POLITECNICO DI MILANO

Master of Science in Civil Engineering

Convolutional Auto-Encoders and Markov Transition Fields for Structural Health Monitoring

Advisors: Prof. Alberto Corigliano

Eng. Matteo Torzoni

Master Thesis of:

Andrea Merlati Enr.n° 838867

Table of Contents

1 Introduction....6
2 Structural Health Monitoring....8
2.1 Anomaly Detection of Time Series Data and Machine Learning....9
3 Artificial Neural Networks....11
3.1 Convolutional Neural Networks....13
3.1.1 Auto-Encoders....15
3.2 Optimization Techniques....17
4 Signal Processing....22
4.1 Discrete Fourier Transform and Lowpass Filtering....22
4.2 Imaging Time Series: Markov Transition Field....24
5 Case Study: The S101 Bridge....28
5.1 Introduction....28
5.1.1 Progressive Damage Description....29
5.1.2 Preliminary Modal Analysis....29
5.2 Proposed Methodology....30
5.3 Single Channel Signals through Individual Convolutional Auto-Encoders....31
5.4 Multi-channel Mosaics through a Single Convolutional Auto-Encoder....36
6 Conclusions....39

List of Figures

Figure 1 - Example of stacking of convolutional layers 14
Figure 2 - Example of a convolutional Auto-Encoder 16
Figure 3 - Different stages of the MTF creation: a) raw signal b) Markov Transition Matrix c) Markov Transition Field 26
Figure 4 - SAX of a trial sinusoid function 27
Figure 5 - S101 Bridge, planar view 28
Figure 6 - a) Cutting of the pier; b) insertion of the steel plates; c) lowering with a hydraulic jack 29
Figure 7 – Discrete Fourier Transform : a) vertical signal ; b) transversal signal 31
Figure 8 - Vertical signal: a) MTF image b) filtered signal 32
Figure 9 - Transversal signal: a) MTF image b) filtered signal 32
Figure 10 - a)MSE ; b)MAPE of the vertical signal 33
Figure 11 - MAE of the vertical signal 33
Figure 12 - Training error distribution of the vertical signal 34
Figure 13 - ROC curves of the vertical signal's damage detection procedure 34
Figure 14 - a)MSE; b)MAPE of the transversal signal 35
Figure 15 - MAE of the transversal signal 35
Figure 16 - ROC curves of the transversal signal’s damage detection procedure 36
Figure 17 - Multichannel representation through MTF of the transversal signals 37
Figure 18 - a)MSE; a)MAPE of the multichannel signal 37
Figure 19 - MAE of the multichannel signal 38
Figure 20 - ROC curves of the multichannel signal's damage detection procedure 38

Abstract

Ever since the advent of computers, more and more problems concerning everyday life have been addressed with some sort of automation technique. The possibility of treating an infinitely larger quantity of data in a way in which the solution generation satisfies proper standards to be considered a good generalizable solution puts Machine Learning Methods in first line, also when it comes to Structural Engineering problems. In particular, Structural Health Monitoring problems cover very challenging ongoing topics within the community, such as Anomaly Detection tasks. Anomaly Detection is the first class of operations in which, rather superficially, monitoring signals are analysed and classified in a specific way in order only to retrieve the presence of a potential damage state that alters the average conformity of the signals. Above all, data pre-processing phases assume crucial importance in this context. To this matter, recent studies show that Signal-to-Image transformations are becoming more and more used techniques to retrieve hidden information, especially when it comes to extract structural states embedded into mono-dimensional acceleration sequences.

In this thesis, the focus will be to introduce the concept of Deep Learning, applied through Neural Networks, in order to detect possible anomalies within specific datasets. To do that, the concept of signal-to-image transformation processes are introduced through the Markov-Chain derived technique, the Markov Transition Field. In order to construct the Neural Network, the Convolutional Autoencoder will be also treated, a very simple, yet powerful tool when it comes to Unsupervised Deep Learning.

All of the above-mentioned methods will be tested accordingly in a final chapter in which the case study of the S101 bridge in Austria will be presented. The bridge was used as a benchmark for future SHM studies since two different damage cases were simulated (lowering a pier and prestressing cables cut through), under the same ambient noise loading condition.

The resulting efficiency of the ANNs will be defined by the use of ROC curves, a very important tool to evaluate the percentage of true and false outcomes, between both positives and negatives. Every computation is taken out by Python codes.

1 INTRODUCTION

During their life span, civil infrastructures are subject to detrimental degradation often caused by natural hazards or ambient conditions that decrease their performance capabilities. In order to avoid lacks in the serviceability limit state and avoid collapse limit states, recent research developments highlight monitoring strategies as the best choice to achieve automated structural damage diagnosis and prognosis, especially to program and enable proper maintenance activities. Vibration-based Structural Health Monitoring (SHM) systems represent powerful tools, as integration to visual inspections, to perform early-stage damage detection in real time with limited intrusiveness. An overview on the field of SHM and the principles of Anomaly Detection is presented in *Chapter 2*.

Artificial Neural Network (ANN) algorithms for Anomaly Detection purposes represent a good choice to tackle the matter. Today's advanced computer technology and data mining knowledge put the ANN data-driven technique as one of the best methods to highlight damaged states. This approach is able to learn the structural state from a sufficient number of relevant features and, hence, to acquire experience for classifying these unknown patterns. Since in most real-world scenarios, data availability is often very limited, a suitable unsupervised learning model that can replicate and generalize these patterns with large accuracy can be crucial to recover this information. In this light, the thesis proposes an unsupervised damage detection technique based on the Convolutional Auto-Encoder (CAE) network, a machine learning tool whose interest is growing in the field of SHM. On overview on ANN, CNN and CAE is presented in *Chapter 3*.

Since the CAE deals well with image processing, the methodology aims at transforming vertical and transversal vibration signals, collected into samples and transformed into images of a fixed dimension with the Markov Transition Field (MTF) algorithm, a very useful tool to capture hidden information of the time series that relies on the probabilities of subsequent observations in the time domain, collected in each sample. To increase the clarity in the images, the time series must be carefully analyzed in the frequency domain and accordingly preprocessed. Pre-processing and signal to image transformation through MTF are discussed in *Chapter 4*.

During the training, the CAE network associated to each signal should learn how to correctly reconstruct the input samples, encoded as images, collected in the healthy period. Then, in order to assess regression's reconstruction capability, the Mean Absolute Error (MAE) is selected as a damage-sensitive feature on the individual samples to evaluate the difference between the original input and the reconstructed output. A damaged sample is individuated if its reconstruction MAE is higher than the 90^{th} percentile of the training MAE distribution. To finally assess if a state is damaged or not, a binary classification problem is created by collecting subsequent samples in 10-minute-long macro-sequences. For a given threshold, the classifier clusters the macro-sequence as damaged (1) or undamaged (0) relying on the percentage of damaged samples contained in the macro-sequence itself. The developed approach is applied to the S101 bridge and presented in *Chapter 5*.

Concluding remarks are collected in *Chapter 6.*

2 STRUCTURAL HEALTH MONITORING

Civil structures are constantly exposed to human-induced and environmental factors which shorten their life cycle[1]–[5]. Damage is commonly encountered in civil infrastructure due to creep, corrosion, shrinkage, fatigue and scour. Hence, it is of utmost importance to continually monitor civil structures to evaluate their structural condition and provide early warning against structural damage.

Structural Health Monitoring (SHM) examines the state of the present condition of structural systems to assess their functional fitness and performance levels for real-time damage evaluation[6], but especially to plan for preventive maintenance, both in terms of shutdown time and economic expenses[7]–[11]. Damage is traditionally defined as a change in the geometric or material characteristics of a system that adversely affects its performance, safety, reliability, and operational life. According to this definition, damage does not always indicate a complete failure of a system, yet a comparative deterioration of the system functionality causing a suboptimal performance. If no remedial action is taken, damage may accumulate until reaching the failure state and structures may fail in a gradual or sudden manner depending on the type of the damage. Damage processes in civil structures usually arise in different timespans, differentiated by cases involving various load and degradation scenarios. The increment during long periods of time is usually associated to fatigue and corrosion, but also rapid drops in structural properties can be caused by unstable crack propagation, particularly involving stiffness, mass and damping. This consequently affects the dynamic response of the system, and so it is convenient to monitor and study scenarios in time windows in which the damage's increment rate is comparable. In civil structures, monitoring durations are typically smaller with respect to the time necessary for the damage to fully propagate and manifest itself through a change of mechanical characteristics; in this case, damage can be considered as frozen in time, and consequently the structure's properties can be represented as linear. Also, modelling damage considering a rigidity drop in each element is acceptable if the damage evolution rate is sufficiently small with respect to the monitoring time [12]. SHM techniques act in different ways, depending on what level of depth the damage scenario must be described. The laboriousness with which a SHM system is chosen, placed and used is proportional to its ability

to capture the entity of damage. To this matter, the most important SHM techniques can be classified in five stages[13], [14]: Anomaly Detection assesses the response to determine if damage is present or not; Damage Localization identifies the location of damage; Damage Quantification, assesses its entity; Damage Prediction follows the future progress and remaining service life of the structure; Damage Intervention recommends the appropriate remedial and repair measures to restore both the structure's strength and functionality.

The techniques to implement one or more SHM stages can be summarized as follows:

Global monitoring methods based on changes of natural frequencies and modal shapes caused by damage [15].

Global monitoring methods based on matrix updating, which act by modifying on stiffness, mass and damping of the undamaged structure to better fit the actual dynamic response of the damaged structure. These methods boil down to an optimization problem of the error between actual and reconstructed response. There are several downsides of these methods, such as matrices of the undamaged structure aren't known a priori, and modelling damping is not an easy task most of the time. Also, the changes in the matrices act indistinctively on truly damaged and undamaged elements, causing the solution not to be unique.

Bayesian methods [16] determine the damage scenario that is most probable and compatible with the monitored data. The damage probability is obtained by the difference in forced or naturally induced vibrations and the ones obtained from the analytical model. The main advantage of these methods is to explicitly differentiate between measure and modelling errors when determining if the structural parameters describe a damaged or undamaged scenario.

Machine Learning methods [17], particularly regarding Artificial Neural Networks [18] are based on learning if a state is damaged or not, provided some evidence of the undamaged condition. The advantage of this method is its versatility on the choice of the type of data to analyze [19]and not knowing the mechanical properties does not compromise the outcome at all.

2.1 Anomaly Detection of Time Series Data and Machine Learning

Anomaly (or Outlier) detection has become a field of interest for many researchers and practitioners and is now one of the main tasks of time series data processing[20], [21]. From a classical point of view, a widely used definition for the concept of anomaly has been provided by Hawkins [22]

"An observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism."

Therefore, anomalies can be thought of as observations that do not follow the expected behavior. Anomalies in time series can have two different meanings, and the semantic distinction between them is mainly based on the interest of the analyst or the particularly considered scenario. Erroneous observations can also be related

to noise or unwanted data, which by themselves are not interesting to the analyst, case in which these should be deleted or corrected to improve the data quality and generate a cleaner dataset that can be used by other data mining algorithms[23].

Anomaly detection techniques on time series data vary depending on the input data type, the anomaly type, and the nature of the method. In particular:

- The data type can be *Univariate*, when $X = \{x_t\}_{t \in T}$ is an ordered set of real-valued observations, where each observation is recorded at a specific time $t \in T \subseteq \mathbb{Z}^+$[paper ultimo], or *Multivariate*, when $X = \{x_t\}_{t \in T}$ is defined as an ordered set of k-dimensional vectors, each of which is recorded at a specific time $t \in T \subseteq \mathbb{Z}^+$ and consists of k real-valued observations $x_t = (x_{1t}, \dots, x_{kt})$

- The anomaly type can diversify between *Point Outliers, Subsequence Outliers* or *Outlier Time Series.* A point outlier behaves unusually in a specific time instant when compared either to the other values in the time series (global outlier) or to its neighboring points (local outlier). Subsequence Outliers refer to consecutive points in time whose joint behavior is unusual, although each individual observation is not necessarily a point outlier. Outlier Time Series involve entire time series that can also be outliers and can only be detected when the input data is of multivariate.

- The nature of the detection method employed (i.e., if the detection method is univariate or multivariate). A univariate detection method only considers a single time-dependent variable, whereas a multivariate detection method can simultaneously work with more than one time-dependent variable. The detection method can be univariate, even if the input data are a multivariate time series, because an individual analysis can be performed on each time dependent variable without considering the dependencies that may exist between the variables. In contrast, a multivariate technique cannot be used if the input data are a univariate time series.

