

Typeset in LaTeX

Gothenburg, Sweden 2019

This page is intentionally left blank

Abstract

The unique nature of Structural Engineering allows the field to integrate fresh innovations in its applications only at a slow pace. However, recent advancements in networking and artificial intelligence can greatly upgrade the current processes. This thesis reports the early findings of an ongoing project aimed at developing new methods to upgrade the current maintenance strategies of the civil and transport infrastructure. As part of these new methods, the use of Machine Learning (ML) algorithms is being investigated to constitute the core of a new generation of more accurate and robust structural health monitoring (SHM) systems for concrete structures. Unlike most of the existing SHM systems, relying on the analysis of the natural frequencies of the structure based on data obtained from accelerometers, the present study uses a distributed optic fiber system to monitor the strain distribution along steel reinforcing bars. The preliminary results of the study indicate that a semi-supervised Deep Autoencoder algorithm (DAE) can successfully quantify the damage attributable to transverse cracks in a reinforced concrete beam subjected to three-point loading. Future applications will feature the determination of crack locations, early detection of reinforcement corrosion as well as other types of damage such as splitting cracks or surface spalling.

Keywords: structural health monitoring, machine learning, deep autoencoders, anomaly detection, concrete structures, distributed optic fiber.

This page is intentionally left blank

This thesis was conducted as a proof of concept under the SensIT project, carried out at the Department of Architecture and Civil Engineering, Chalmers University of Technology.
Part of this thesis is submitted and at the time of writing under review at the 2019 IABSE Congress New York City [26].

This page is intentionally left blank

Acknowledgements

I would like to thank Rasmus Rempling for introducing me to the SensIT team, Carlos Gil Berrocal who orchestrated the project planning and patiently walked me through the modeling parts of the project, of which I was not so familiar with and Sebastian Almfeldt for setting up and executing all the experiments. I would also like to thank my examiner and supervisor, Mats Granath, who guided me throughout this thesis. Finally, special thanks to my fellow CAS classmates, with whom I had an amazing time these 2 years at Chalmers.

Dimitrios Karypidis, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1

Introduction

1.1 Preface

We live in a fascinating era. The advent of internet can inarguably be considered as one of the most impactful events that changed the course of humanity. This instant, global interconnection has drastically changed all aspects of our lives, from social relationships to newer and ground breaking technologies built on top of it. The most recent niche is the **Internet of Things** (**IoT**), which can be (partially) truncated to *"An open and comprehensive network of intelligent objects that have the capacity to auto-organize, share information, data and resources, reacting and acting in face of situations and changes in the environment"* [30]. One important application of IoT is designing *real-time monitoring systems*, by constantly receiving, processing and assessing stream of data taken from strategically placed sensors.

Handling and using big streams of data is a tedious task, which has occupied data scientists for the past years. One of the most promising tools for handling such tasks is **Machine Learning** (**ML**). ML is a sub-domain of artificial intelligence. By quoting Arthur Samuel (1959), one of the most important pioneers in the fields of computer gaming and artificial intelligence, ML is :*"the field of study that gives computers the ability to learn without being explicitly programmed"*[42]. In recent years, with the increase in computational capacity of modern computers, a particular sub-field of ML called **Deep Learning** (**DL**) has dominated the research, since it seems to be the best known approach to prediction and classification [20].

DL has successfully been implemented, and is considered to be the state of the art tools in a plethora of applications such as image recognition, self-driving cars, machine translation, financial time series prediction, anomaly detection etc. Building monitoring systems using DL systems is widely used. Unfortunately, these approaches have only recently started being popular in the field of structural engineering, via the introduction of **Structural Health Monitoring** (**SHM**).

SHM is the constant monitoring of structural systems to detect, localize and assess irregularities and defects. Despite the successful implementation of SHM in numerous sectors like the aerospace and automotive industry, its application in the transport infrastructure has been hindered by the singular nature of civil structures. Current SHM systems still rely on deterministic methods, such as signal processing and **Finite Element Analysis** (**FEA**). These techniques, although very practical when used by themselves, suffer from two main issues: (**i**) **non-robustness to noise**, i.e. noise can impair the predictive capacity of our monitoring system, and (**ii**) **inflexibility**, i.e. the monitoring system cannot adapt its predictions in case of permanent changes in the structural system. Nevertheless, more recent approaches

have applied DL techniques in order to optimize the SHM procedures, most of them falling under the category of object detection (computer vision) and multi-feature classification[44, 17, 18, 9],
which sometimes are non-practical to implement in large scale projects.

1.2 Goals

The goal of this thesis is to develop the algorithmic part of an SHM system, based on DL, that will be monitoring a simply supported **reinforced concrete** (**RC**) beam, without any transverse reinforcement. In this stage, the monitoring will be based only on the strains along the longitudinal reinforcement. The reason behind this is that the experimental RC beams for the cracking experiments will be equipped with optical fiber sensors.

1.3 Approach

In this section we will briefly describe how this thesis sub-tasks were tackled, which will be described in more details in the upcoming chapters.

1.3.1 Data acquisition and processing

The data used were gathered from both computer simulations and real experiments. The software used for the numerical experiments was DIANA Finite Element Analysis (Version 10.2). The experimental data was generated from three-point loading tests on RC beams conducted at the Chalmers structural engineering lab.

1.3.2 Monitoring algorithms

The DL architecture used in this application is Deep Autoencoders, which will be discussed in detail in a latter section. Different configurations of the networks were tested, in order to optimize the results.

1.4 Results

Using the results from the implementation above-mentioned, the network was tested. Apart from a successful mapping of the states of the beam, we were able to construct a rule for classifying the strain state of the concrete element.

1.5 Scope and limitations

The current work was tested on simply supported beams only. In order to examine its effectiveness, the application must be extended into more complex structures, with different load combinations that will mimic real scenarios of structural overloading.

Additional work must also be conducted in examining and categorizing the proper criteria for damage classification.

1.6 Thesis Outline

In this section we will present the thesis' chapters, along with a small description of them.

1. *Introduction*: In this chapter we addressed the main motivations behind this thesis and presented a small outline of the preceding chapters.
2. *A brief overview on SHM*: SHM is the backbone of this application. In this chapter we will shortly present the basic principles behind SHM, as well as some important applications.
3. *Deep Neural Networks Overview*: The current application is carried out using a Deep Neural Network (DNN) architecture, which is one of the most recent ML fields. DNN's are inspired by neuroscience, and have been vastly applied the past decades in many fields. In this chapter we will have a short overview on biological neurons and the historical evolution of DNN's. There will be a brief explanation on how DNN's work, with an additional part on Deep Autoencoders, which is the architecture that was used.
4. *Methodology*: In this chapter will be explained in detail the methodology used in this thesis. That is, the exact network configurations, the way data was prepossessed before training and how the different states of the beams were classified.
5. *Results and Discussion*: In this chapter will be presented all the results and plots that were produced. In addition, the efficiency of the algorithm will be discussed, regarding findings. Finally, there will be a small discussion about possible further research on the topic.

2

A brief overview on SHM

The current chapter will be an introduction to SHM. A small description behind the motivation and the importance of SHM will be presented. Afterwards a small historical overview will follow, with an emphasis in SHM applications in bridges. In the end we will present the basic principles of SHM.

2.1 Preface

As society evolves technologically, people's well-being is becoming more complex and fragile. Nowdays we rely on the use of sophisticated structural, mechanical and electrical structures, such as roads, transport infrastructure and power grids. The frequent assessment of these public-access structural systems is of crucial need for society. Identifying and locating a defect in time, before severely impairing the integrity of the whole structure is vital for public safety, financial efficiency and environmental sustainability. Among structural engineering structures, the most important are probably bridges, since by nature they are more large-scale, cost demanding and extremely important for the financial growth of a state. In Sweden, there are more than 20000 bridges, which, taking into consideration the harsh environmental conditions and the constant traffic loading in which they are exposed to, renders their monitoring and proper maintenance an incredibly complicated task. This is really important regarding the environmental footprint of concrete bridges, since in a life cycle assessment study conducted in 2005 [27], it was shown that the total energy consumption of a conventional reinforced concrete bridge with a service life of 60 years is about 46000 Gigajoules and the total CO_2 equivalent is about 3600 tons (including traffic).

In addition, the recurrent amount of money spent in inspecting, maintaining and repairing existing bridges is enormous. According to the OECD database for transport infrastructure investment and maintenance spending [35], in 2015 Sweden invested nearly 3250 million euros in the transport infrastructure including road and railway structures. Approximately two thirds of the total investment (2094 million euros) were used for preservation of the existing transport network with a high percentage of the expenditure devoted to inspection and maintenance operations of deteriorated concrete structures [3]. However, it must be noted that most of the investigation works are scheduled either recurrently or whenever an apparent damage has emerged. The first cases may result in unnecessary or belated investigations, and the second in costly rehabilitation works that could have been minimized if the damage was detected in advance. It is thus obvious that targeted and punctual monitoring will result in optimum maintenance, which will allow the bridges

to exhaust their service cycle, is extremely important for a sustainable environment and a healthy economy. In order to achieve that, the most popular process is called Structural Health Monitoring.

2.2 Historical overview

Structural Health Monitoring (SHM) refers to the damage detection discipline that involves the periodic or constant monitoring of a system by means of damage measurement and localization. Almost all major industries apply some form of SHM in their products, services or facilities, since detecting a defect in time is of great significance. Early damage detection can have a crucial impact on both life-safety and optimal resource management. Especially nowadays, industry extends the use of their infrastructure beyond their initial lifespan, thus rendering their detailed and frequent monitoring even more important.

2.2.1 Early stages

It is reasonable to assume that humans have tried to assess the integrity of structures since the early stages of civilization. In antiquity, people were using visual inspection to assess the condition of structures. The first recorded in-situ SHM application was recorded in the 19^{th}, where railroad wheel-tappers were striking their hammers on the train wheels (Fig 2.1), in order to evaluate by the generated sound whether damage was present [21, 15], a method that is still occasionally used today.

Figure 2.1: Wheel-taper monitoring train wheels with the use of acoustics

The first systematic SHM application was on rotating machinery. In fact, SHM for rotating machinery is mostly known as Condition Monitoring [13]. The initial approaches were pretty crude. The inspector would touch a screwdriver on the machine and would assess its integrity by changes in the acoustic signals or in the vibrations [33]. Since then, monitoring of rotating machines still uses the same data as it did before, but the data is collected using modern data acquisition systems. These include various types of sensors, which are more robust and reliable compared to human perception [19].

2.2.2 SHM in bridge monitoring

Historically, bridge monitoring programs were pivotal for understanding and refining computational models of the load–structure–response chain. One of the earliest documented systematic bridge monitoring exercises [8], was conducted on the Golden Gate and Bay Bridges in 1937 in San Francisco (Fig 2.2) in an elaborate program of measuring vibrational periods of the various components during their construction to learn about the dynamic behaviour and possible consequences of an earthquake [6].

Figure 2.2: Accelerometer antenna sensor at the top of Golden Gate bridge (credit: Steven D. Glaser and Tommi Parkkila, ERCIM News).

The next noteworthy bridge SHM application was on Tacoma Narrows Bridge in Washington, where the bridge was thoroughly monitored due to its wind-induced instability, which was the reason of the final collapse (Fig 2.3). What was monitored was the vibration measurements, which posed a major concern about the integrity of the structure [45]. This experience was really important in understanding the response of long-span suspension bridges in wind-induced vibrations. Despite the great progress, long-span bridge aerodynamics can still present unexpected behaviour [28].