The general architecture of all anomaly based network intrusion detection systems (A-NIDS) methods is similar. All of them consist of the following basic modules or stages. These stages are parameterization, training and detection. Parameterization includes collecting raw data from a monitored environment. The raw data should be representative of the system to be modelled. The training stage seeks to model the system using manual or automatic methods.

Detection compares the system generated in the training stage with the selected parameterized data portion. Applying machine learning techniques for anomaly detection can automatically build the model based on the training data set, which contains data instances that can be described using a set of features and associated labels [18], [24]–[26].

Different machine learning techniques including supervised, unsupervised and semi-supervised, have been proposed to enhance the performance of anomaly detection.

3 ARTIFICIAL NEURAL NETWORKS

Commonly known as Artificial Intelligence, the field of Machine Learning developed in many directions, particularly towards the innovative concept of Artificial Neural Networks [17], [23]. In this regard, Deep Learning is essentially a subset of Machine Learning in which the ANN is composed of more than three layers, including the output and input layers. Machine-learning systems are used to identification, speech transcription speech, item matching and result selection. Conventional machine-learning techniques were limited in their ability to process natural data in their raw form, since they are more dependent on human intervention to learn. Human experts determine a hierarchy of features to understand the differences between data inputs, so, taken image recognition as an example, the human brain decides an aleatory number of features, depending on the image, to assign specific labels to each element. This is known as supervised learning since the process incorporates human intervention or supervision and requires considerable domain expertise. Humans determine a hierarchy of features to understand the differences between data inputs, so, taken image recognition as an example, the human brain decides an aleatory number of features, depending on the image, to assign specific labels to each element. This is known as supervised learning since the process incorporates human intervention or supervision.

On the other hand, Deep Learning algorithms are representation-learning methods and do not necessarily require supervision datasets, and it can ingest unstructured data in its raw form, and it can automatically determine the set of features which distinguish the hypothetical number of labels that are associated to the dataset[27]. For classification tasks, higher levels of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. So, by observing patterns in the dataset, a Deep Learning models can cluster inputs appropriately, discovering hidden information without the need of human intervention, and these are known as unsupervised learning techniques. As the name suggests, the concept of Neural Networks is inspired by the brain. A neuron is essentially a cell that holds a number that can space between minus and plus infinite, called activation. The neurons can be collected in bunches, called layers, that are linked together and organized one by one in a successive manner. The layers always start with an input

layer, collecting the initial data to be used, and finish with the output layer, collecting the data to be finally achieved. Layers between these are called hidden layers. The way activations are stimulated in one layer determines the way the ones collected in the next layer will be, and the heart of the network as an information processing mechanism comes down to exactly how those activations in one layer bring about the ones in the next layer. As mentioned, the reasoning is meant to be loosely analogous to how in biological networks, some groups of neurons firing cause some other groups to fire. So, each integral information can be seen as the summation of many subsets of information that, pieced together in a specific matter, compose a particular output. Each hidden layer has the scope of retrieving, as precisely as possible and on a different scale, depending on its position in the network's architecture, each feature composing the dataset to be able to reconstruct it with accuracy. Adding layers can be seen as breaking down a feature recognition procedure in many other sub-features to be extracted. And so, the question must be set on what dials and knobs should be tweaked so that it's expressive enough to potentially capture each pattern. The procedure in which this is achieved is called *training*, and it is developed on a specific dataset that represents, for sure, the true essence of the problem at hand. During training, weights are assigned to each one of the connections between recurrent layers, so that each layer can extrude a weighted sum to be able to pick up information. Sums will then be larger when the enlightened neurons are triggered. Usually, the weighted sum is processed by an activation function to avoid divergence problems, like a sigmoid, so that very negative inputs end up closer to a lower boundary and positive inputs end up closer to a higher boundary. The activation of the neuron is basically a measure of how positive the relevant weighted sum is. Also, a common class of parameters other than weights are biases, that control inactivity. So, ultimately, the weights tell what feature pattern a neuron is picking up on and the biases tell how high the weighted sum needs to be before the neuron starts getting meaningfully active. To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weights were slightly increased. The weight vector is then adjusted in the opposite direction to the gradient vector. The objective function, usually an error function, can be seen as a high-dimensional plane, and the gradient vector indicates the direction of steepest descent, taking it closer to the global minimum, where the cumulated output error is the lowest. The process is repeated for many small sets of grouped data, called batches, for a set number of times, called epochs, until the average of the objective function stops decreasing. A different dataset, never seen by the training algorithm, but of the same nature as the training dataset, called *validation* set, is used to monitor the good generalization performance of the learning mechanism. After training, the same generalization performance is tested with an analogous dataset, referred as *test* set. The training of any ANN works on the computation of a preliminary defined cost function, which relates the differences between the original and predicted outputs that the network estimates. The sum is small when the network confidently classifies the data correctly, and it's larger when the randomicity of the process prevails. The average cost over the dataset is the measure of how the network is approaching its full capability to reproduce the output. Since the network itself is basically a function, parametrized by all the weights and biases, the cost function adds a degree of

complexity, taking the weights and biases as input and providing an estimate of how well those parameters are been retrieved. In order to minimize the cost function, and so fitting the parameters in the best way possible, the main goal is to derive its gradient to understand the direction of steepest decent. In other words, finding minima of an unknown function often boils down to calculating the differences per each step of a process to move with a negative slope along the function, until a reasonable threshold is met. From a theoretical point of view, depending on which random input the process starts at, there is no guarantee that the local minimum landed in is, in fact, the global minimum. Since the gradient contains information on which direction and which magnitude the change in the parameters is taken, the adjustment on some weights might have a greater impact on the cost function over others, meaning that the gradient, in fact, encodes the relative importance of each weight and bias. The way in which the training phase corrects the weights and biases based on the gradient decent of the cost function must keep track of which adjustments should take place to obtain that specific output, nudging up or down relations to the latter. By focusing on just a neuron, ideally the activation can be modified in two ways: modifying the bias and the weights or also changing the activations from the previous layer. The weights, as said, have different levels of influence: the connections with the most active neurons have the biggest effect, since they are multiplied by larger activation values. Increasing those weights has a stronger influence than others connecting dimmer neurons, so the biggest increases imply stronger connections between the neurons that are most active and the neurons that are supposed to be most active after training. On the other hand, by increasing previous layer activations positively related to the neuron, and decreasing the ones negatively related, the same goal would be achieved. Since the network has no direct influence in nudging activations, this concept is important only to keep a note of what those desired changes are. By adding together all the desired effects of the neurons of one layer to the neurons of the layer before, a list of nudges is formed on the weights linking those two layers. This intuitive way of thinking takes the name of *Back Propagation.* In practice, it takes computers an extremely long time to add up all the influences of every weight at every single gradient decent step, so what is commonly done to improve the computational burden is to shuffle the data and organize it in groups, called batches. An individual gradient at each step is computed according to the batch, then a summation through the batches is performed at each epoch, until cost convergence is achieved. The batch size and the number of epochs influence both convergence accuracy and speed. The error improvement is controlled by the *learning rate*: if this is too small, then the algorithm will need more iterations to converge and, vice-versa, a greater value could force the training to never reach the required minimum.

3.1 CONVOLUTIONAL NEURAL NETWORKS

The concept of Convolutional Neural Network (CNN) was inspired by the structure of the visual cortex of living organisms and, so, programmed to locate distinctive patterns in input data. They are very successful today in the field of video classification, image recognition and voice identification. Research, particularly with David H. Hubel e Torsten Wiesel [28] shows how higher-level neurons are partially

linked with lower-level neurons, located in their receptive field. A CNN tries to reproduce this concept, alternating two types of layers, not common to other ANN architectures: convolutional layers and pooling layers. All the characteristics of 2D CNN are descripted in the following.

In a convolutional layer, neurons proper of a layer aren't all connected to all the neurons of the layer before, but only to the ones in their receptive field. Each convolutional layer is composed by a series of *kernels*, or filters, that are a series of small matrices that represent the features that the network will want to extract. Given an input matrix, arbitrarily big, the operation that is obtained by sliding the kernel over the input, multiplying adjacent values and summing them up is called *convolution*. The functional form of a convolutional layer is given by:

$$f_{conv}(h; k) = \sigma((h * k))$$

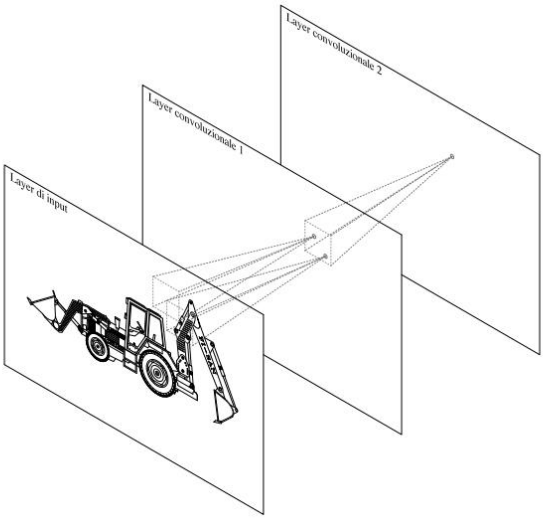

Figure 1 - Example of stacking of convolutional layers

In 2D problems, at a sliding step centered at (i_x, i_y) the 2D convolutional dot product of a kernel $k \in \mathbb{R}^{(2w_x+1)x(2w_y+1)}$ on an input x is given by:

$$(x * k)_{i_x,i_y} \triangleq \sum_{p=w_x}^{-w_x} \sum_{q=w_y}^{-w_y} x_{i_x-p} k_p$$

A convolution operation can be valid, which ends up decreasing the dimension of the output with respect to the input, achieved by inserting a stride between adjacent receptive fields, or full, when the size output becomes larger than the input. The same dimension can be held by choosing to use *zero padding*, in which the input is simply surrounded by a series of zeros. Every layer is forced to the same filters, drastically reducing the number of parameters with respect to fully connected dense architectures. The filtering result will provide a mapping

of the characteristics that highlight the parts of the image that are similar to the convolution kernel itself, with the latter that is learned during training. Every image in input can be seen as n-dimensional matrices, with n equal to the color channels (the basic is 3). This is to say that color channels are equivalent to the characteristic maps of a convolutional layer.

A pooling layer reduces the dimension of the input, and consequently reduces the computational burden of the problem by limiting the number of parameters, limiting the risk of overfitting. The same small neuron triggering field is applies to pooling layers as for convolutional layer. Nevertheless, pooling layers do not have kernels (so they don't add parameters to the problem), but only squish the outputs of the previous layer, usually sliding a window of pre-defined length, calculating the average of its content or taking the maximum value. The typical CNN architecture is built by alternating convolutional and pooling layers; this consequently reduces the size of the images and increases their "depth" (i.e. the number of characteristic maps). The last pooling layer can sometimes be attached to a series of fully connected dense layers.

The downside of convolutional networks lies mainly in the great number of hyperparameters that must be accordingly set by the user. Hyperparameters are the values that control the learning mechanism, they usually remain constant during training, but could also be scaled with scheduling algorithm. Usually, the hyperparameters are: the number of layers, the number of filters, the dimension of filters, the spacing, the padding, the learning rate, the number of epochs and the mini batch size.

As a summary, the success of CNNs can mainly be attributed to the following advantages [27]:

- CNNs fuse the feature extraction classification and feature extraction stages into a single learning body, learning to optimize the features during the training phase directly from the raw input.
- Since CNN neurons are sparsely connected with tied weights, CNNs can process large inputs with a great computational efficiency compared to the conventional fully connected Multi-Layer Perceptrons (MLP) networks.
- CNNs are immune to small transformations in the input data including translation, scaling, skewing, and distortion.
- CNNs can adapt to different input sizes.

3.1.1 Convolutional Auto-Encoders

Among the unsupervised learning methods, the Auto-Encoder is the only architecture used in vibration-based applications. The efficiency of the Auto-Encoders against conventional ANNs is due to the fact that the learned features describe the original data in a much better way, which makes them an excellent choice for both classification and regression problems.