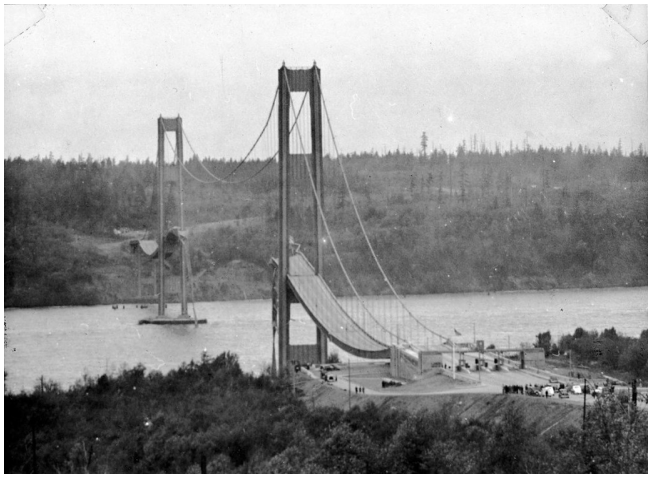


Figure 2.3: Tacoma Narrows Bridge after partial collapse.

In the last decades bridge monitoring has evolved and consequently formalized into SHM. The major bridge projects have been initially implemented in Hong Kong and Japan and more recently in USA. Long-span bridge monitoring systems also provide ideal opportunities to implement and study SHM systems; for example, the wind and SHM system [48] implemented on the Lantau fixed crossing has stimulated SHM research in Hong Kong, not only concerning the performance of the bridges themselves, but also of SHM methodologies [6].

Although SHM can greatly improve and optimize the lifespan and serviceability of a bridge structure, it is rarely applied. The reason being the high initial cost of installation, as well as the rigidity shown by the structural engineering community in adapting modern technologies. In most bridges, monitoring and health assessment are carried out by regular inspection and maintenance programs. This has proven to be sub-optimal, since after car accidents, one of the most frequent causes of reduced accessibility of the transport infrastructure and the consequent traffic delays is the performance of inspection operations and the application of maintenance measures to combat aging of the infrastructure. Consequently, the aging and deterioration of the transport infrastructure not only poses a very serious issue of public safety but also has a cascading detrimental impact on the nation's economy that negatively affects business productivity, the gross domestic product (GDP) and international competitiveness [3].

2.3 Basics of SHM

2.3.1 Fundamental Axioms

As written in the previews sections, SHM is a very modern discipline, formalized only during the recent decades. Some of the pioneers of SHM have constructed the SHM fundamental axioms [49]. Although these axioms are not based on strictly posed derivations, their significance was elaborated in the initial script. These are the following:

- **Axiom I: All materials have inherent flaws or defects.** This holds true for all kinds of materials, even for those which are produced industrially. However, engineers have overcome this problem by using the statistical properties of each material, and designing using relevant safety factors.

- **Axiom II: The assessment of damage requires a comparison between two system states.** Regardless if the SHM estimation mechanism is a sophisticated ML algorithm, a statistical pattern recognition method or a simple visual inspection, the current state must always be compared to a predefined healthy one. Detecting and analyzing the differences between these states bring to the fore possible defects, as well as their location.

- **Axiom III: Identifying the existence and location of damage can be done in an unsupervised learning mode, but identifying the type of damage present and the damage severity can generally only be done**

in a supervised learning mode. This is reasonable, since unsupervised methods cannot provide detailed information about the characteristics of the damage and are unable to classify it as severe or not. This kind of information must explicitly be provided by the designer, with prior thorough knowledge on the monitored system, thus a supervised approach is necessary.

- **Axiom IVa: Sensors cannot measure damage. Feature extraction through signal processing and statistical classification is necessary to convert sensor data into damage information.** Sensors' function is to receive raw data measured at the point of installation. This data can be noisy, and possibly inappropriate for direct processing. In most applications, some signal processing, data cleaning/preprocessing or feature extraction is necessary. After extracting the essential information, this has to be utilized by some analysis algorithm, either in the form of statistical learning or some deterministic method (e.g. FEA reverse engineering).

- **Axiom IVb: Without intelligent feature extraction, the more sensitive a measurement is to damage, the more sensitive it is to changing operational and environmental conditions.** As stated in the previous level, sensor data by itself is not utilizable and usually have some level of noise contamination. This can be caused due to imperfect installation, harsh environmental conditions or some minor defect in the sensor (see Axiom I). It is thus the designer's responsibility to construct an algorithm that takes care of such instabilities, and render the final result reliable and robust.

- **Axiom V: The length- and time-scales associated with damage initiation and evolution dictate the required properties of the SHM sensing system.** Depending on the system and monitoring devices, there might be gradual degradation or it can be a result of an infrequent event. These two case might be interpreted differently from the SHM system. This has to be taken into consideration during assessment.

- **Axiom VI: There is a trade-off between the sensitivity to damage of an algorithm and its noise rejection capability.** This axiom is very popular among statisticians and data scientists, often called *Bias–variance trade-off*. In supervised learning, whenever we increase the complexity of the trained model, we force the model to predict every observation with the maximum accuracy. This increases the variance of the predictions i.e. we fit the noise into the model. On the contrary, when we restrict the complexity of the model, our predictions have higher error, but the variance is reduced, allowing for more consistent predictions. This topic will be discussed further in the next chapter, but the designer must always find the optimal balance between those two, using and exogenous metric as a guideline.

- **Axiom VII: The size of damage that can be detected from changes in system dynamics is inversely proportional to the frequency range of excitation.** This is more relevant to vibrational measurements. In a

nutshell, when the frequency of the measured event is higher, then it is easier for the sensor to measure it, since there is a higher chance the the measurement frequency of the sensor will coincide with the event.

- **Axiom VIII: Damage increases the complexity of a structure.** The final axiom is probably the one that renders statistical methods more robust than analytical ones. Assuming that we have an ideal linear concrete beam, where concrete has no imperfections and behaves linearly in its elastic region (false by due to Axiom I). When the loading exceeds the elastic threshold, then we introduce cracks, which consequently introduce nonlinearities in the system. This dynamic behaviour is very complex to capture by analytical methods, since in reality we have an alteration in the member's behaviour.

These axioms, arbitrary as they might seem, summarize the essence of all modern SHM applications.

2.3.2 SHM principals

SHM systems, according to their functionality, are characterized differently. Rytter [40] and Lehmhus et al. [29] have formulated the following levels/categories of SHM:

- **Detection**: The system will alert whether the is damage in the structure. According to each application, this will result from the monitored measure(s) (e.g. vibrations, acceleration, strains etc).

- **Localization**: After it is verified that the system has been damaged, it will provide information on the possible location(s) of the defect.

- **Assessment**: At the same time, the system will provide feedback on the severity of the damage, in order to take the appropriate action.

- **Consequence**: After assessing the damage, the system can give an estimation about the safety and serviceability of the structure, according to its remaining capacity.

- **Prognosis**: This is more sophisticated category, where after analyzing the previous level, the system can give a prediction about its remaining service-life.

- **Self-healing** The ultimate level of automation. Given that we manage to localize and assess the defect with extreme level of precision, then it may be possible to have self-healing mechanisms, that reduce the level of human interaction for the restoration of the structure. One example might be a shape-memory allow, which can change its geometry when applying heat load [10].

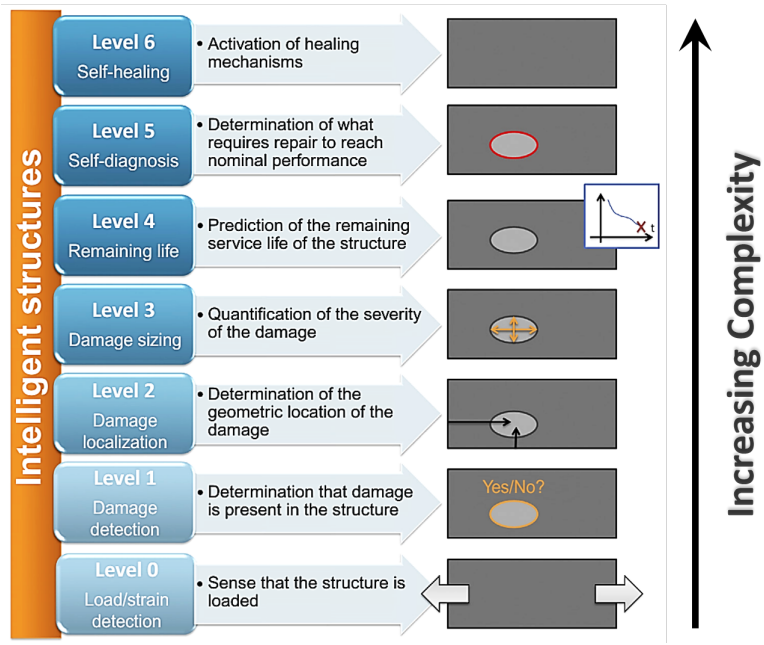


Figure 2.4: The 6 levels of SHM (taken from [1]).

In general, these categories are in ascending hierarchical order, i.e. in order to acquire information about a level, one must achieve the previous one. However, sometimes these boundaries are more fluid, and sometimes they tend to overlap. In addition, some of these levels might be omitted. All the above mentioned is in unique for each implementation, regarding the underlying problem and the available means of solving it (sensors, computational resources etc).

In the current application, the levels that were used were **Detection**, **Assessment** and **Consequence**, although localizing the damage can be easily implied using deterministic methods. More details will be presented in the next chapters.

2.4 Summary

In this chapter a brief overview on SHM was presented, with a particular focus in bridge applications. This was necessary to understand the foundations of the current thesis. After the short historical reference and the initial motivation behind SHM, the reader should have a closer look at the SHM action and the different SHM levels.

3

Deep Neural Networks Overview

In the current chapter we will present the foundation of Deep Neural Networks (DNN's). DNN's are directed, multi-layer computational graphs that try to mimic the mammalian brain. This chapter will be divided in the following parts: a brief historical overview and initial applications, the basic theory, math and algorithms behind neural networks and finally a reference to more advanced DNN structures, with an emphasis in Deep Autoencoders (DAE's). It must be noted that in this chapter we will not address the networks optimizers, apart from the vanilla gradient descent. Optimization is nonetheless a very important part of DNN's design [20].

3.1 Preface

Machine learning (ML) is the natural evolution of statistical learning. It is considered to be the most dominant subset of Artificial Intelligence (A.I.). In a nutshell, ML is the computer science discipline of building and using adaptive algorithms that iteratively learn to tackle a task without being explicitly programmed to do so. Common tasks solved using machine learning are regression and classification.

One particularly important group of ML algorithms are Deep Neural Networks (DNN's). DNN's are Artificial Neural Networks (ANN's) consisting of several stacked intermediate layers. ANN's are computational graphs that can be modified through proper training. They have been a field of study since the 40's, but because of the scientific and technological immaturity, have not been popular until the 80's. Current technological development and easier access to greater computational resources have made DNN's a prominent tools in both the industry and academia, in almost all scientific fields.

3.2 Brief overview

3.2.1 Biological neurons

ANN's were heavily inspired by the human brain. The human brain, and consequently the mammalian brain, is part of the central nervous system. The human brain is often characterized as the most complex object that has ever been discovered in the universe [25]. In terms of adaptability, the human brain is the best processing unit. It can process large amounts of complex, inconsistent, noisy data with robustness and remarkable precision. In addition, it is extremely efficient in adapting into new environments and stimuli, without having to explicitly be rewired like a conventional sensor-based computational system.

The basic structural component of the brain is the biological neuron. There are at least 10^{11} neurons in the human brain, which form about 10^{14} synaptic connections [14]. Neurons consist of 4 parts: **cell body**, **dendrites**, **axon** and **synapses** (Fig 3.1). Neurons are a type of cell, thus they have a main body which contain all the necessary material for the cell to live. The cell body contains the cell's core, which in turn contains the cell's genetic material. Inside the body all the chemical processes of the neuron take place, such as enzymes and protein synthesis. Neurons transmit information using electrical signals, via the axon. The axon itself is divided into several branches, called dendrites. Each branch is forming a connection with a neighboring neuron, in order to establish a communication link via the transfer of electrical signals. The two branches are not in contact. In fact a microscopic space is formed, namely the synapse (synaptic cleft), which according to the distance and the chemical composition of the surrounding fluid, the neural activity weakens or strengthens [46]. Of course this is not a detailed overview of how biological neurons work, but it is sufficient to examine how were their artificial counterparts inspired.

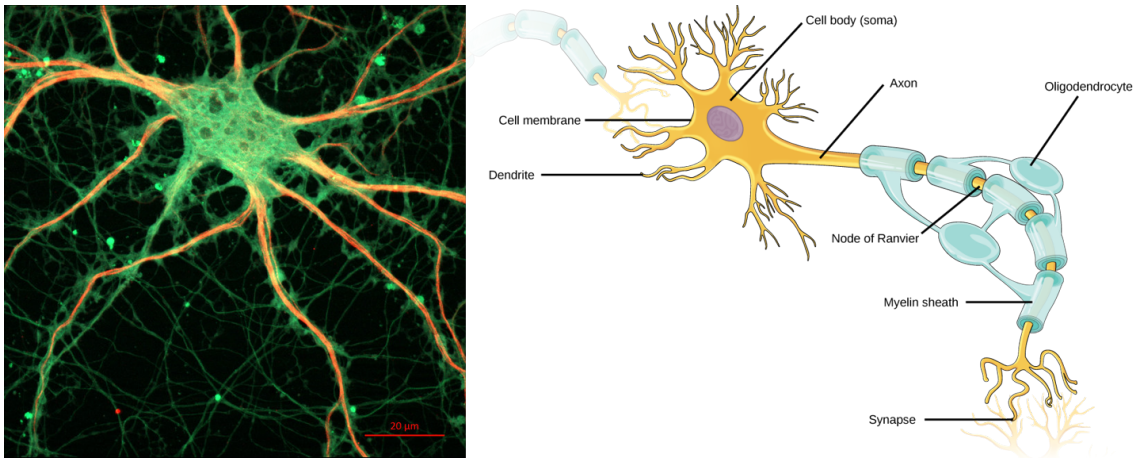


Figure 3.1: *Left picture*: A microscope photo of a system of biological neurons (rat hippocampus). *Right picture*: A labeled schematic picture of the structure of a biological neuron

3.2.2 Historical overview

For the rest of the report, we will use the terms "neuron" and "node" interchangeably. The first artificial neuron was formalized in 1943 by McCulloch and Pitts, with the initial name threshold logic unit (TLU)[31]. For the i_{th} neuron, the signal is propagated as follows:

$$y_i = f(\sum_{j=1}^{m} W_{ij}x_j - b_i) \tag{3.1}$$

where W_{ij} is the weight matrix, which shows the the percentage of the signal passing from node i to node j, x_j is the j_{th} input signal and b_i is the *bias* term, which serves the purpose of shifting the decision boundary line (or any hyper-plane in multidimensional problems). The function f is called the activation (or transfer function), which regulates in what manner will the neuron be activated, and thus

what it will export in the preceding nodes (Fig 3.2). As the its name suggests, in TLU the activation function that was used was the Heaviside step function:

$$f(x) = \begin{cases} 1, & \geq 0 \\ 0, & otherwise \end{cases} \tag{3.2}$$

As we will discuss below, this activation function has the disadvantage that is has a constant derivative (also not defined at x = 0), thus it is not widely used anymore. It must be noted though that it is similar to how signal is propagated in biological neurons.

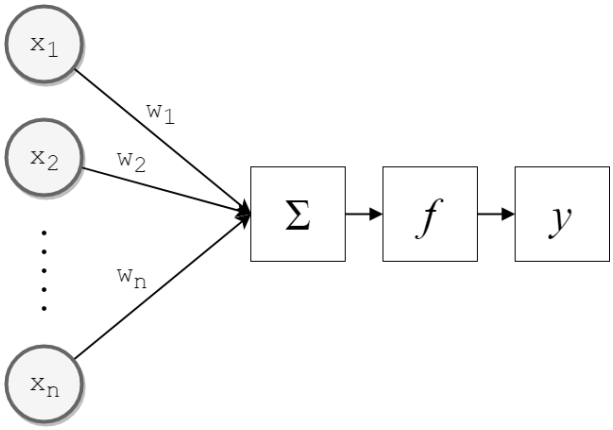


Figure 3.2: The pipeline of an artificial neuron

The next notable progress took place in 1958 by Rosenblatt [38], who evolved the TLU into the famous perceptron. Perceptron was similar to TLU, with the major difference that now there was a learning rule to adjust the weight matrix. The rule is know as delta rule, which uses gradient descent to minimize the error from a perceptron network's weights. Thus, the a weight connection w_{ij} is updated as follows:

$$w_{ij}^{t+1} \leftarrow w_{ij}^{t} + \Delta w_{ij}^{t} \tag{3.3}$$

The rate of change of the current connection is proportional to the contribution of it to the final error. Thus:

$$\Delta w_{ij}^{t} = -\eta \frac{\partial E}{\partial w_{ij}^{t}} \tag{3.4}$$

where η is the learning rate, which regulates the speed of the training process. It must be chosen wisely, since too little might lead to premature convergence, and too much might lead to oscillations around the global optimum (overshooting).

These initial architectures could only be utilized to solve linearly separable problems, which lead to their loss of popularity. Neural nets started being popular again in 1982, after the discovery of Hopfield Networks [23], which were proven to have the ability to store and retrieve a memory, partially or completely. One of the most important pieces of work, which is considered to be the one that initiated the rise of DL was published in 1986 by David Rumelhart, Geoffrey Hinton and Ronald

Williams [39], which combined together previews papers and formalized the final form of Multilayer Perceptron (later known as DNN) [47] which was trained using the back-propagation algorithm [32].

3.3 DNN: architecture and functionality

A DNN consists of several stacked layers of computational nodes. Each node is connected with all the nodes of the preceding layer. Herein a DNN is a unidirectional computational graph, which readjusts its adjacency matrix. A schematic diagram of a DNN is presented below (Fig 3.3):

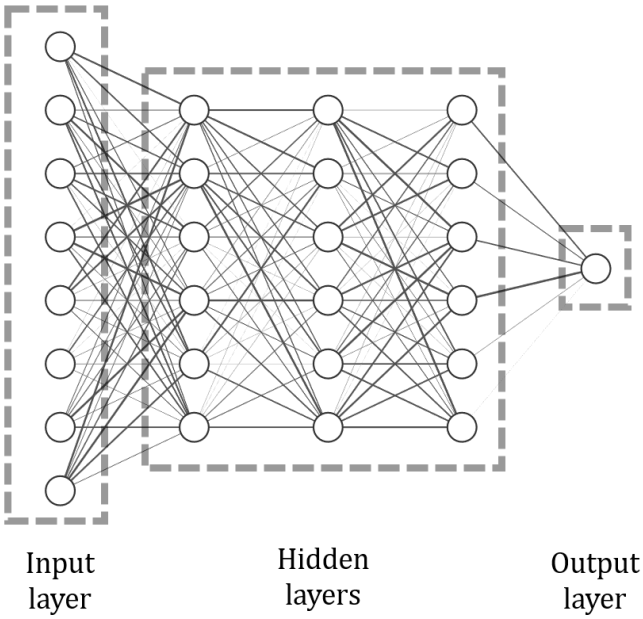


Figure 3.3: A schematic representation of a Deep Neural Network

3.3.1 Transfer functions

The output of each node is dictated by Eq. 3.1. Let us examine how different activation functions determine the properties of the neuron's output. Since DNN training uses the activation function's derivative, as we will discuss in backpropagation, each activation function will be defined along with its derivative:

3.3.1.1 Logistic

The logistic function is a subset of the sigmoid functions family. They are called sigmoid because the shape resembles an "S" shape. The function is the following:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

$$f'(x) = f(x)(1 - f(x)) \tag{3.6}$$

The range of the output is $(0, 1)$. The logistic function measures probability of a binary event, thus it is the main transfer function in the output layer whenever we have a binary classifier network. In case of multiclass classification, the logistic function can be used in an "all vs one" scheme [37], or alternatively the softmax function could be used, which is the implementation of the logistic function for multiclass problems [5].

3.3.1.2 Hyperbolic tangent function

Similar to the logistic function, the hyperbolic tangent function (tanh) is also a sigmoid function, with a different output range:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.7}$$

$$f'(x) = 1 - f(x)^2 \tag{3.8}$$

The range of the output is $[-1, 1]$. Tanh is used both as a classifier and as a regression/signal propagation function. However, it suffers from a vanishing gradient, due to its output restriction. Thus, except from special cases, some variation of the next transfer function is preferred.

3.3.1.3 Rectified linear unit

Rectified linear units (ReLU's) we considered to be a weak choice in the early stages of DL. Researchers believed that it was too simplistic, and also that the loss of information on negative outputs would be detrimental. Nowadays, due to our better understanding of DNN's, and due to the important work of Geoffrey Hinton [34], ReLU is the most popular and useful non-classifying transfer function in DL. The equations are presented bellow:

$$f(x) = max(0, x) \tag{3.9}$$

$$f'(x) = \begin{cases} 1, & x \geq 0 \\ 0, & otherwise \end{cases} \tag{3.10}$$

ReLU does not suppress any positive output. Because of its simplicity it is very computationally efficient. It has also the very important trait that it introduced non-linearity in DNN's. Of course the blow-up problem, which might lead to network overfitting, is existent. However, with certain techniques that will be discussed bellow, this can be greatly reduced. In case that it is advised to have a non zero derivative for negative outputs, one could use either a Leaky ReLU [22], or an Exponential Linear Unit (ELU) [12].

This was not an exhaustive presentation of all the transfer functions that are being used. However, these are the most common ones and more likely for the reader to encounter.

3.3.2 Supervised Training

The most important aspect of DL models is their ability to adapt their configuration according to the desired output. This is known as supervised learning, which is the only type of learning that we are going to explore in this section. However, the principles of supervised learning are used in most techniques of all learning paradigms (unsupervised, reinforcement and semi-supervised learning), as well as in the current application.

3.3.2.1 Backpropagation algorithm

As we showed before, the initial training algorithm that was used was the Delta rule. However its functionality is limited to a single layer network. The great breakthrough that the backpropagation algorithm offered was that we could now change the weight of each connection, according to their contribution to the final error. For our example, consider a Summed Square Error (SSE) function:

$$E(y, \widetilde{y}) = \frac{1}{2}\sum_{j=1}^{m}(y - \widetilde{y}) \tag{3.11}$$

where E is the chosen loss function, (SSE in this example), m is the number of samples and y and $\widetilde{y}$ are the expected and predicted output respectively. Note that this type of evaluating the loss using all the samples every epoch is called batch or on-line training. By backtracking the network, we examine how the previews outputs influence the error using the chain rule. Assuming that we have k hidden layers, the for any neuron we calculate the following:

$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial E}{\partial \widetilde{y}} * \frac{\partial \widetilde{y}}{\partial net_{ik}} * \frac{\partial net_{ik}}{\partial w_{ik}} \tag{3.12}$$

where net_{ik} is the value entering the activation function as in Eq. 3.1. The index ik means the i_{th} neuron of the k_{th} hidden layer. By examining the individual derivatives, it is easy to calculate all of them from left to right. By repeating the chain rule process until we reach the first hidden layer, we then have all the quantities by which the individual weights have to change, according to Eq 3.3. This is the vanilla version of backpropagation and there have been proposed more advanced, computationally efficient and parallelizable variations. However, this is the core of how modern DL models are trained.