Auto-Encoders learn to reconstruct the input observations x_i with the lowest error possible [29]. The Convolutional Auto-Encoders' main components are three: an Encoder, a Latent Feature Representation, and a Decoder (Figure 2) that can be simply seen as functions. The Auto-Encoder must reconstruct the input well

enough and, at the same time, it should create a latent representation that is useful and meaningful. This latent representation (how to reconstruct the input) can then be very useful for various tasks (for instance feature extraction that can then be used for Anomaly Detection) or simply understanding the essential features of a dataset.

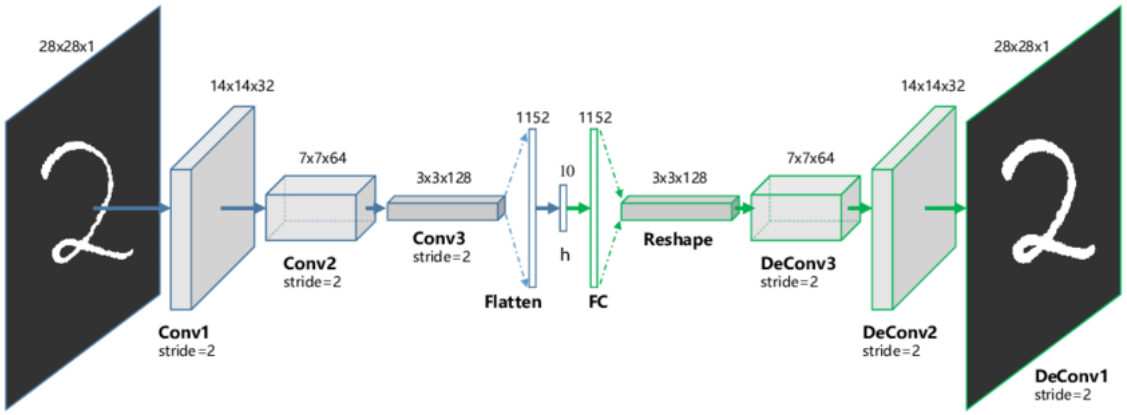

Figure 2 - Example of a convolutional Auto-Encoder

In general, the Encoder can be written as a function g that will depend on some parameters

$$h_i = g(x_i)$$

where $h_i \in \mathbb{R}^q$ (the Latent Feature Representation) is the output of the Encoder block when we evaluate it on the input x_i . Intuitively:

$$g : \mathbb{R}^n \to \mathbb{R}^q$$

The Decoder (and the output of the network that will be identified with $\tilde{x_i}$) can be written then as a second generic function f of the latent features:

$$\tilde{x_i} = f(h_i) = f\big(g(x_i)\big)$$

Where $\tilde{x_i} \in R^n$. Training an autoencoder simply means finding the functions features g and features$f$ that satisfy:

$$\arg\min_{f,g} < [\Delta\big(x_i\,, f\big(g(x_i)\big)\big] >$$

where Δ identifies a loss function and $< \cdot >$ indicates the average over all observations. Using Auto-Encoders is a very viable technique to reduce the training time by several orders of magnitude while incurring in a minor drop in accuracy.

3.2 OPTIMIZATION TECHNIQUES

It has been widely known that the deeper a neural network is, the harder it is to train [30]. The intriguing ability of gradient-based optimization is perhaps one of the major contributors to the success of deep learning. Since a gradient-based method is either a first- or a second-order method, once converged, the optimizer is either a local minimum or a saddle point. The existence of local minima poses a serious problem in training neural networks [31]. Usually, the gradient method can find a global minimum, can escape saddle points and can avoid spurious local minima. However, these assumptions do not always hold. This limits the understanding on what contributes to the success of the deep neural networks, causing sometimes the impossibility to meet theoretical conditions in practice. To this matter, the optimization process plays a critical role in training and has a significant effect on the trained result.

Activation and Cost Functions

Activation functions are needed in ANNs to scale the outputs of each layer in a non-linear manner. If an activation function is not used in a neural network, then the output signal would simply be a simple linear function which is just a polynomial of degree one. Although a linear equation is simple and easy to solve but their complexity is limited, and they do not have the ability to learn and recognize complex mappings from data. A Neural Network without a series of activation functions acts as a Linear Regression. By adding nonlinearity with the help of non-linear activation functions to the network, the latter will be able to achieve multiple mappings from inputs to outputs.

The types of activation functions concerning the regression problem in this work (input reconstruction) are Re-Lu and Sigmoid.

Re-Lu stands for rectified liner unit and is a non-linear activation function which is widely used in neural networks. It can be defined mathematically as:

$$f(x) = \max(0, x)$$

The upper hand of using Re-Lu function is that only positive neurons will be activated, and this comes handy when the input is composed only by positive values.

The Sigmoid function is the most widely used activation function as it transforms the values in the range 0 to 1. This is useful in multi-classification problems or when applied to output layers in Auto-Encoders when the input is composed of values between 0 and 1. It can be defined as:

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

Cost functions are the basis on which the gradient decent problems acts. In regression problems, very common can be the Mean Squared Error, the Mean Absolute Value or the Mean Absolute Percentage Error.

Adam Optimizer

Momentum learning [32] consists in adding a term to the gradient decent algorithm such that previous updates can be considered. The main concept is to try and accelerate the convergence by dampening oscillations using gathered "velocity", so not moving in the opposite direction of the gradient but also moving in the "averaged" direction of the last updates. Momentum learning is also very useful to escape local minima traps. The basic way to represent a momentum term could be:

$$\Delta w_{i,j}(t) = \alpha\, \Delta w_{i,j}(t-1) + \mu\, \frac{\partial L}{\partial w_{i,j}(t)}$$

With $\Delta w_{i,j}(t)$ referred as velocity term, α is a damping parameter, $\frac{\partial L}{\partial w_{i,j}}$ is the term of the gradient depending on the weight, μ is the learning rate.

Adaptive learning essentially about accelerating or decelerating the learning rate at the right moment, depending on if the algorithm is moving respectfully if the gradient decay stays consistent or if it is changing its direction. Conceptually, there are two main steps: the first consists in initializing some sort of local gain (g) for each weight parameter, initialized through $g = 1$, so:

$$\Delta w_{i,j} = \mu\, g_{i,j} \frac{\partial L}{\partial w_{i,j}}$$

$g_{i,j}$ is the gain associated to that weight. The second step is to modify the gain, increasing it if its consistent or decreasing it if it's not:

$$g_{i,j}(t) = g_{i,j}(t-1) + \beta \quad \text{if consistent}$$

$$g_{i,j}(t) = g_{i,j}(t-1)(1-\beta) \quad \text{if not consistent}$$

Where t is the iteration step. Note that multiplying by a factor has a larger impact if gains are large, compared to adding a term, so the dampening effects if the updates oscillate in the wrong direction are larger than the increases in the right direction. Examples of very popular adaptive learning rates can be seen in RMSProp, which identifies the gain with an exponentially decreasing moving average of the squared components of the gradient, as shown:

$$MSQ(w_{i,j}, t) = \beta\, Err(w_{i,j}, t-1) + (1-\beta)\left(\frac{\partial L}{\partial w_{i,j}(t)}\right)^2$$

The mean squared value is used to scale the gradient component in the gradient decent update:

$$w_{i,j}(t) = w_{i,j}(t) - \mu \frac{\left(\frac{\partial L}{\partial w_{i,j}(t)}\right)}{\sqrt{MSQ\left(w_{i,j}, t\right)} + \epsilon}$$

Where ϵ is a term inserted to avoid by 0 division errors. The reasoning behind momentum and RMSProp are directly related to the Adam (Adaptive Moment Estimation) algorithm, probably the most widely used optimization algorithm today [33]. In fact, Adam is essentially a combination of RMSProp and momentum learning. The momentum and the RMSProp terms (respectfully m_t and rms_t) can be exploited for the Adam algorithm as:

$$m_t = \alpha\, m_{t-1} + (1 - \alpha)\, \frac{\partial L}{\partial w_{i,j}(t)}$$

$$rms_t = \beta\, Err\left(w_{i,j}, t-1\right) + (1 - \beta)\left(\frac{\partial L}{\partial w_{i,j}(t)}\right)^2$$

And combined to obtain the Adam weight update:

$$w_{i,j}(t) = w_{i,j}(t) - \mu \frac{m_t}{\sqrt{rms_t} + \epsilon}$$

In many papers in literature, α and β are referred as β_1 and β_2. This to show how Adam is capable to adapt the learning rate during training, all done avoiding calibration and converging much faster to a better solution.

Kernel Regularization

Regression techniques are usually very accurate in reconstructing the approximation of a function using a series of polynomials. Given a series of observations in an n-dimensional space, the approximation that best fits the data can be identified as the function that minimizes the cumulated error between the observations and the function itself. The outcome of a regression analysis, depending on the type of function used and its power in capturing features, can lead to three different scenarios:

Underfitting: when the chosen power of the polynomials is not enough to capture the main features in order to create a well fitted approximation, the outcome will be a function that leads to a not acceptable accumulated error.

Balanced fit: when the chosen power of the polynomials captures very well the observation pattern, leading to a well fitted accumulated error.

Overfitting: when the chosen power of the polynomials is very high, a possible outcome could be a drop in the generalizing ability of the algorithm, so the approximation function minimizes the error on the training set, but will increase the error on any other set.

Overfitting is a very common issue in the field of Machine Learning and regularization is one of the techniques that can be used to overcome this problem. The most common regularization techniques are L1 and L2 regularizations.

Before analyzing the basics of the algorithm, a better understanding must be given to overfitting. By reasoning with simple cartesian coordinates, the theoretical function to represent will be $y = f(x)$ and its approximation will be $\hat{y} = g(\hat{x}) = \underline{\theta}\,\underline{x}^n + b$ where $\underline{\theta}$ is the vector containing the weights, $\underline{x}^n$ is the vector containing the increasing power of the polynomial and b is a bias. As a principal, overfitting could compromise the outcome of the approximation if the degree $j \in [0, n]$ is higher than a threshold k, for which we still have a balanced fit. So, the idea is to "shrink" the number of parameters of the polynomial to k to avoid overfitting, and the way regularization achieves this shrinkage is by manipulating the reconstruction loss function. In particular:

- L2 regularization

L2 regularization acts directly on the reconstruction function, adding a parameter in this way:

$$Err = \frac{1}{n}\sum Err(y_i - \hat{y}(x_i)) + \lambda \sum \theta_i^2$$

Where err is the chosen cost function, λ is a hyperparameter set by the user and θ_i^2 are the squares of each weight of the polynomial. The reasoning behind this method stands basically in penalizing the largest weights (hence the squares), and consequently slowing down the training process, but making sure that the convergence wont overfit the training observations.

- L1 regularization

$$Err = \frac{1}{n}\sum Err(y_i - \hat{y}(x_i)) + \lambda \sum |\theta_i|$$

L1 regularization is basically the same as L2 but uses the absolute values of the parameters instead of the squares (hence a smaller impact on higher weights).

Essentially, the choice between L1 and L2 kernel regularization techniques lies within the essence of the problem, so depends on the chosen hyperparameters, the number of kernels and the cost function.

Learning Rate Scheduling

Batch learning is a particular corollary of gradient decent algorithms in which each batch can be considered as a sample drawn from the training set. As already stated, treating the training set in batches is very important to improve the convergence velocity of the cost function, but, in return, inevitably increases the noise of the gradient. A noisy gradient mustn't be misunderstood as only detrimental to the training procedure: it is true that it could lead to an extensive oscillation around the global minimum, but it could also help to escape local minima, especially in the early stages of the training. Learning Rate Scheduling is the idea behind achieving these two goals: by applying a decay function to the learning rate, initial larger values at the beginning of

training would act avoiding local minima, and last lower values at the end of training would act against the oscillations.

Decay schedulers set the learning rate λ as a function of the iteration step t; the most common are:

- Exponential Decay

  $$\lambda_t = \lambda_0 \, e^{-rt}$$

  Where λ_0 is the initial learning rate and r is the decided decay rate.

- Piecewise Constant Decay

  The values of the decay rates are kept constant for a set number of steps. Typically, the values get halved at every skip.

- Polynomial Decay

  $$\lambda_t = (\lambda_0 - \lambda_N)\left(1 - \frac{t}{\Delta t}\right)^p + \lambda_N$$

  Where λ_0 is the initial learning rate, λ_N is the end learning rate, Δt is the number of the decay steps and p is the desired power of the polynomial.