3.3.2.2 Validating the network

The purpose of ML is to give information about unobserved data, after appropriate training using observed data. In order to achieve that, it is a common practice to split the dataset in the following parts:

- **Training set:** The training set consists of all the observations needed to fit the weights of the network via training.

- **Cross validation set (CV):** In order to have a reality check during training on the progress of our training, we separate a portion of the data, in order to use it for CV. In prefixed intervals during training, we feed CV data into the network and monitor the CV-error. In case the CV-error starts increasing, regardless of the training error, it is advised to stop the training procedure, since our model starts to overfit (more on that in the next subsection).

- **Test set:** After the training has been successfully completed, we need to check the network's efficiency with new observations. These observations are within the test set, but consequently, if the DNN is put to use, every new observation can be considered to be in the test set.

There are various recommended data split schemes. For average observations number, a 60, 20, 20% is common, although these ratios must be chosen individually for each application.

3.3.2.3 Bias-Variance trade-off

In supervised learning the task is to create a capable with enough flexibility, so that it can capture the spatial complexity of the data distribution. However, there is a very common pitfall. If the model is very complex, and manages to fit a line that passes from all the training observations, then we have the problem of **high variance** (overfitting). An overfitted model has fitted perfectly all the training samples, including noise and outliers, thus failing to generalize and provide accurate predictions in observations outside the training dataset. Herein, the slightest perturbation can result in vastly different predictions. The exact opposite occurs when the complexity not high enough. In that case the model suffers from **high bias** (underfitting). The model has such low complexity that, no matter how many observations we feed it, it cannot adapt to the distribution of the dataset. Thus, whatever observation we feed the network, we will always get similar results. This is best known as the Bias-Variance Trade-Off [24], and finding a balance between those two is the daunting task of most ML applications (Fig 3.4).

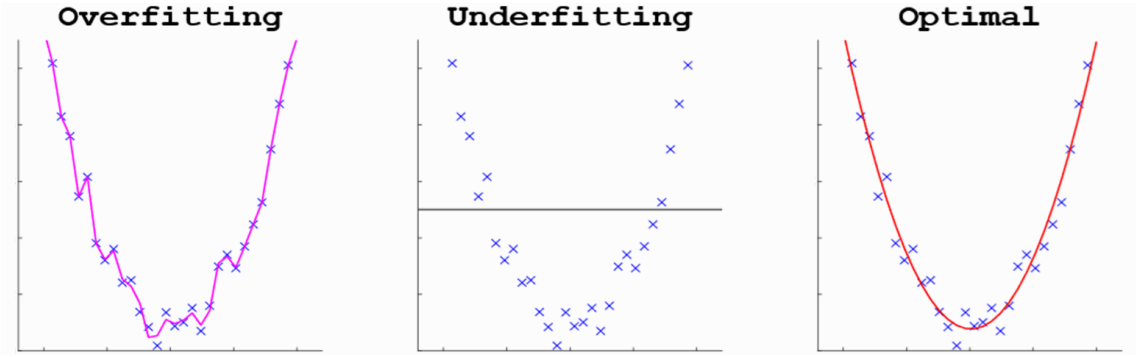


Figure 3.4: A 2D example of a models of different complexity. *Left*: *20 degrees*. *Middle*: *Linear*. *Right*: *Quadratic*

By nature DNN's are high-complexity models. Herein the overfitting problem is

something that must be addressed. Fortunately, we live in a era which is data-abundant, where vast amounts of data are available to solve ML tasks. Thus the high complexity trap of DNN's is usually compensated by large training sets, since it is shown that more data can resolve high variance problems and help the network generalize better [16]. However, overfitting can still be an issue, which can be tackled with some of the following methods:

- **Regularization:** Regularization in data science is the process of using additional information in order to solve an ill-posed problem or reduce overfitting [7]. In ML it usually refers to methods that restrain that sudden weight changes. This happens be adding a penalization term in the loss function. Two of the most common regularizers are, **L1** (Lasso) and **L2** (Ridge regression):

$$J_{L1}(y, \widetilde{y}, \mathbf{w}) = E(y, \widetilde{y}) + \lambda ||\mathbf{w}||_1 \tag{3.13}$$

$$J_{L2}(y, \widetilde{y}, \mathbf{w}) = E(y, \widetilde{y}) + \lambda ||\mathbf{w}||_2 \tag{3.14}$$

  where J is the final cost function and the positive hyperparameter λ is the penalty factor or regularization rate. It is apparent that the weight alteration increases with λ, making the network less flexible, and having $\lambda = 0$ deactivated the regularizer. Common values are between 0 and 1, with 1 being considered as highly restrictive. Of course tweaking it is a matter of design. Regularizers achieve greatly reduced variance without significantly increase in bias [4]. It is the designer's choice to add the bias term in the regularizer, although in most cases it is unnecessary.

- **Dropout:** A very brute way of reducing overfitting is dropout. Dropout is the random pruning of network nodes (excluding the input and output), in order to avoid favouring some features over others. By randomly deactivating nodes in each iteration, the parameter matrix is more evenly distributed. This method is rather counter intuitive, and it should be used with caution, since it mutates the structure of the network, and it might remove important information (Fig 3.5).

- **Early Stopping:** The direct approach is to monitor out network during training using an evaluation metric (e.g. accuracy or loss), and decide when it is an appropriate instance to terminate our training. Usually both CV and training will be improving in the initial steps of training, and there will probably be an epoch when the CV metric will start degrading while the training one will continue improving. This is usually a typical indicator that the model starts overfitting, and probably a good instance to terminate the training procedure.

- **Data augmentation:** Finally, as we discussed before, complex models are able to generalize better when they are fed with more data. However, increasing the size of our dataset is not always possible, and even if it were it could be costly. Thus, we can use data augmentation techniques in order to generate more data by distorting slightly some of the features of existing samples. This is a great technique and is very popular in image classifier networks.

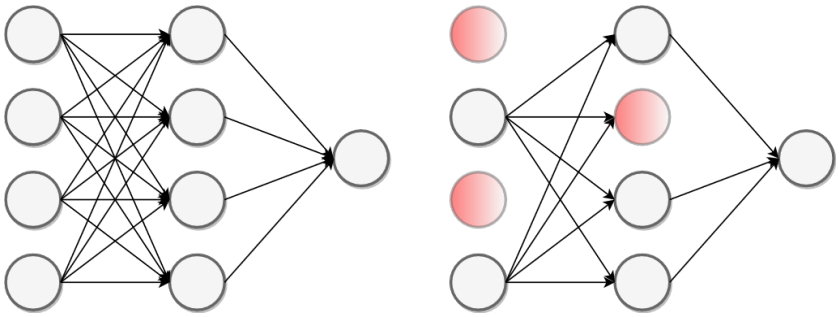


Figure 3.5: Dropout visual example

3.4 Deep Autoencoders

A very interesting type of DNN, is the Deep Autoencoder (DAE). Although autoencoders do not fall under the supervised learning category (some characterize them as unsupervised and others as semi-supervised networks), their training utilizes the same principles as supervised models (Fig. 3.6). Main purpose of autoencoders is to solve the task of representation learning. They reduce the input dimensions and keep the essential structural information of the data, similar to Principal Component Analysis (PCA). In fact, when the decoder is linear and our loss function is the mean squared error (MSE), an undercomplete autoencoder (latent space smaller than the input space) reduces the dimensionality of the input exactly the same way as PCA [20].

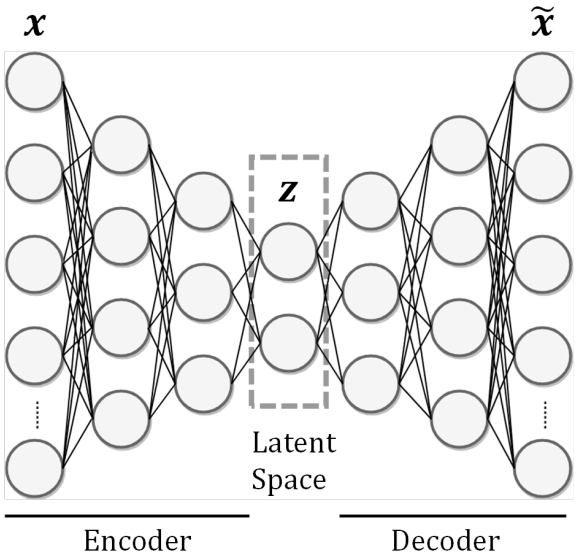


Figure 3.6: A typical DAE architecture

The unique property of DEA's compared to conventional artificial neural networks is that the network architecture consists of two parts: the **encoder** and the **decoder**. The encoder consists of sublayers of decreasing node number, "squeezing" the initial input x into the smallest sublayer, called bottleneck or latent-space layer. The

number of neurons in the bottleneck layer correspond to the number of features we want to compress the data into. The decoder is the part of the network from the bottleneck till the output. The topology of the decoder sublayers is a mirror of the encoder.The bottleneck architecture inhibits the non-useful mapping $g(f(x)) = x$. The efficiency of the network is measured by the reconstruction error produced by feeding unseen data of the dataset. The purpose of autoencoders is to map the input data to itself, i.e. the cost function becomes:

$$J(x, \widetilde{x}) \equiv J(x, g(f(x))) \tag{3.15}$$

where f is the encoding function and g is the decoding function. Notice the parameters of the loss function. For DAE's, the loss is the differnce between the input x and the reconstructed input $\widetilde{x}$, after passing through the encoder part of the network. Thus, for DAE's the loss is also called **reconstruction error**. A very typical choice for the error function is the regularized MSE. DAE's work well when the data is multidimensional, and some of the features are either correlated or non-uniformly significant. Depending on the application, there are several variations of DAE's, such as Denoising, Variational and Sparse Autoencoders. They are a very popular choice for anomaly detection tasks, as we will discuss in the next chapter, thus chosen for the current application [26].

3.5 Summary

In this chapter a brief overview on DNN's was presented. It was explained how biological neurons and their artificial counterpart work, while making a historical passing from the earliest applications to the most recent ones. This chapter was pivotal in understanding the application in this thesis, especially the last part explaining the function of autoencoders. Supervised learning using DL models is probably the hottest topic in the past decade, and is still a very active topic of research. For the interested, the main supervised learning DL architectures are convolutional neural networks (CNN's), recurrent neural networks (RNN's) and adversarial generative networks (GAN's). All of the above have evolved, and one can find more advanced applications, and even a combination of these which is not uncommon [43, 36]. Very promising, but yet not fully implemented are capsule networks (CapsNets) [41] and Neural Ordinary Differential Equations [11].

4

Methodology

Having laid out the foundations in the previews chapters, we are now able to elaborate on the implementation details. The main concept of the current thesis is to create an anomaly detection system, which will almost constantly give feedback on the concrete's beam condition. In this chapter all the implementation steps, as well as the motivation behind them will be presented.