- Cosine Decay

  $$\lambda_t = \lambda_0 \, d = \lambda_0 \, (1 - \alpha) \, C + \alpha = \lambda_0 \, (1 - \alpha) \frac{1}{2}\left(1 + \cos\frac{\pi \hat{t}}{\Delta t}\right) + \alpha$$

  $$\hat{t} = \min(t, \Delta t)$$

  Where λ_0 is the initial learning rate, Δt is the number of the decay steps, p is the desired power of the polynomial and α is an added term.

4 SIGNAL PROCESSING

4.1 DISCRETE FOURIER TRANSFORM AND LOWPASS FILTERING

The global setting of the necessary tools to perform vibration measurements can be referred as measurement chain: the sensor picks up electrical voltage variations that are conditioned to avoid noise distortion. The data acquisition board takes an analog signal and transforms it into digital, in the sense that a wave is discretized into a step function. After these steps, the data processing can commence.

The Fourier Transform is a very useful tool to capture relevant information on the harmonic components of the signal in the frequency domain. Fourier Series can replace the Fourier Transform provided the assumption of a fictitious period, equal to the duration of the phenomenon. The coefficients of the Fourier Series are spectral quantities defining the amplitude of the harmonic components of the recorded signal. Using Fourier Series, frequency is discretized, since:

$$X_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)e^{-i2\pi\,\Delta f\ t}dt$$

By writing $x(t)$ in the amplitude and phase format, and discretizing the period in k intervals:

$$x(k\Delta t) = \sum_{n=1}^{\frac{N}{2}} R_n \sin(2\pi n\,\Delta f\ k\,\Delta t\ +\ \varphi_n)$$

and deriving that the mean squared value of the recorded signal that is represented through Fourier Series is equal to the sum of the squares of the amplitudes:

$$E[x^2(t)] = \sum_{n=1}^{N} \frac{R_n^2}{2}$$

The spectrum of the squares of the Fourier Series coefficients represents the frequency components in terms of contribution to the mean squared value. Since the latter is often taken to represent the energy of the signal (in physical applications, squares of functions are often related to energy), so the coefficients R_n are giving the distribution of this energy over frequency. The mean squared value of the recorded signal averaged over a number of realizations is called Power Spectral Density (PSD):

$$G_X(n\Delta f) = \lim_{T\to\infty} T\, E\left[\frac{R_n^{(j)^2}}{2}\right]$$

As said, the PSD is a measure of the energy distributed along the frequency. Since the input is a series of data, and since during data acquisition the analog signal has been discretized into a sequence of values, according to the sampling frequency:

$$x(t) = \hat{x}(k\Delta t) \qquad k = 1, \dots, N$$

The further assumption will be:

$$\hat{x}(0) = \hat{x}(N\,\Delta t)$$

The direct consequence of the discretization is the computation of the Discrete Fourier Transform (DFT):

$$\hat{X}(n\Delta f) = \sum_{k=0}^{N-1} \hat{x}(k\Delta t) e^{-2i\pi k\frac{n}{N}}$$

The index k runs through time, whilst the index n runs through frequency, identifying the Fourier Series coefficient. The DFT is what is computed in practice with numerical tools to represent Fourier Series coefficients.

The choice of the time step plays a crucial role, so a criteria must be implemented: to capture a waveform, at least two samples must be found per period (i.e. Nyquist frequency is $\frac{f_{sampling}}{2}$) , so it means that the time step must be smaller than $\frac{T}{2}$, so, for multi-harmonic components:

$$\Delta t < \frac{T_{min}}{2} = \frac{1}{2f_{max}}$$

Violating this criterion would mean capturing an erroneous lower frequency waveform, denoting a phenomenon called *aliasing.*

The DFT is created with the Fast Fourier Transform (FFT) algorithm that can be paired also with a Lowpass Filter algorithm, which acts on the signal in the frequency domain by damping out frequencies higher than a cutoff decided by the user. This is very important in problems regarding unknown ambient excitations, since it is most likely that ambient noise could compromise the clearness of the signal, also when sampling at lower rates.

4.2 IMAGING TIME SERIES: MARKOV TRANSITION FIELD

The past few decades have shown immense importance in applying Deep Learning methods to time series data to solve real issues involving SHM scenarios. However, traditional numerical methods in forecasting both univariate and multivariate time series have proven to be difficult due to their problematic features or inability to preserve temporal correlation, lack of pre-trained models and difficulty in training ANN models. Traditional models also tend to act erroneously when presented with multiple input variables and incorrectly pre-processed time series data. This has posed the question of leveraging existing and less common techniques in ML to improve the predictive performance on time series data. Inspired by recent successes in supervised and unsupervised learning techniques in computer vision, the problem can be considered by encoding time series as images to allow machines to visually recognize, classify and learn structures and patterns. This enables common DL algorithms, such as CNN, to exploit their fullest performance capacity.

The concept of reformulating features of time series data as visual clues has already been explored by data scientists in the past, with revolutionary improvements concentrated in the past decade. In temporal order, Hermansky [34] proposed a method to transform acoustic/speech data into images. Donner [35] proposed a method, posing recurrence networks, to analyze structural properties by building adjacency matrices from recurrence functions to interpret time series. Campanharo [36] proposed to extend the concept of building adjacency matrices by extracting transition dynamics from first order Markov matrices. These complex visual models all successfully present different topological properties amongst different time series. That is, they maintain most spatial relationships of data by representing objects (points, lines and area features) as underlying graph of topological primitives (nodes, faces and edges). However, these models fail to maintain a relationship to the original time series data due to their lack of exact inverse operations. Thus, the problem becomes finding an acceptable method to reformat time series data as visual clues, whilst maintaining a relationship with the original time series data via exact inverse operations. Two novel representations for encoding time series data as images was postulated by Oates & Wang [37], sought to address the issues previously mentioned: the Gramian Angular Field (GAF) and the Markov Transition Field (MTF).

The GAF relies on the concept of the Gramian matrix, enabling to highlight the correlation between each set of observations to further be used in prediction. The Gramian matrix calculates the inner products of vectors in a 2D space, definable by cartesian coordinates, therefore a pre-processing polar ordinate transformation method of must be applied to the 1D observation sets. The key property of this method is that it insures exact inverse operations to revert to the original time series format. However, the inner product in the 2D polar space has several limitations, such as inevitable bias in favor to the more recent observation, as the norm increases with time. Alternatively, considering the trigonometric sum or difference between each point still identifies the temporal correlation between each point and accounts for the limitations. The MTF is an extension to the algorithm proposed by Campanharo. The motivation behind the MTF is to preserve temporal and frequency

correlation dependencies embedded in the time series data. Thus, the MTF must encode dynamic frequency information in the temporal ordering incrementally. It is based off the concept of the Markov matrix, a square matrix that is used to describe the transitions of a Markov Chain and describes the probability of moving from one state to another in a dynamic system. As prior GAF, this method preserves an exact natural inverse, satisfying the previously stated limitations. A description of the adopted method of MTF will be presented. To create a Markov matrix of a series of observations, which will be referred as sample, the data must be discretized along the different values it can take.

Given a time series $X = \{x_1, x_2, \dots, x_n\}$, the values can be discretized into a number Q of bins, meaning each value x_i is mapped to a q_j with $j \in [1, Q]$. After this process, a QxQ weighted adjacency matrix W is constructed by counting transitions among bins in the manner of a first order Markov chain along each time step. In matrix W, $w_{i,j}$ is given by the frequency with which a point in the quantile q_j is followed by a point in the quantile q_i, meaning:

$$W = \begin{bmatrix} w_{11|P(x_t \in q_1 | x_{t-1} \in q_1)} & \cdots & w_{1Q|P(x_t \in q_1 | x_{t-1} \in q_Q)} \\ \cdots & & \cdots \\ w_{21|P(x_t \in q_2 | x_{t-1} \in q_1)} & \cdots & w_{2Q|P(x_t \in q_2 | x_{t-1} \in q_Q)} \\ \cdots & & \cdots \\ w_{Q1|P(x_t \in q_Q | x_{t-1} \in q_1)} & \cdots & w_{QQ|P(x_t \in q_Q | x_{t-1} \in q_Q)} \end{bmatrix}$$

By nature, W is insensitive to the distribution of X and temporal dependencies on time steps t_i. However, if temporal dependencies are not captured in the imaging algorithm, too much information is lost from W. The MTF acts as an extension of the Markov matrix: by spreading out matrix W which contains the transition probability on the magnitude axis into the MTF matrix by considering temporal positions. By assigning the probability from the quantile at time step i to the quantile at time step j at each pixel M_{ij} , the MTF M encodes the multi-span transition probabilities of the time series. Thus, the matrix becomes square and $n_{pixel}\ x\ n_{pixel}$:

$$M = \begin{bmatrix} \omega_{ij|x_1 \in q_i | x_1 \in q_j} & \cdots & \omega_{ij|x_1 \in q_i | x_n \in q_j} \\ \cdots & & \cdots \\ \omega_{ij|x_2 \in q_i | x_1 \in q_j} & \cdots & \omega_{ij|x_2 \in q_i | x_n \in q_j} \\ \cdots & & \cdots \\ \omega_{ij|x_n \in q_i | x_1 \in q_j} & \cdots & \omega_{ij|x_n \in q_i | x_n \in q_j} \end{bmatrix}$$

$M_{ij\ |\ |i-j|=k}$ denotes the transition probability between the points with time interval k.

Signal to image construction with Markov Transition Fields has mainly two focal points that must be regarded in order to increase the correct pattern-elaboration necessary for feature extraction: low sparseness and high resolution. Sparseness is mainly controlled by two factors: the number of instances for each sample and the topology of the bins, whilst high enough resolution is mainly controlled by the blurring kernel. The number of instances per sample must be set to a minimum value for which the reconstruction of the patterns is the roughly the same in each sample itself. The patterns are implicitly controlled by frequency, so this number is set in a way in which the first order modes are always the same for each sample, and this is done by controlling the DFTs.

For a steady state, normally distributed vibration, since the algorithm processes the signals in the time-amplitude domain, a preliminary standardization over each sample can be made, leading to standard normal distribution over each sample (0 mean, 1 standard deviation). Standardization is a useful tool because the bin distribution can also be normal. A normal bin distribution on a normally distributed sample lowers sparseness for any number of bins, compared to a uniform bin distribution. Nevertheless, the number of bins does not cease to be a hyperparameter, since the right number of bins still enhances the number and the clearness of the features in the images. Each pixel represents a probability (a number between 0 and 1), and since the aim of the process must be to recreate patterns in such a way that these are highly visible, their difference in absolute values must be high enough. A very high number of bins would decrease this difference, so sparseness would increase, and a very low number of bins would end up lowering the number of features to retrieve.

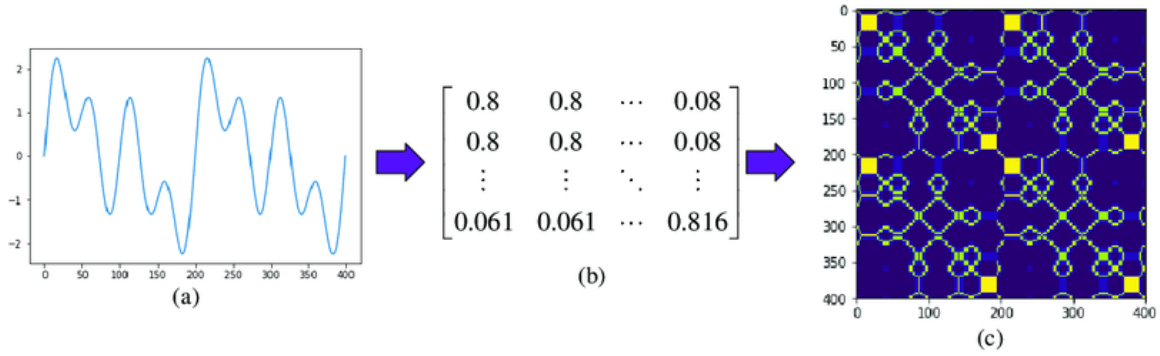

Figure 3 - Different stages of the MTF creation: a) raw signal b) Markov Transition Matrix c) Markov Transition Field

For what concerns the blurring kernel, which is obtained by averaging pixels in each non-overlapping square patch, this must be set in such a way the features are still visible and retrievable, but also keeping in mind that a higher number of pixels will lead to a higher computational burden during the training phase.

Bin managing: SAX Algorithm

To overcome the high dependency on the optimal bins' setting, the Symbolic Aggregate Approximation (SAX) algorithm [38], [39] is briefly summarized. Through the application of SAX analysis, the original time-series signal is transformed into an equiprobable symbolic sequence.

For a time series C of length n, a lower dimensional vector can be obtained by standardizing the n-dimensional time-series data and then converting it into a w-dimensional time series D that has the elements according to:

$$\overline{c_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j$$

w is defined as the window size and n as the length of the signal. Typically, w is much smaller than n. This simple representation is known as Piecewise Aggregate Approximation (PAA), which is the first step in

obtaining the SAX representation. The subsequent process is to transform the lower dimensional time series D into a number of equiprobable symbols. Having transformed a time series database into the PAA a further transformation to obtain a discrete representation can be applied. It is desirable to have a discretization technique that will produce symbols with equiprobability.

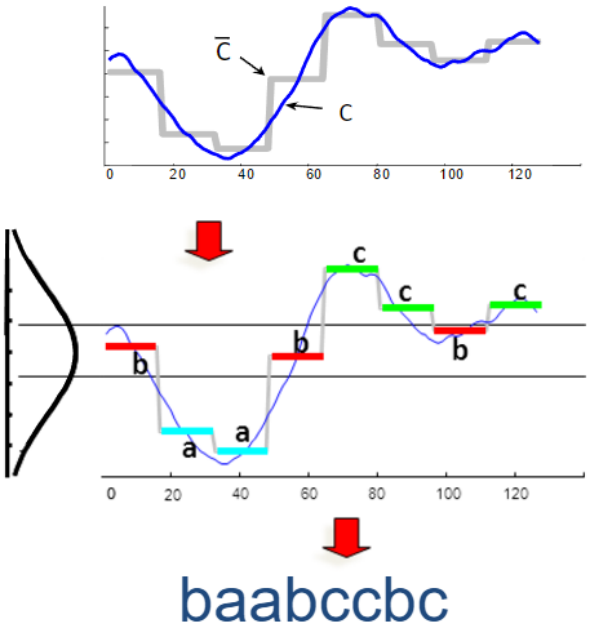

Figure 4 - SAX of a trial sinusoid function

As already stated, given that normalized time series have highly Gaussian distribution, what remains is the determination the “breakpoints” of the bins that will produce equal-sized areas under a Gaussian curve. Once the breakpoints have been obtained, the discretization of a time series can be made by mapping the PAA coefficients between the breakpoints with symbols. The concatenation of the symbols is called a *word* [39] .

5 CASE STUDY: THE S101 BRIDGE

5.1 INTRODUCTION

The S101 Bridge was a prestressed concrete bridge from the early 1960s and therefore a characteristic representative of the European highway asset. This case study was originally proposed for the monitoring of a progressive damage scenario induced by lowering a pier. A progressive damage description is given in 5.1.1. An unknown ambient excitation was considered as loading condition, since the traffic was closed on the bridge itself, but maintained open underneath. For the vibration measurements a BRIMOS® measurement system with a permanent sensor grid was used. The grid consisted of 15 sensor locations on the bridge deck (FIGURA) and in each location, three sensors for measurements in the bridge deck's vertical, longitudinal and transversal direction. The sensor layout was motivated by the symmetry of the bridge, so that 14 locations were set only on one side of the bridge deck, while the 15th location on the other side had the purpose of distinguishing between bending and torsional modes. All the data was sampled at a frequency of 500 Hz and no temperature changes were recorded during monitoring.

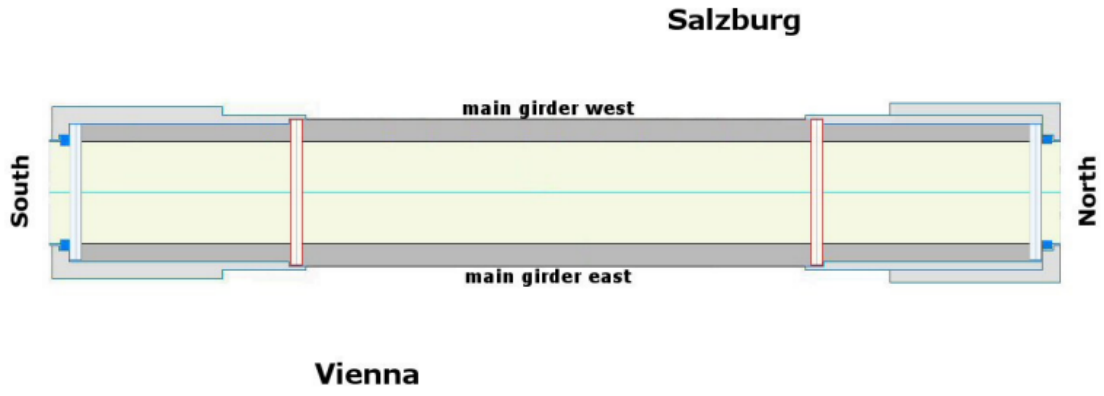

Figure 5 - S101 Bridge, planar view

5.1.1 Progressive Damage Description

The case in object features the damage of one of the four piers, which was introduced by cutting a slice through its lower end, causing a change in the boundary conditions, since an extra hinge was formed just above the foundation. The removed slice was 5mm thick, and the column was steadily lowered, first by 3 cm, then by 2 cm, all while moderate cracks were caused in the adjacent pier; nevertheless, no cracking in the bridge deck was noticed. The settling of the pier was intentionally set to force the bridge deck to reach its elastic limit. After returning the column in its original position, steel plates were inserted to re-establish the clamp above the foundation.

Figure 6 - a) Cutting of the pier; b) insertion of the steel plates; c) lowering with a hydraulic jack

5.1.2 Preliminary Modal Analysis

In order to have an indication about the natural frequencies, we refer to the modal analysis performed in [40]. In Table 2 [41], the system identification results with their variation coefficients are summarized. In [41] the

mode shapes are summarized as follows: Modes 1, 3 and 5 are vertical bending modes and modes 2 and 4 are torsional modes.

Mode	f (Hz)	$\bar{\sigma}_f$	ξ (%)	$\bar{\sigma}_\xi$
1	4.036	0.12	0.78	15
2	6.281	0.08	0.56	20
3	9.677	0.18	1.3	14
4	13.27	0.13	1.5	13
5	15.72	0.37	1.3	17

Table 1 - System identification results from preliminary modal analysis

5.2 PROPOSED METHODOLOGY

The proposed methodology to detect the anomalies caused by the progressive damage consists in the following five steps:

1. Signal selection: since the nature of the problem is recovering the hidden features of the undamaged vibration in the time-amplitude domain, the user must know in advance which will be the most probable signals that will suffer the biggest reconstruction error dispatchment in the damaged states. This means selecting the right signal's position and direction of reference. A good reconstruction could lead to no anomaly detection if the retrieval of the latter is misplaced.

2. Frequency filtering: the damage reflects on the natural frequencies of the structure by lowering their values. Since the vibrations are naturally induced by unknown ambient excitation, the modal frequencies could be confused with higher amplitude noise. This would lead to high sparseness in the MTF images to be processed by the auto-encoder, likely to hamper its reconstruction capabilities. For this reason, cancelling out noise through a custom lowpass filter is crucial for the Auto-Encoder to work properly.

3. Creation of the images, with reference to *Chapter 3*.

4. Setting of the Convolutional Auto-Encoder: the depth of the autoencoder and the kernel sizes must be set to reconstruct the training samples with precision.

5. Post-processing of the reconstruction error: in this work, the reconstruction error will be always retrieved through the Mean Absolute Error (MAE) score. Each MAE gives a measure of how well an image is being reconstructed, but alone, this could not yield a clear indication of the structural health state. To increase the possibility to well assess the latter, the problem is shifted from a regression one to a classification one. The MAE distribution in the training period allows to fix a proper threshold T_1, corresponding to the 90-th percentile, in order to build a damage detector. Then, the vector labeling each record with 0 (healthy) or 1 (damaged) is divided into 10-min long macro-sequences. Hence, if the percentage of inner damaged records exceeds a defined threshold T_2, the macro-sequence is classified as damaged. Since healthy and damaged test data are available, ROC curves are used to estimate the optimal threshold $T_{2,opt}$ [42] as the value of T_2 in the point of maximum curvature.

The damage case consists in a simulation of a fast foundation settling by cutting a portion of a pier and lowering the column by means of a hydraulic jack. The location of the cut is the Northern-Western pier. Five different damages stages can be differentiated as follows:

- Damage Case A1 – Column cut through
- Damage Case A2 – First step of lowering
- Damage Case A3 – Second step of lowering
- Damage Case A4 – Third step of lowering
- Damage Case A5 – Insertion of steel plates

The predicted modal analysis [43], averaged over all the sensors, particularly regarding the drop in natural frequencies, is collected in table x.

Modes | Case	Undamaged	A1	A2	A3	A4	A5
	Hz	Hz	Hz	Hz	Hz	Hz
1st bending	4,05	3,95	3,96	3,92	3,62	3,98
1st torsion	6,30	6,08	6,01	5,88	5,88	5,91
2nd bending	9,69	9,44	9,44	9,28	9,28	9,34
2nd torsion	12,15	12,15	11,65	10,70	10,79	10,92
3rd bending	15,83	15,83	16,22	15,96	15,79	15,79

Table 2 - Natural frequencies of the S101 bridge

5.3 Single Channel Signals through Individual Convolutional Auto-Encoders

In order to maximize the damage-sensitivity of the processed measurements, sensor E202 in the proximity of the damage is selected [40]. The signals in the vertical and transversal direction are studied, since the change in the boundary conditions would highlight larger modal frequency changes in the vertical direction and spurious horizontal vibrations in the transversal direction are most likely to be activated.

Being interested in the identification of the first modes in a frequency range of 0–18 Hz, the data was initially down-sampled from a rate of 500 Hz to 50 Hz.

Training and test vibration samples are created. Each sample covers just over 1 minute of monitoring, collecting 3072 entries. The vertical signal is pre-processed by a lowpass filter set at 7 Hz Figure 7.a and the across signal set at 16 Hz Figure 7.b. Each entry in the sample is then standardized over the sample. This was the first contribution in reducing the sparseness in the images to an acceptable amount.

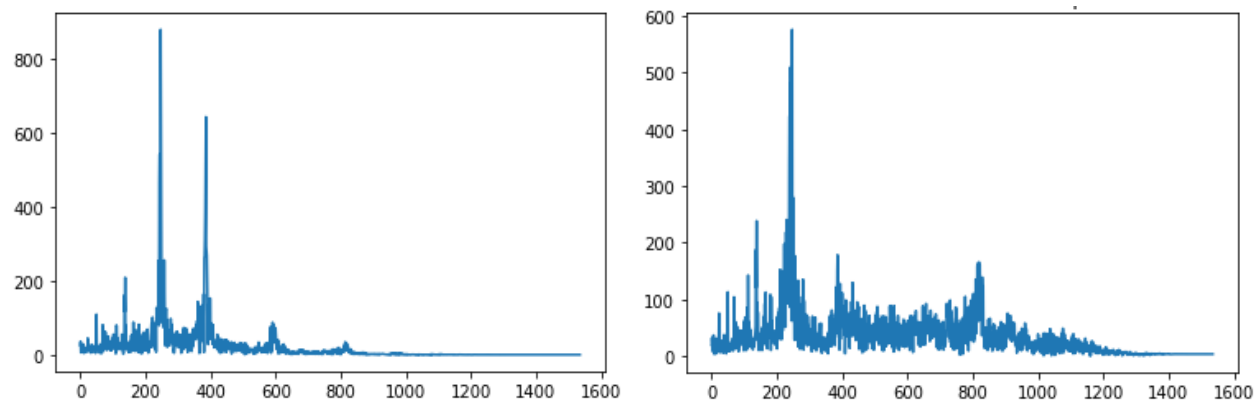

Figure 7 – Discrete Fourier Transform : a) vertical signal ; b) transversal signal

Each sample is then fed to the MTF algorithm, using 20 normally distributed bins and the MTF images are reduced to a size of 32x32 pixels.