4.1 Anomaly detection

Anomaly detection (also outlier detection) refers to a family of techniques that systematically monitor a system in order to identify rare items, events or observations which raise suspicions by differing significantly from the majority of the data [50]. These unusual observations quite often indicate either that our system has reached an anomalous condition or that the newest observation is defective, both of which should be examined with care. The main assumption utilized by anomaly detection algorithms is that the features of our data are generated by an underlying distribution $p(x)$.

4.1.1 Anomaly detection with DAE's

The use of DAE's has become a staple for anomaly detection tasks. With proper implementation, it is a robust and reliable model. The algorithm steps are the following:

Step-1 Examine the data and use an appropriate preprocessing scheme.

Step-2 Decide which data fall under the normal category.

Step-3 Train the autoencoder with the normal data only, until convergence.

Step-4 Feed all the training data into the model (training+CV), in order to obtain the training reconstruction error.

Step-5 Construct a threshold error according to some statistic derived from the training reconstruction error (a common choice is $T_{rec} = max_{x \in X_{train}} J(x, \widetilde{x})$) [2]

Step-6 Each sample that has a greater reconstruction error than the threshold, is characterized as an anomaly. With the total reconstruction error profile, we can have a sanity check on the efficiency of our network.

The neural network was built in Python 3.7 using the open source ML library Keras. The network configuration will be presented in the appendix.

4.2 Concrete monitoring using strain profile

The goal of the current implementation is to examine the possibility of using DAE as a monitoring system in concrete structures. More specifically, the aim is to monitor structures using only measurements of reinforcement strains obtained from a distributed optic fiber sensor (Fig 4.1).

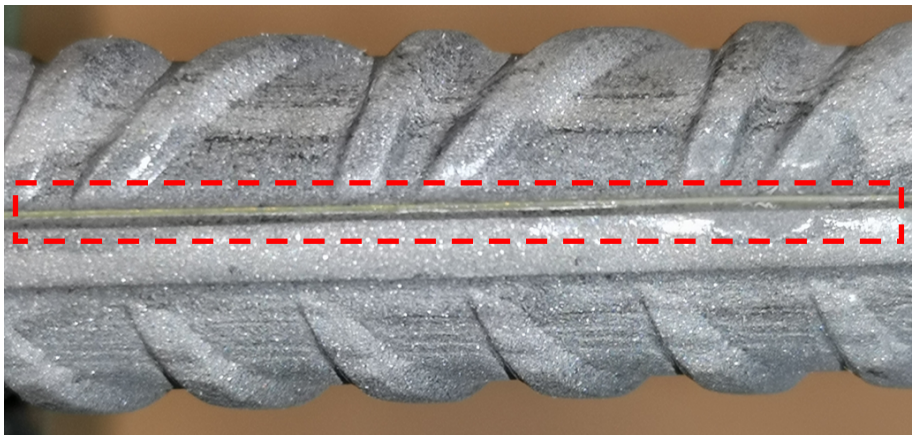


Figure 4.1: Close up view of the optic fiber sensor attached to a steel reinforcing bar used in the experimental tests

The idea is to model the structure using Finite Element Analysis (FEA), load it until failure with different load cases, gather the features needed for the damage estimation and train the model with the normal states of each loading case. This approach, while reasonable, does not account for the unavoidable error existing in all data acquired from integrated sensors (imperfect application, device noise etc.) nor the inherent heterogeneity of concrete. Since concrete is commonly modelled as a homogeneous material in FEA, the existing randomness of the material is not well captured by conventional FE, thereby rendering them divergent from the real structure. To tackle this issue, the initial states of the physical element were included into the training data. This step, which acts as a "calibration", can be easily applied to large scale projects and makes the neural network generalize even better.

4.2.1 Experimental set-up

4.2.1.1 Optic fiber properties

The optic fiber system that was used for the experimental setup was a high-definition (HD) fiber optic strain sensor provided by Luna Innovations Inc. (Fig 4.2). They provide an almost continuous measurement along the length of the measured element up to 50 meters, with a single sensor's capacity surpassing the 1000 strain measurements per meter.

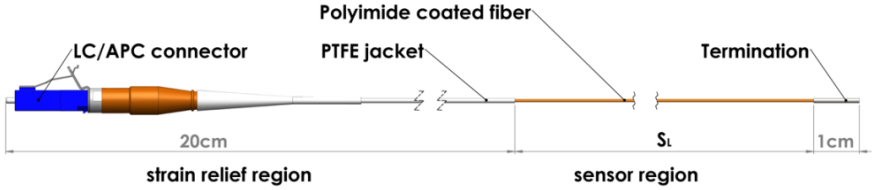


Figure 4.2: A diagram of the sensor configuration

This sensor has the benefit of being embedded without influencing significantly the integrity of the measured element and it remains intact in many hazardous environments. The specifications of the fiber are presented in Table 4.1.

Table 4.1: Specifications of the fiber provided by the manufacturer.

Spatial Resolution (typical):	2 – 20mm
Sensing Range:	30m standard (70m optional)
Strain Resolution:	± 1.0μStrain (@10mm resolution)
Strain Range:	±20,000μStrain
Temperature Resolution:	± 0.1°C (@10mm resolution)
Temperature Range:	-50 to 300°C (practically limited by packaging)
Sampling Rate:	∼0.1Hz
Sensor Capacity:	= length of sensing fiber / resolution (3000 sensing locations typical)
Operating Temperature (Measurement System):	10°C to 35°C

4.2.1.2 Beam properties

The experimental set up consisted of 6 concrete beams with dimensions 90×15×10cm, reinforced with two ⌀ 10mm rebar of B500B steel placed with a concrete cover of 25mm. The concrete had a cube compressive strength of 60 MPa and a tensile splitting strength of 3.5 MPa, both measured at 28 days. For each beam, only one of the longitudinal bars was outfitted with an optic fiber sensor. The signal frequency was 1.25 Hz and spatial resolution was 0.65mm.

4.2.1.3 Load

The beams were tested to failure under three-point loading using a displacement-control setup at displacement rate of 1mm/min. Two of the beams were loaded monotonically and four were subjected to cyclic loading, but all six exhibited shear failure (Fig 4.3).

Figure 4.3: Tested beam after failure

4.2.2 Finite Element Model set-up

As mentioned before, the FEA software used was DIANA Finite Element Analysis (Version 10.2). DIANA is a very user friendly software which is appropriate for quick modeling of concrete structures. Since this is the proof-of-concept project, it was reasonable to use a simple model, that could nonetheless encapsulate the underlying behaviour of the real structure. Moreover, from an engineering perspective, a simpler model is always preferred than a more complex one, since it is more flexible, maintainable and easier to monitor. Thus we used a 2D plane stress model which was meshed into quadrilateral elements. The reinforcement was modeled using 1D truss elements (Fig 4.4).

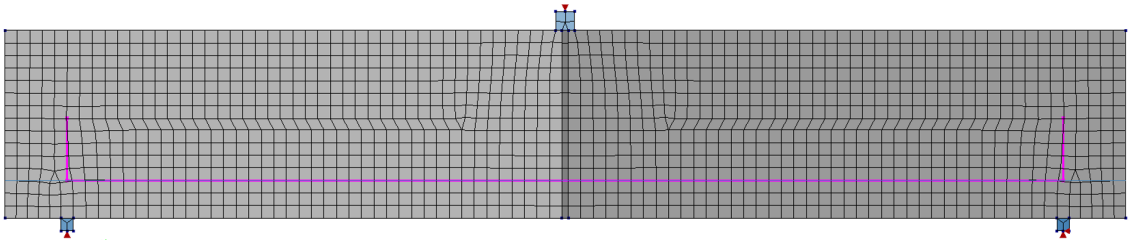


Figure 4.4: The 2D FE model (mesh mode)

The two supports were modeled as sheet elements made from steel, one roller and one pin. Since the purpose of the modeling was to simulate the actual experiment, the loading approach that was used was displacement-based. To achieve that, a loading support was added at the middle top of the beam. The modeling was an adequate approximation of the real experiment, as it is apparent by comparing the force-displacement curves from the FEA and the beam measurements (Fig. 4.5). The properties of the materials were in correspondence to those of the real experiment.

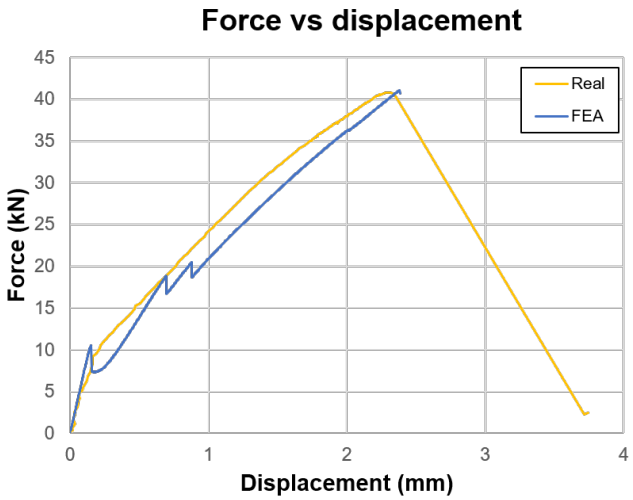


Figure 4.5: Force-displacement curves of FEA and the measured beam (Beam 1)

4.3 Training set up

4.3.1 Pre-filtering of real data

In order to install the sensor, it is necessary to extend the fiber beyond the actual length of the measured element. The reason behind this is to connect it with the acquisition system and assure complete monitoring along the beam. Thus an initial filtering step is to remove the outermost left and right features from each beam (Fig 4.6).

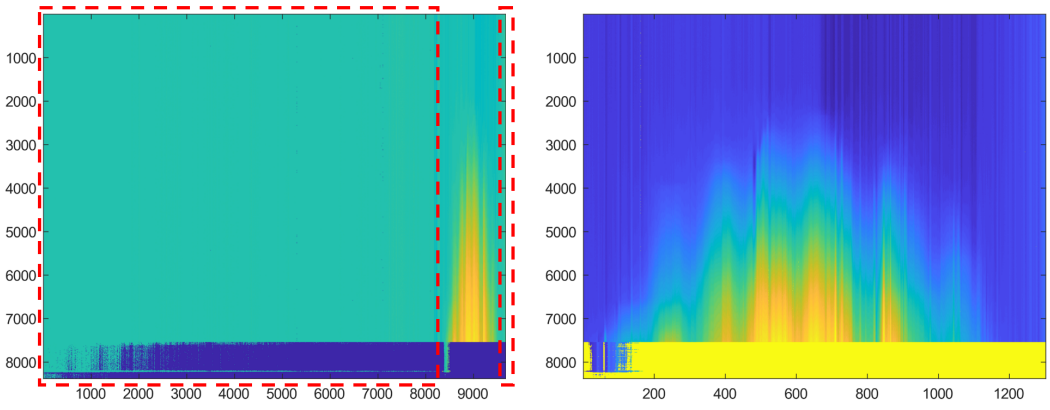


Figure 4.6: First pre-filtering step featurewise (Beam 1)

It is also important to remove the last timesteps (samples) of each beam after failure. Since the beam loses its integrity in an abrupt fashion, the strains get so large that they exceed the measurement range of the fiber. Hence most of the data points are nulls and thus must also be manually removed from the dataset (Fig 4.7).