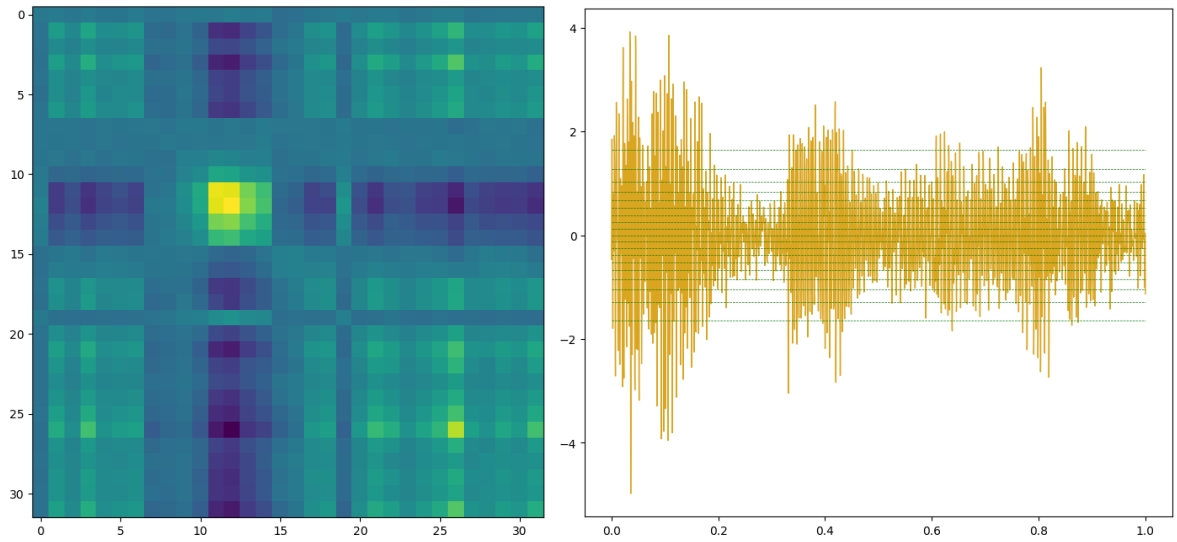

Figure 8 - Vertical signal: a) MTF image b) filtered signal

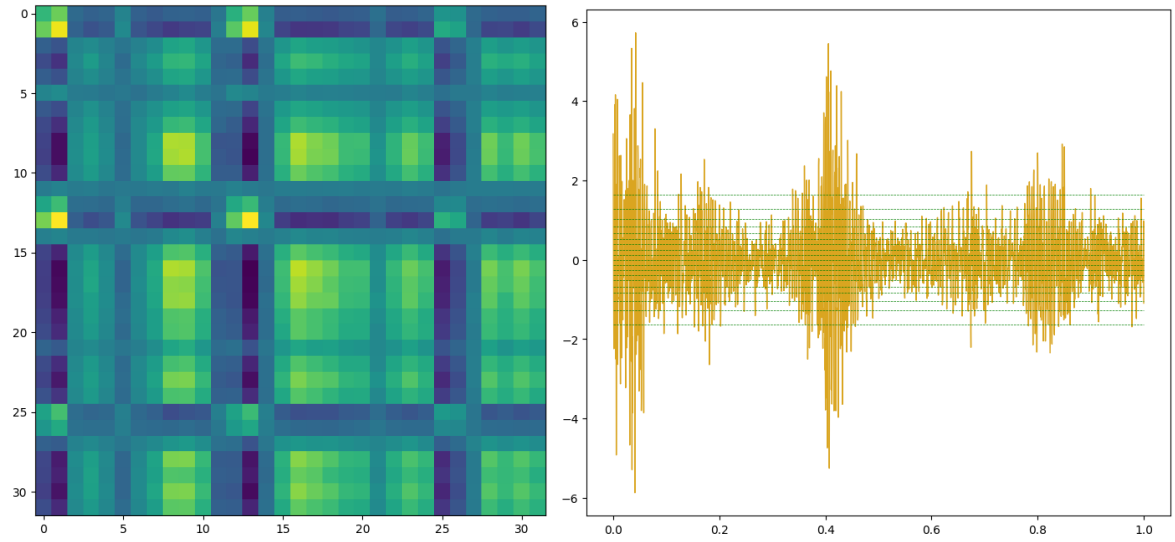

Figure 9 - Transversal signal: a) MTF image b) filtered signal

The Auto-Encoder of the vertical signal is composed by 3 convolutional layers in the Encoder, respectively using kernels of size 3x3x16, 3x3x32 and 3x3x64, activated with the Re-Lu function, zero padding and no L2 regularization. Each convolutional layer is followed by a Max Pooling layer of size 2x2 with zero padding. The Decoder is specular to the Encoder. The output layer has a kernel of size 3x3x1 to return to the input size and a sigmoid function is used to squash each pixel back to a value between 0 and 1, as is the input. The training is set for 1000 Epochs, the initial learning rate is set at 0.001 and a Cosine Decay Scheduling is applied, as shown in fig. The batch size was set to 8 since the number of the samples was very low, in order to avoid as many convergence issues as possible. The cost function used is Mean Squared Error (MSE), but also the

Mean Absolute Percentage Error (MAPE) was taken as a metric. Figure 10 a,b show respectively the MSE and the MAPE versus the Epoch. The final convergence values of MSE are respectively 2.44e-05 and 3.73e-05 for training and validation, showing good performance against overfitting. For MAPE, the final convergence values are 7.06% and 8.1%, confirming the last statement.

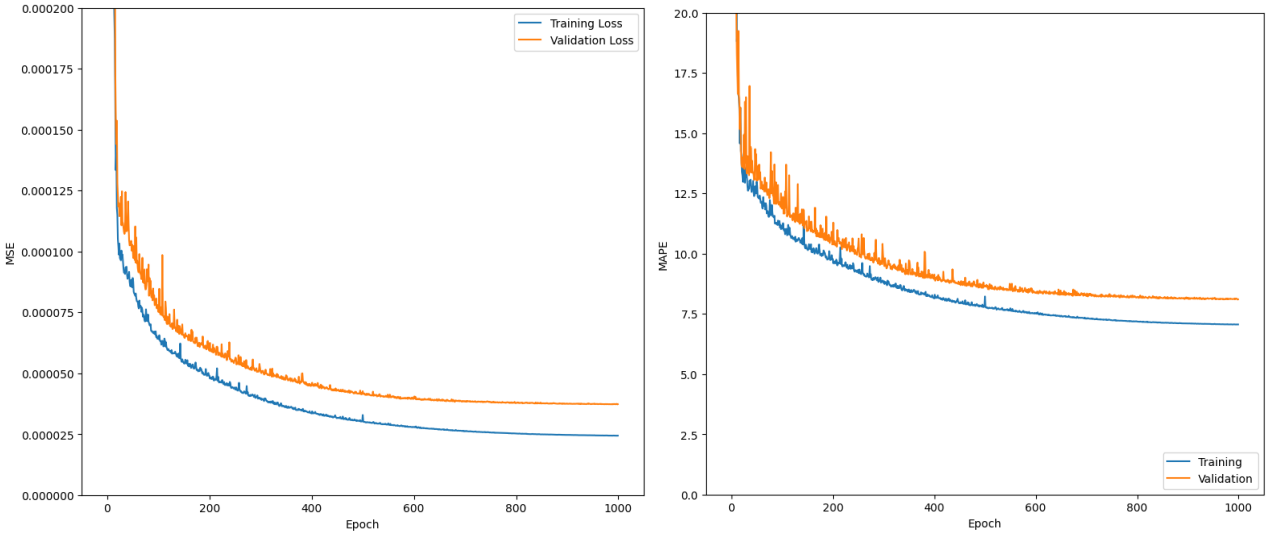

Figure 10 - a)MSE ; b)MAPE of the vertical signal

Then the reconstruction error between input and output is calculated via MAE (Figure 11). The good performance in anomaly detection is already very visible, except for damage state A1, but this is acceptable since this is the state in which the damage is such that very poor changes are reflected on the modal parameters.

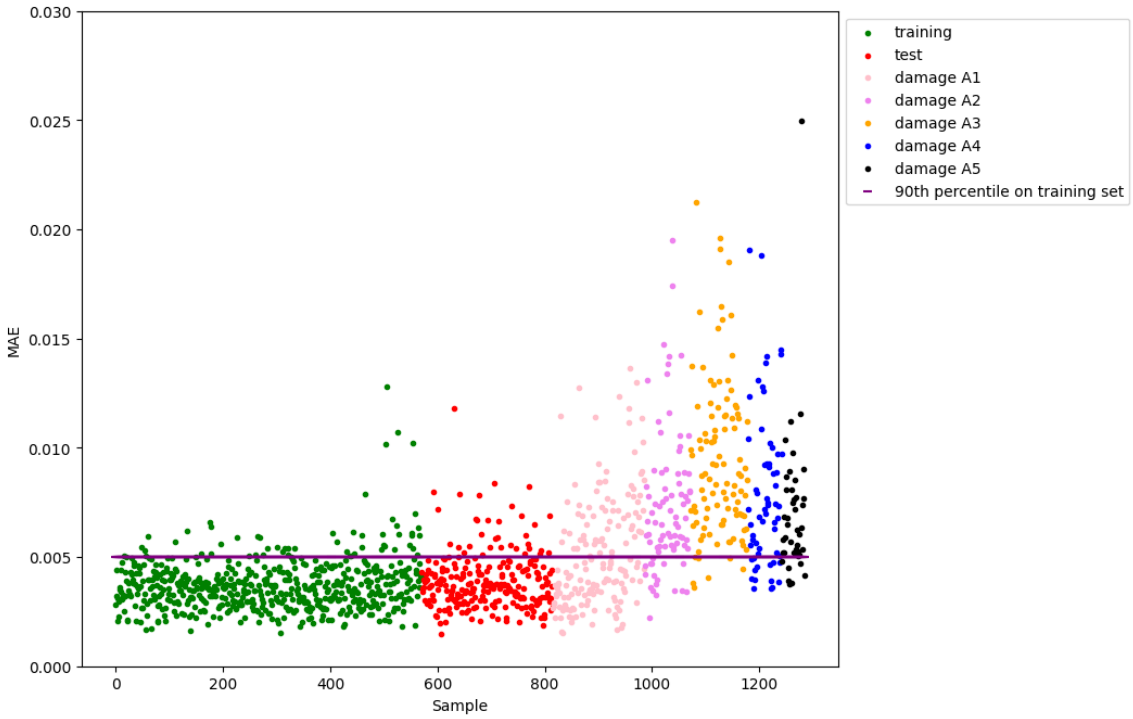

Figure 11 - MAE of the vertical signal

By investigating the training error distribution, the 90th percentile is equal to T_1=0.005 (Figure 12).

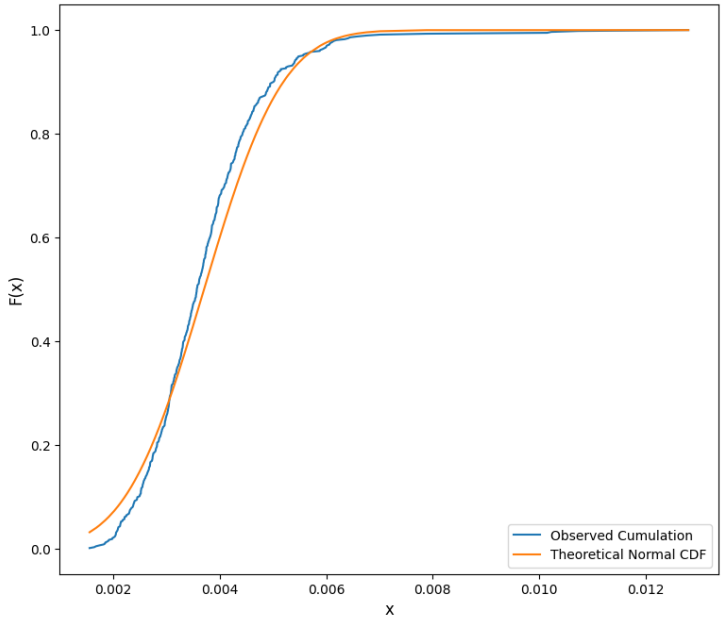

Figure 12 - Training error distribution of the vertical signal

Figure 13 shows the ROC curves for values of the thresholds T_2 ranging between 0 and 1. The indication of the precision is given by the AUROC, showing that by neglecting damage state A1, the proposed methodology clearly identifies damaged and undamaged states with 99,7% accuracy. By considering only the case A1, the accuracy is of 85.8%. The optimal value of T_2 considering all the damage states is 0.3.