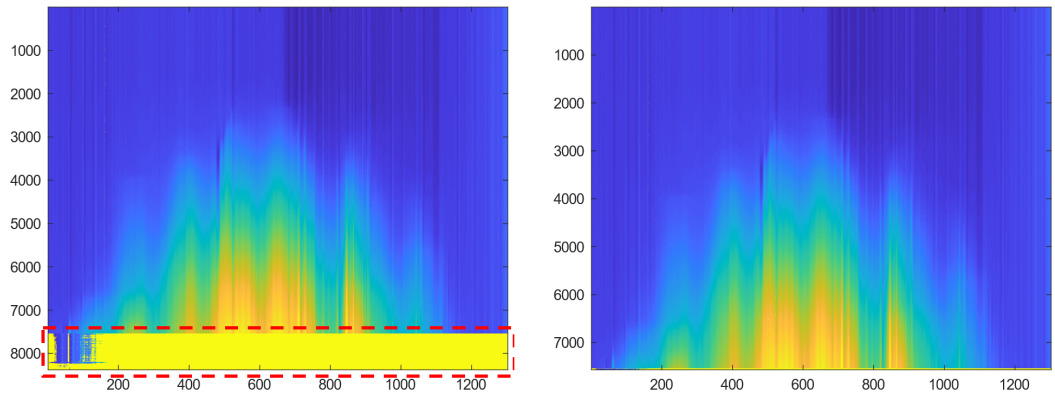


Figure 4.7: Second pre-filtering step samplewise (Beam 1)

4.3.2 Downsampling spatial resolution

As stated before, HD optic fibers measure with an extremely high resolution. Herein, for a 90 cm beam we may end up with a dataset with more than 1200 features. Although there might be applications where this kind of detail might be required, for the current (and most probably in any similar one) project, this abundance of information is unnecessary, and will increase the computational complexity for no concrete reason. Thus, a simple down sampling algorithm was implemented:

```
% Determine downsampling values
D: Original dataset
D_fin : Final dataset
m_start: number of original features
m_final: number of final features
rem = m_start - m_final % m_start
div = int(m_start - m_final / m_start)

% Downsampling algorithm
D = D(1 : end - rem)
Initialize D_fin

for i=1 : length(D)
    initialize helper array temp
    for j=1 : m_final
        a = mean(D(i, div*(j)+1 : i, div*(j+1)))
    end
    Append a in D_fin
end
```

After the FEA, we ended up with 93 final features. The number of features in the real beams were different, depending on the application of the fiber. Thus the above-mentioned algorithm was crucial. One might notice that some of the real features might be unused, but due to the high resolution and the extension of the

fiber beyond the actual length of the beam, it is negligible and does not degrade the coherence of the final dataset (Fig 4.8). If it actually mattered, one could question the robustness of the methodology.

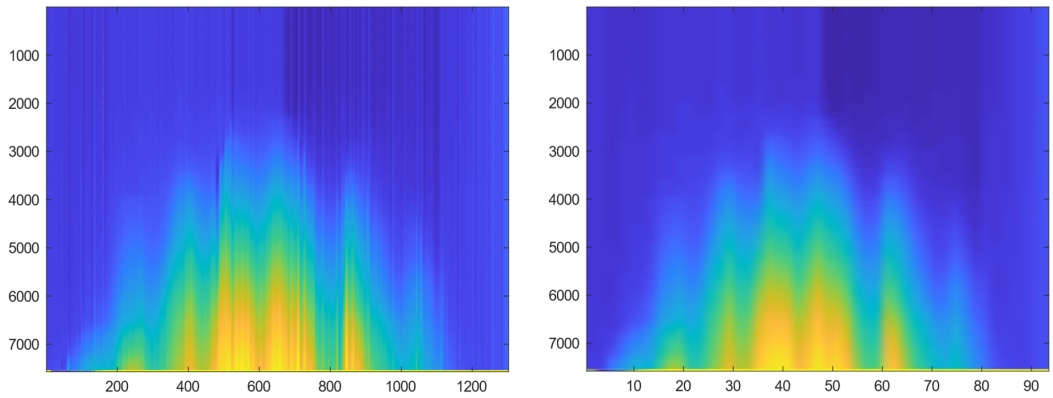


Figure 4.8: Downgrading the spatial resolution samplewise (Beam 1)

Bellow the strain profiles acquired for all the experimental beams are presented, after being pre-filtered by the processing described above:

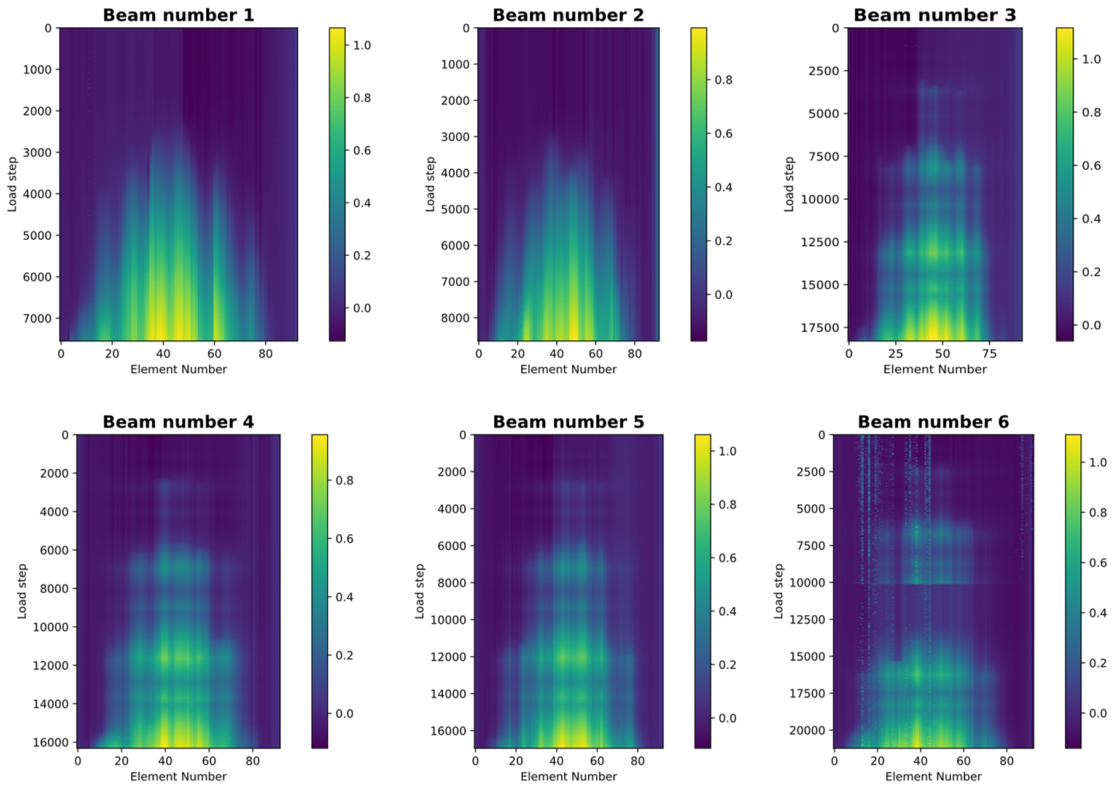


Figure 4.9: Strain profiles for all the real beams

4.3.3 Data usage

The training data consisted of the strain states up to 50% of the total capacity of the results obtained from the FEA, whereas only about the first 35% of the strain

states obtained from each experiment were used for training. In Figure 4.10 we can distinguish the data used from the FEA and the first sample beam.

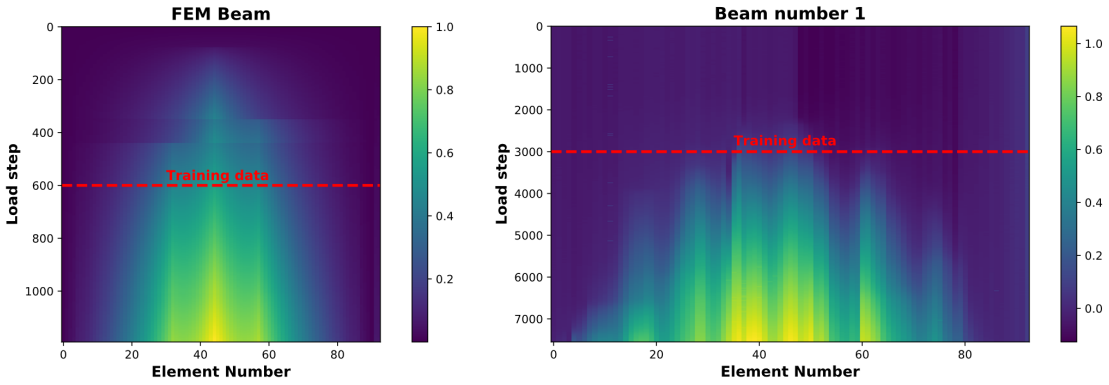


Figure 4.10: On the left, the strain profile of the beam FEA. On the right, the strain profile of one of the tested beams. The red dashed line shows until which state the networks were tested on (Beam 1). It must be noted that even the load did not increase linearly, since the elastic modulus of the beams was changing at the event of cracking

4.3.4 Data preprocessing

The most crucial part of the implementation is the data preprocessing. Feature-wise preprocessing such as z-score standardization and min-max normalization was discarded. The reason is that for the current application all features $F_j, j = 1, \ldots, n$ have the same physical meaning (strain at some rebar position), thus having the same underlying range while the total range of the values will not be known beforehand in a real application. Consequently, the preprocessing scheme that was tested was a row zero mean centering approach where each feature for each datapoint is transformed as follows:

$$x^{ij}_{new} = x^{ij} - \bar{x}^i, i = 1, \ldots, m; j = 1, \ldots, n \tag{4.1}$$

where m is the number of observations and $\bar{x}^i$ is the mean of all the features in the current observation. In common ML applications, this type of preprocessing is discouraged, since transforming different units with the same device removes a great part of relational information, and it is unreasonable to compare features of different nature. Nevertheless, for the reasons discussed above, it is an ideal candidate for the current application.

4.3.5 Damage classification

In bridge condition assessment (BCA) a common practice to assess the level of existing damage in the structure is by creating different criteria or thresholds. Accordingly, three damage thresholds were created by examining the distribution of the strains across the rebar from the analysis: R_0, R_1, R_2 which are the small,

significant and hazardous damage thresholds respectively. Thus, we have:

$$R_0 = max(Err)(1 + \frac{1}{2}\widetilde{\omega}) \tag{4.2}$$

$$R_1 = R_0(1 + \lambda_1\sqrt{\widetilde{\omega}}) \tag{4.3}$$

$$R_2 = R_0(1 + \lambda_2\sqrt{\widetilde{\omega}}) \tag{4.4}$$

where:

$$\widetilde{\omega} = \sqrt{exp(\widetilde{\sigma}^2) - 1} \tag{4.5}$$

$$\widetilde{\sigma} = \sigma(log(Err)) \tag{4.6}$$

with σ being the standard deviation and Err is a vector with all the training reconstruction errors. Intuitively, $\widetilde{\omega}$ is a custom dispersion, fitting the current application. It must be noted that the above rules were constructed empirically, after trial and error, taking into consideration the dispersity of the reconstruction error resulting from measurements of different noise levels. The coefficient λ_1,λ_2 are multipliers that dictate the sensitivity of the system. In our application, $\lambda_1 = 3$ and $\lambda_2 = 5$. In fact, one could construct arbitrary levels of damage λ_i, depending on the significance of the structure. It must be noted that the proposed thresholds should be used with the training scheme that involves both FEA and the initial states data.

4.4 Summary

In this chapter a detailed summary of the proposed methodology was presented. In particular, all the steps were explained, from the data acquisition process to the properties of the conducted experiments. In the next chapter will be presented the results of the current application.

5

Results and Discussion

5.1 Results

As it was mentioned in Section 4.2, incorporating the initial real states in the training procedure acted as a calibration, in order to have a more accurate damage classification. To address that, we examined the two cases. In the following graphs (Fig 5.1-5.3) the reconstruction error for all the experimental data is shown, as well as the maximum training error, with two models trained in a different manner: in the first case the network is trained using only the FEA data and in the second case both the FEA and the initial states of the real data. Apart from the small deviations in the reconstruction error, one must observe the important difference in the maximum training reconstruction error, used to construct the damage classifiers. It is apparent that the first case is more unstable, thus not reliable for robust classification.

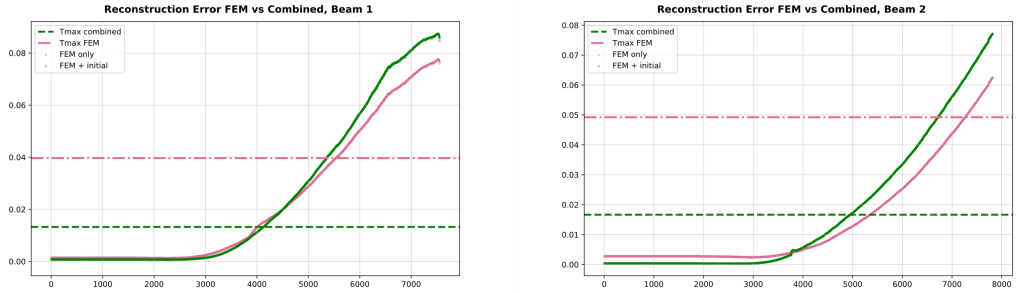


Figure 5.1: Comparing the effect of including the initial states of the monitored element in the training dataset (Beam 1 & 2)

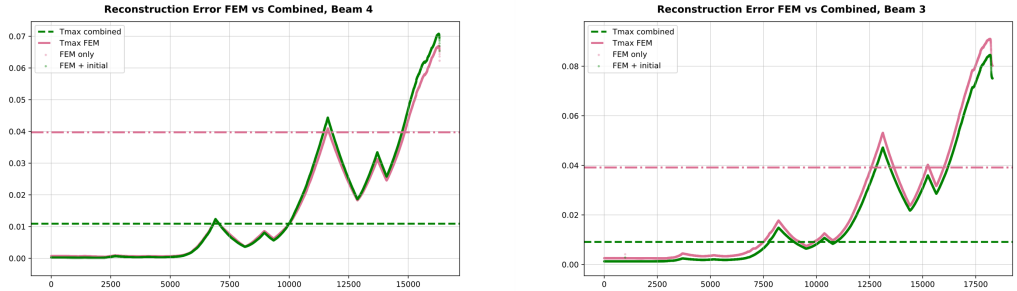


Figure 5.2: Comparing the effect of including the initial states of the monitored element in the training dataset (Beam 3 & 4)

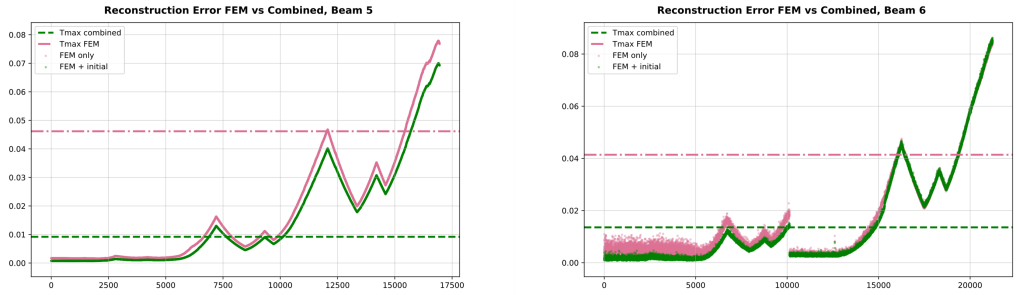


Figure 5.3: Comparing the effect of including the initial states of the monitored element in the training dataset (Beam 5 & 6)

In the following figures are presented the results of the DAE for all the tested beams in terms of reconstruction error and train/validation loss. All the results presented were tested with various random seeds, in order to verify the results. In all beams, the model easily fits the data. This was to be expected, since the features were easily distinguishable due to the loading taking place in the middle of the span.

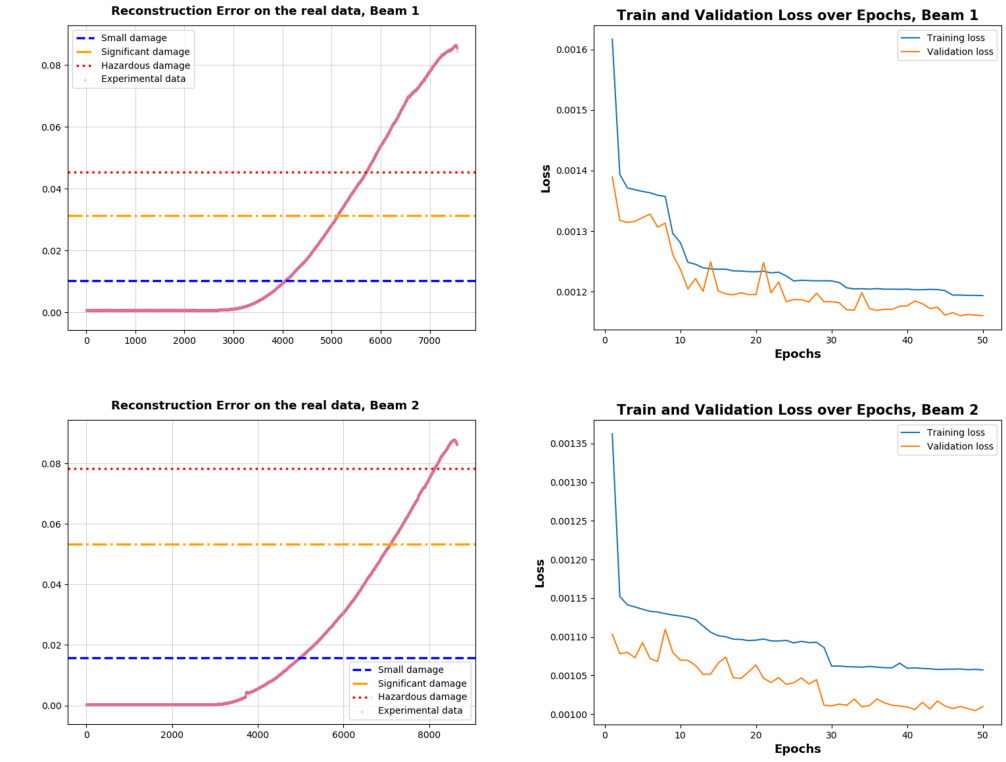


Figure 5.4: The final classification and losses (Beam 1 & 2)

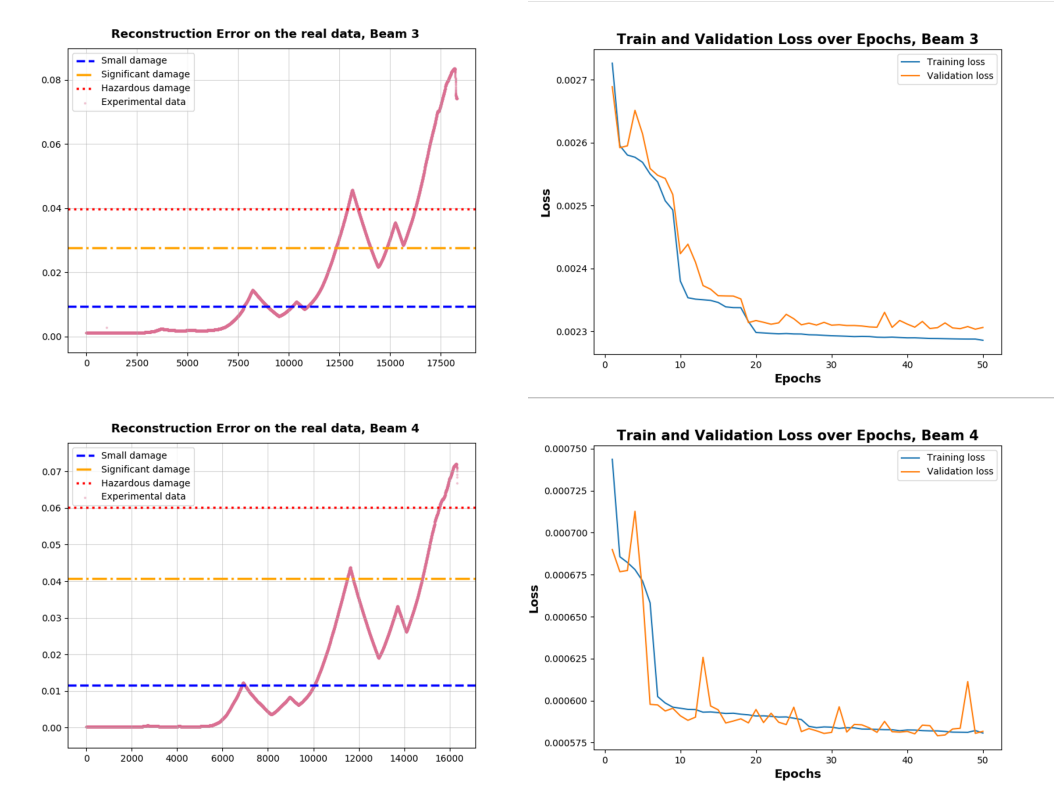


Figure 5.5: The final classification and losses (Beam 3 & 4)

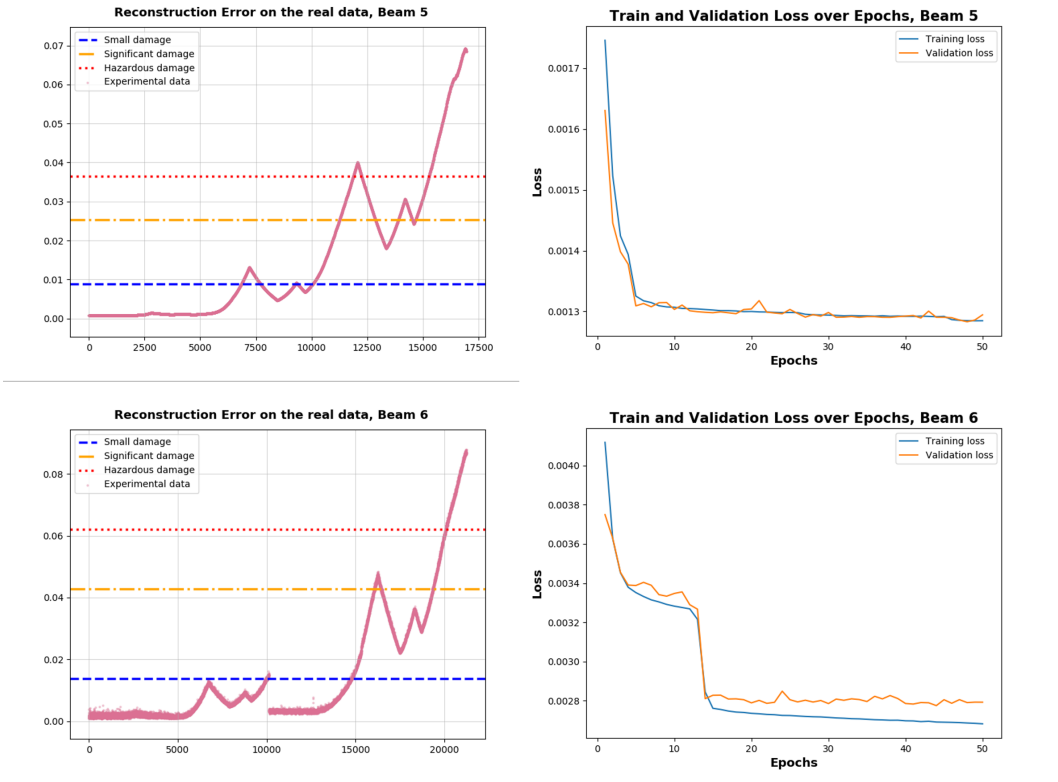


Figure 5.6: The final classification and losses (Beam 5 & 6)

The damage classification had some diversity, depending on both the loading pattern that was applied and also the levels of the signal noise. In particular, from the measured strain profiles, the ones that were considered to be the most problematic were the Beams 2 and 6. However, only Beam 2 detected the Hazardous level a little later than what it should have. For the rest of the beams, the hazardous damage was at approximately 45%-83% of the total beam capacity (Table 5.1). However, in all cases, the significant damage threshold can efficiently trigger an inspection, avoiding the total destruction of the concrete element.