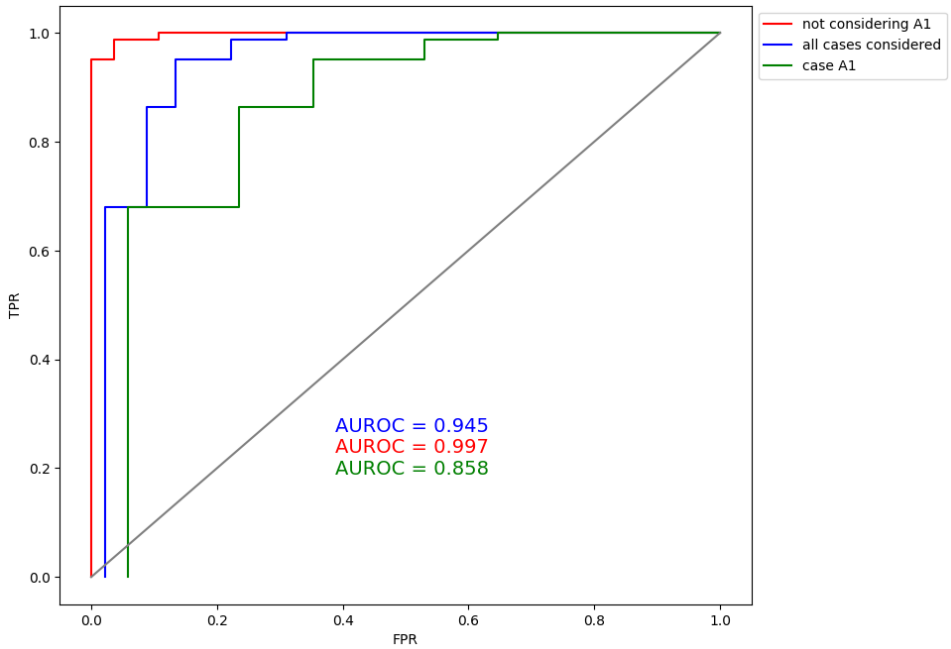

Figure 13 - ROC curves of the vertical signal's damage detection procedure

The Auto-Encoder in the transversal direction is built with the same architecture as the one proper of the vertical direction, but the kernels are downsized respectively to 3x3x8, 3x3x16 and 3x3x32. Also, the same

optimization tools are used. Figure 14 a,b show respectively the MSE and the MAPE versus the Epoch. The final convergence values of MSE are 6.32e-05 for training and 8.47e-05 for validation, showing good performance against overfitting, but a worse reconstruction precision. For MAPE, the final convergence values are 11.27% and 12.99% confirming the last statement. The MAE distribution is shown in Figure 15.

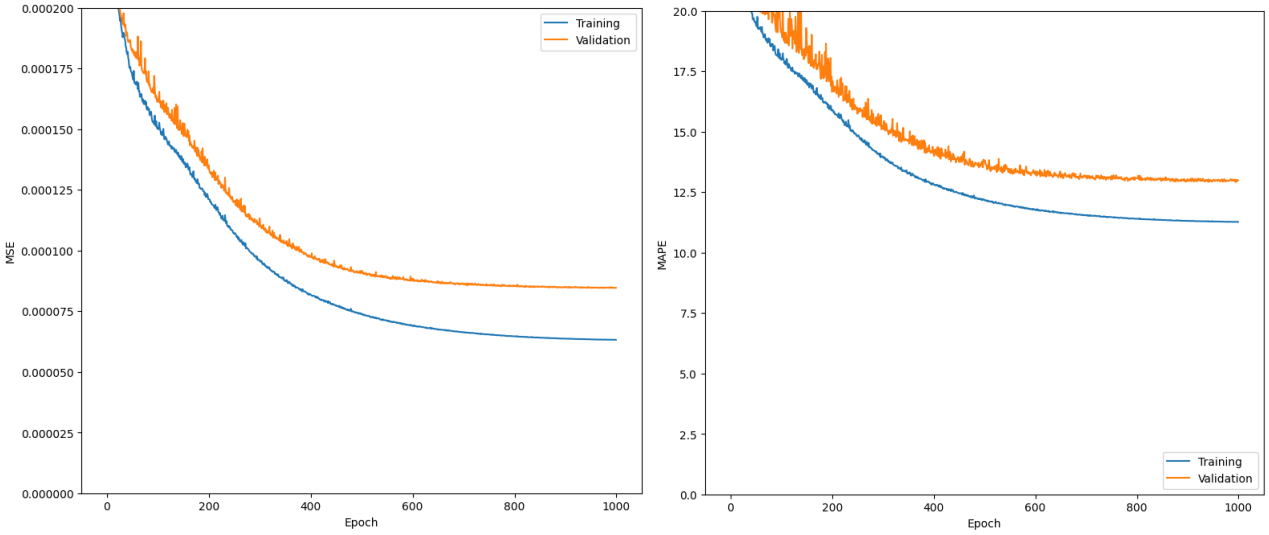

Figure 14 - a)MSE; b)MAPE of the transversal signal

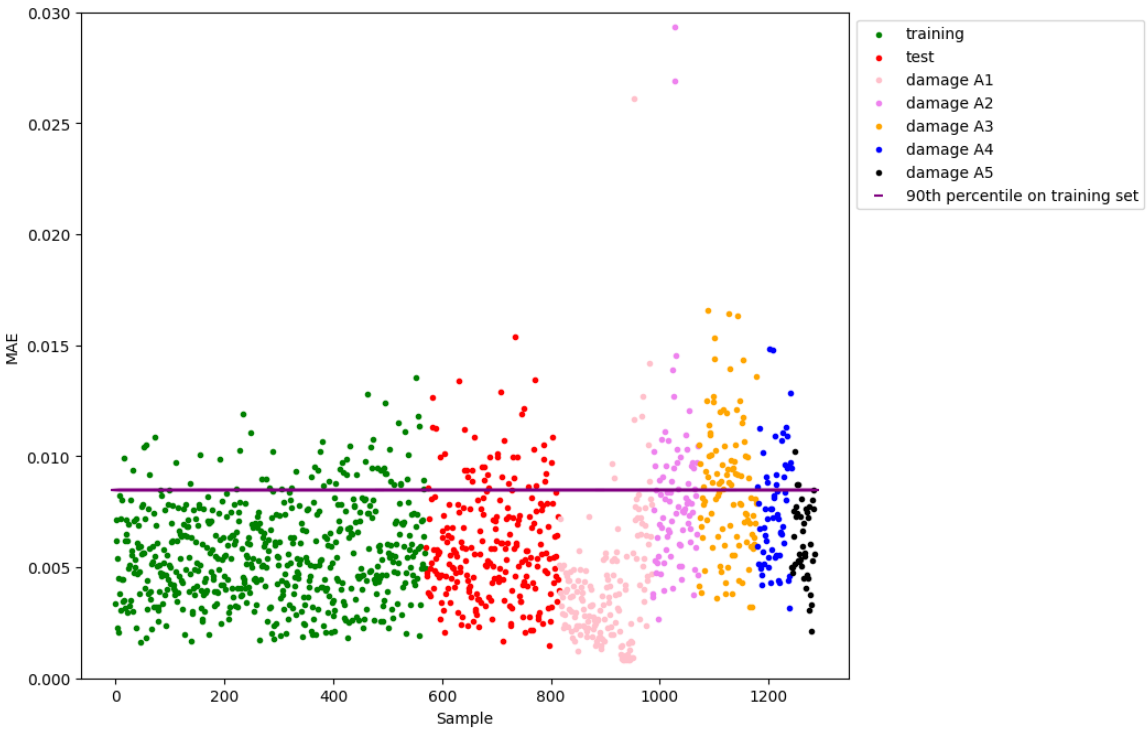

Figure 15 - MAE of the transversal signal

The damage states are less visible, since the reconstruction procedure is less efficient due to the excessive sparseness in the images and since the local spurious horizontal vibrations have a lesser effort in differentiating

the various stages. In particular, stages A1 and A5 have a very bad performance, but this is backed up by the fact that, during these stages, the boundary conditions are still transitioning, and cannot be fully considered as a clamp. This means that the spurious horizontal vibrations are yet to show up for A1 and stopped before A5.

Nevertheless, the 90th percentile of the training error is equal to T_1=0.0085 and the ROC curves are shown in Figure 16. The performance to be looked at is the one neglecting stages A1 and A5, showing 88.7% accuracy and the optimal value of T_2 is 0.15.

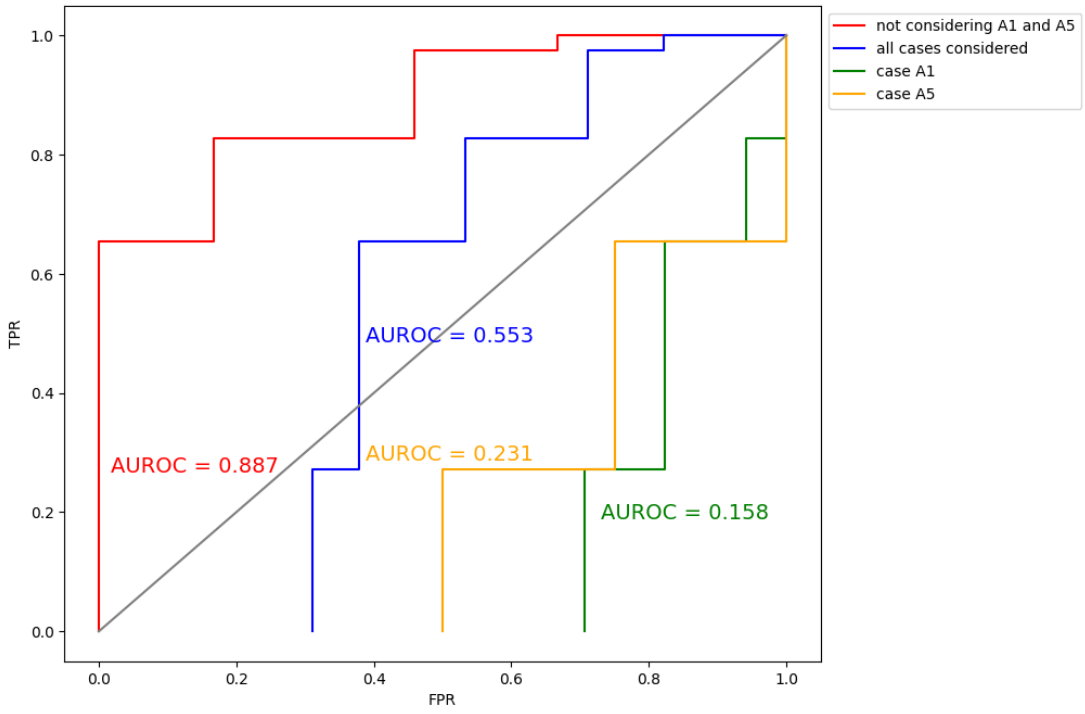

Figure 16 - ROC curves of the transversal signal's damage detection procedure

5.4 Multi-channel Mosaics through a Single Convolutional Auto-Encoder

To improve the accuracy of the anomaly detection in the transversal direction, a further attempt is made by processing more signals at once by means of a single Convolutional Auto-Encoder. Multi-Channel inputs can be processed by simply aggregating four 32x32 pixel images in a 4x4 mosaic, obtaining a 64x64 pixel image Figure 17. The aim is to try and cumulate the error over the four signals to improve the MAE dispatchment in the damaged states. Also, processing four signals in the same direction ideally shouldn't complicate the architecture of the network, since, in principle, the kernels act on the same features among the 4 portions of the image. In fact, the utilized architecture is still composed by three convolutional layers in the Encoder, respectively using kernels of size 3x3x16, 3x3x32 and 3x3x64, activated with the Re-Lu function, zero padding and no L2 regularization. Each convolutional layer is followed by a Max Pooling layer of size 2x2 with zero padding. The Decoder is specular to the Encoder. Also, in this case a Cosine Decay Scheduling is used on the learning rate, which has an initial value of 0.001. The training is set for 1000 Epochs and the batch size was

set to 8. The 4 signals chosen are proper of the sensors closest to the damaged pier (E201, E202, E103 and E210) [40].