Table 5.1: Results obtained for all beams

	B1	B2	B3	B4	B5	B6
Training loss	0.00123	0.00107	0.00230	0.00058	0.00130	0.00270
CV loss	0.00112	0.00102	0.00234	0.00058	0.00132	0.00280
Small damage	11%	20%	11%	17%	13%	14%
Significant damage	36%	59%	31%	57%	37%	28%
Hazardous damage	50%	**88%**	**45%**	83%	54%	68%

5.2 Discussion

As seen from the results, the detection system that was built was efficient and could detect damage in three different levels. In some cases (Beams 1 and 3) the classification might be more overprotective than what is should be, but that way the designer can be more confident that there will be no critical damage omission in the more noisy examples (Beam 2).

One could possibly argue that the difficulty of the task was rather simple, since point loading at the middle of the span is the simplest case. While this holds true, one must verify the proof-of-concept in the simplest scenarios, and gradually increase the complexity until it becomes appropriate for a real application. Constructing the criteria that will characterize the damage levels was the most important part of the application. It is a very daunting task to compose rules that can have the same behaviour in all real tests. However, with the appropriate preprocessing techniques as discussed in the previous chapter, it was possible to construct a more general rule. Nevertheless, the damage criteria were also dependent on the training reconstruction error statistics, thus stabilizing the training even further is of crucial importance, and must be investigated even further.

To the author's understanding, there are 3 main directions in which this thesis could continue. First, it would be really interesting to examine the possibility of creating a damage classification system using the initial states only. FEA is a really fine procedure, which gets even more sensitive when the structure gets more complicated. By just using only data that is gathered from the sensors, one must be assured that the data that are being used correspond to the normal states of the structures.

Secondly, more ML models could be investigated. Apart from testing different approaches, which is the straightforward action, one could examine the the effect

of using some kind of an ensemble system, where many ML models (the same with different configurations or different ones) are stacked together, and the final result is an outcome of the combined output of the super-system.

Finally, one could test the statistics of the actual errors more rigorously, and construct damage criteria which are more robust and generalizable.

Bibliography

[1] Baltopoulos, A., & Kostopoulos, V. (2015). Chapter 14-Multifunctional carbon nanotube-based nanocomposites for aerospace applications. Multifunctionality of Polymer Composites.

[2] Beggel, L., Pfeiffer, M., & Bischl, B. (2019). Robust Anomaly Detection in Images using Adversarial Autoencoders. *arXiv preprint arXiv:1901.06355*.

[3] Berrocal,C.,G., Fernandez, I., P., Rempling, R., & Logg, A. (2017). SensIT – Sensor driven cloud-based strategy for infrastructure management. *Chalmers University of Technology*, Manuscript in preparation.

[4] Bishop, C. M. (1995, October). Regularization and complexity control in feed-forward networks. In *Proceedings International Conference on Artificial Neural Networks ICANN* (Vol. 95, pp. 141-148).

[5] Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *In Advances in neural information processing systems* (pp. 211-217).

[6] Brownjohn, J. M. (2006). *Structural health monitoring of civil infrastructure. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *365*(1851), 589-622.

[7] Bühlmann, P., & Van De Geer, S. (2011). *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media.

[8] Carder, D. S. (1937). Observed vibrations of bridges. *Bulletin of the Seismological Society of America, 27*(4), 267-303.

[9] Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *32*(5), 361-378.

[10] Chang, H., Lee, C., & Park, S. (2011). Self-monitoring and self-healing bolted joints using shape memory alloy. In *The 28th international symposium on automation and robotics in construction* (pp. 824-825).

[11] Chen, T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems* (pp. 6571-6583).

[12] Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289.*

[13] Dawson, B. (1976). Vibration condition monitoring techniques for rotating machinery. *The shock and vibration digest*, *8*(12), 3.

[14] Department of Biochemistry and Molecular Biophysics Thomas Jessell, Siegelbaum, S., & Hudspeth, A. J. (2000). *Principles of neural science* (Vol. 4, pp. 1227-1246). E. R. Kandel, J. H. Schwartz, & T. M. Jessell (Eds.). New York: McGraw-hill.

[15] Dhakal, D. R., Neupane, K. E. S. H. A. B., Thapa, C. H. I. R. A. Y. U., & Ramanjaneyulu, G. V. (2013). Different techniques of structural health monitoring. *Research and Development (IJCSEIERD)*, 3(2), 55-66.

[16] Domingos, P. M. (2012). A few useful things to know about machine learning. *Commun. acm*, *55*(10), 78-87.

[17] Dorafshan, S., Thomas, R. J., & Maguire, M. (2018). Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Construction and Building Materials*, *186*, 1031-1045.

[18] Dung, C. V. (2019). Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction*, *99*, 52-58.

[19] Farrar, C. R., & Worden, K. (2012). *Structural health monitoring: a machine learning perspective*. John Wiley & Sons.

[20] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT press.

[21] Harvey, D. Y., Flynn, E. B., Taylor, S. G., Farrar, C. R., Ramos Jr, O., & Parker, K. L. (2015). *SHMTools: Structural Health Monitoring Software for Aerospace, Civil, and Mechanical Infrastructure* (No. LA-UR-15-22862). Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[22] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).

[23] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8), 2554-2558.

[24] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112, p. 18). New York: Springer.

[25] Kaku, M. (2012). *Physics of the future: How science will shape human destiny and our daily lives by the year 2100.* Anchor.

[26] Karypidis, D. F., Berrocal, C. G., Rempling, R., Granath, G., & Simonsson, P. (2019). Structural Health Monitoring of RC structures using optic fiber strain measurements: a deep learning approach. *IABSE 2019*, Sep 2019, New York City, Manuscript submitted for publication.

[27] Keoleian, G. A., Kendall, A., Dettling, J. E., Smith, V. M., Chandler, R. F., Lepech, M. D., & Li, V. C. (2005). Life cycle modeling of concrete bridge design: Comparison of engineered cementitious composite link slabs and conventional steel expansion joints. *Journal of infrastructure systems*, *11*(1), 51-60.

[28] Larsen, A., Esdahl, S., Andersen, J. E., & Vejrum, T. (2000). Storebælt suspension bridge–vortex shedding excitation and mitigation by guide vanes. *Journal of Wind Engineering and Industrial Aerodynamics*, *88*(2-3), 283-296.

[29] Lehmhus, D., Brugger, J., Muralt, P., Pané, S., Ergeneman, O., Dubois, M.-A., ... & Busse, M. (2013). When nothing is constant but change: Adaptive and sensorial materials and their impact on product design. *Journal of Intelligent Material Systems and Structures*, *24*(18), 2172–2182. https://doi.org/10.1177/1045389X13502855

[30] Madakam, S., Ramaswamy, R., & Tripathi, S. (2015). Internet of Things (IoT): A literature review. *Journal of Computer and Communications, 3*(05), 164.

[31] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133.

[32] Minsky, M., & Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry.* MIT press.

[33] Mitchell, J.S. (2007) From vibration measurements to condition based maintenance seventy years of continuous progress. *Journal of Sound and Vibration*, *41*(1), 62–75.

[34] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807-814).

[35] OECD (2018), *Infrastructure investment (indicator)*. doi: 10.1787/b06ce3ad-en. Retrieved from https://data.oecd.org/transport/infrastructure-investment.htm

[36] Ouyang, X., Zhang, X., Ma, D., & Agam, G. (2018, August). Generating Image Sequence from Description with LSTM Conditional GAN. In *2018 24th International Conference on Pattern Recognition (ICPR)* (pp. 2456-2461). IEEE.

[37] Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. *Journal of machine learning research*, *5*(Jan), 101-141.

[38] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

[39] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*(3), 1.

[40] Rytter, A. (1993). *Vibrational based inspection of civil engineering structures* (Doctoral dissertation, Dept. of Building Technology and Structural Engineering, Aalborg University).

[41] Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems* (pp. 3856-3866).

[42] Samuel, A. L. (2000). Some studies in machine learning using the game of checkers. *IBM Journal of research and development, 44*(1.2), 206-226.

[43] Xingjian, S. H. I., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems* (pp. 802-810).

[44] Tang, Zhiyi & Chen, Zhicheng & Bao, Yuequan & Li, Hui. (2018). Convolutional neural network-based data anomaly detection method using multiple information for structural health monitoring. *Structural Control and Health Monitoring.* 10.1002/stc.2296.

[45] Vincent, G. S. (1954). Aerodynamic stability of suspension bridges: with special reference to the Tacoma Narrows Bridge.

[46] Wahde, M. (2008). *Biologically inspired optimization methods: an introduction.* WIT press.

[47] Werbos, P. J. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph. D. thesis, Harvard University, Cambridge, MA, 1974.

[48] Wong, K. Y. (2004). Instrumentation and health monitoring of cable-supported bridges. *Structural control and health monitoring*, *11*(2), 91-124.

[49] Worden, K., Farrar, C. R., Manson, G., & Park, G. (2007). The fundamental axioms of structural health monitoring. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *463*(2082), 1639-1664.

[50] Zimek, A., & Schubert, E. (2017). Outlier Detection. *Encyclopedia of Database Systems.* New York: Springer.

A

Appendix

In Appendix the network configuration will be presented. The framework used was Keras, and in particular the *functional API* was used. The layering configurations, as well as the number of nodes is presented in the default Keras format style. The rest will be presented in a normal table

Table A.1: Layer structure.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 93)	0
dense_1 (Dense)	(None, 150)	14100
dense_2 (Dense)	(None, 75)	11325
dense_3 (Dense)	(None, 37)	2812
dense_4 (Dense)	(None, 18)	684
dense_5 (Dense)	(None, 15)	285
dense_6 (Dense)	(None, 15)	240
dense_7 (Dense)	(None, 8)	128
dense_8 (Dense)	(None, 15)	135
dense_9 (Dense)	(None, 15)	240
dense_10 (Dense)	(None, 18)	288
dense_11 (Dense)	(None, 37)	703
dense_12 (Dense)	(None, 75)	2850
dense_13 (Dense)	(None, 150)	11400
dense_14 (Dense)	(None, 93)	14043

Total params: 59,233
Trainable params: 59,233
Non-trainable params: 0

Table A.2: Specifications of the fiber provided by the manufacturer.

Epochs:	50
Train size:	80%
CV size:	24%
Activation function:	ReLU
Regularizer:	$L2 = 1e^{-5}$
Batch size:	20
Optimizer:	RMSprop