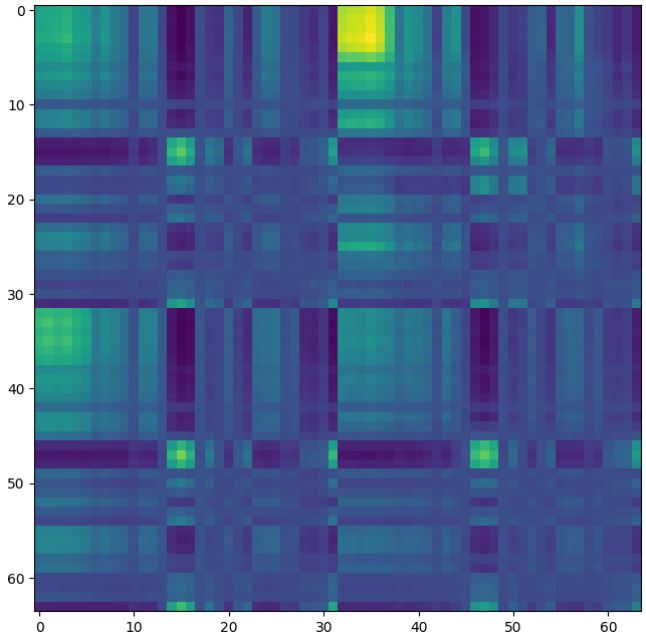

Figure 17 - Multichannel representation through MTF of the transversal signals

MSE and MAPE are plotted respectfully in Figure 18. The convergence values of both the errors show good precision and performance against overfitting, since the MSE converges to 2.5e-05 for training and 4.7e-05 for validation, whilst the MAPE converges to converges to 5.89% for training and 7.47% for test.

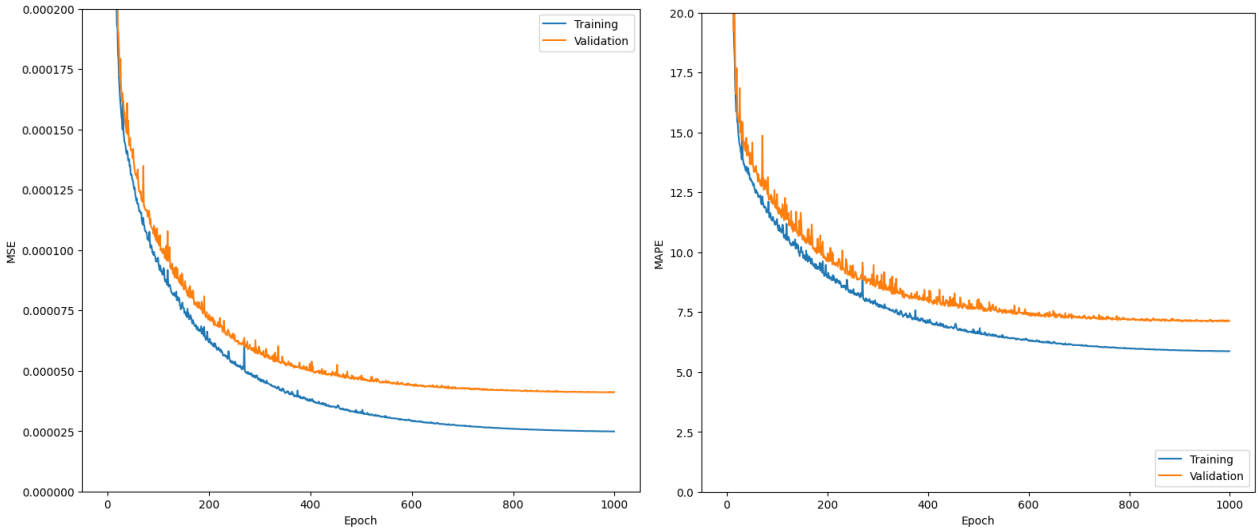

Figure 18 - a)MSE; a)MAPE of the multichannel signal

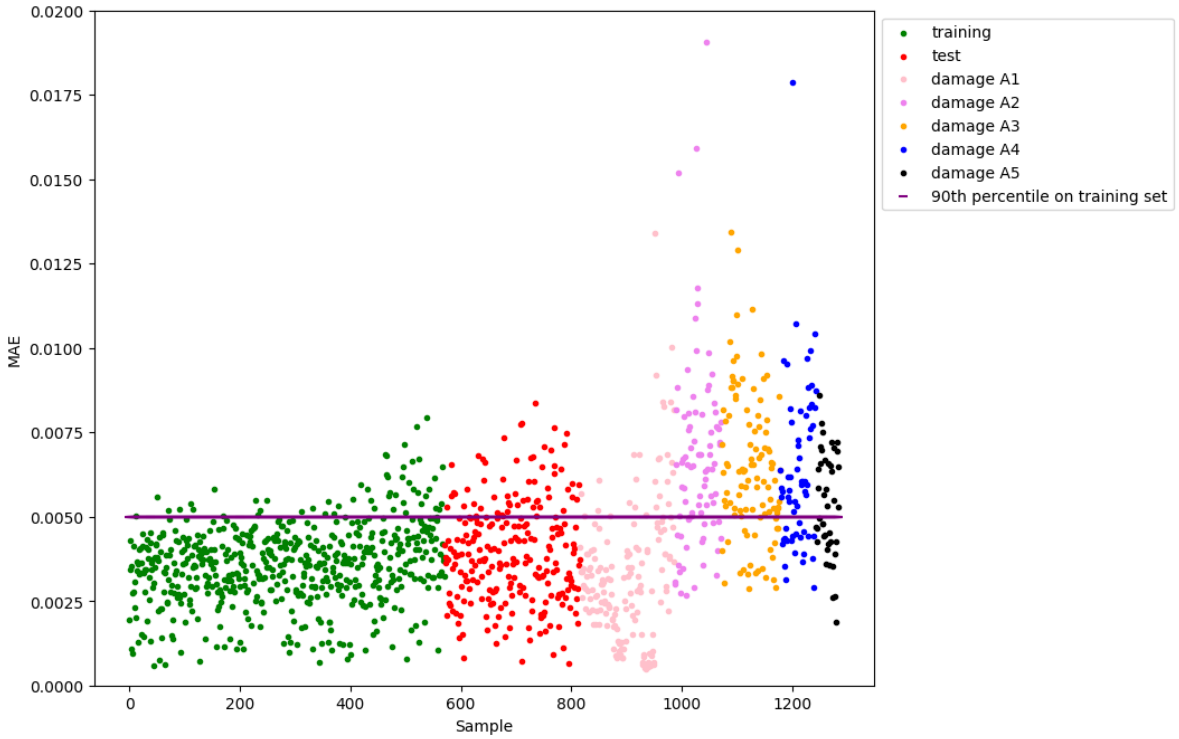

Figure 19 - MAE of the multichannel signal

The MAE distribution is shown in Figure 19. The better performance in damage detection is very clear for stages A2, A3, A4 and A5. Damage stage A1 also improves, but still suffers the transition of the boundary condition mentioned before. The 90th percentile of the training error is equal to T1=0.005 and the ROC curves are shown in Figure 20. As expected, the accuracy of the state detection increases drastically, especially neglecting stage A1, showing a value of 98.4%.

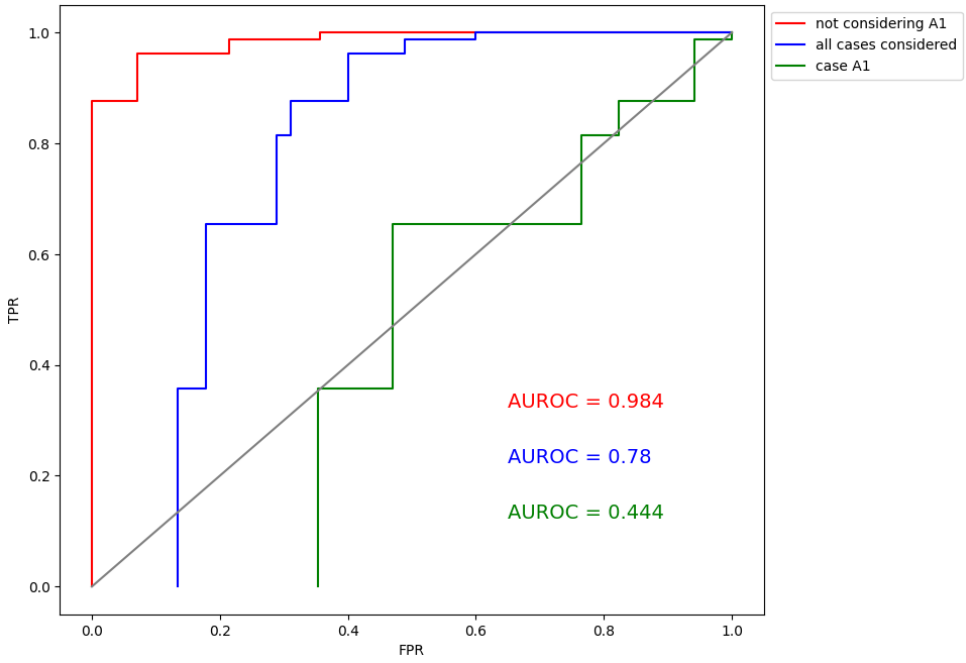

Figure 20 - ROC curves of the multichannel signal's damage detection procedure

6 CONCLUSIONS

This work has illustrated an innovative unsupervised deep-learning-based damage detection technique applied to the S101 Bridge. The methodology uses a Convolutional Auto-Encoder architecture to reconstruct 1-minute-long standardized and lowpass filtered acceleration time series, recorded under unknown ambient excitation and encoded as Markov Transition Field images. Both images created by processing single signals one at a time, and multichannel mosaics are considered.

The reconstruction loss is evaluated through the Mean Average Error score. The damage assessment is afterwards performed by computing the percentage of damaged sequences within a 10 min macro-sequence that are processed by a binary classifier (0 if undamaged, 1 if damaged).

Early results prove the capabilities of the proposed ML algorithm to successfully detect, with limited computational power, the damaged states, as demonstrated by the achieved values of reconstruction accuracy over 90% and AUROC values very close to 1.

Nevertheless, the high resolution of the method is mostly dependent on many users' defined parameters, such as the original size of the images, the lowpass filter's frequency cutoff, the blurring kernel and the kernel size of the convolutional layers in the Auto-Encoder. This directly reflects on the method's generalizability, since a prior investigation of the handled time series and the pre-processing phase are crucial for the neural network to perform well.

For these reasons, future developments should direct the attention of researchers to try and mitigate user defined efforts, in order to be able to create inputs and networks that excel in precision, but also in generalizability.

Bibliography

[1] C. Liu, J. Olund, A. Cardini, P. D'attilio, E. Feldblum, and J. Dewolf, "Structural health monitoring of bridges in the State of Connecticut," *Earthquake Engineering and Engineering Vibration*, vol. 7, no. 4, pp. 427–437, Dec. 2008, doi: 10.1007/s11803-008-1003-8.

[2] A. J. García Palencia and E. Santini-Bell, "Estimation of structural health in concrete bridge decks: A study case," *SHMII-5 2011 - 5th International Conference on Structural Health Monitoring of Intelligent Infrastructure*, no. December 2011, 2011.

[3] H. Sohn, C. R. Farrar, F. Hemez, and J. Czarnecki, "A Review of structural health monitoring Literature 1996-2001," *Library.Lanl.Gov*, pp. 1–7, 2001, [Online]. Available: https://library.lanl.gov/cgi-bin/getfile?00796820.pdf

[4] S. Bianchi *et al.*, *Structural Health Monitoring of Two Road Bridges in Como, Italy*, vol. 200 LNCE. 2022. doi: 10.1007/978-3-030-91877-4_45.

[5] M. Gatti, "Structural health monitoring of an operational bridge: A case study," *Eng Struct*, vol. 195, no. November 2018, pp. 200–209, 2019, doi: 10.1016/j.engstruct.2019.05.102.

[6] Y. Kaya and E. Safak, "Real-Time Structural Health Monitoring and Damage Detection," *Conference Proceedings of the Society for Experimental Mechanics Series*, vol. 39, no. 4, pp. 10–20, 2013, doi: 10.1007/978-1-4614-6555-3.

[7] G. Lu and Y. J. Yang, "Structural health monitoring: an overview," *Internet of Things and Data Analytics Handbook*, pp. 665–674, 2017, doi: 10.1002/9781119173601.ch40.

[8] C. C. Comisu, N. Taranu, G. Boaca, and M. C. Scutaru, "Structural health monitoring system of bridges," in *Procedia Engineering*, 2017, vol. 199, pp. 2054–2059. doi: 10.1016/j.proeng.2017.09.472.

[9] C. Mandache, M. Genest, M. Khan, and N. Mrad, "Considerations on Structural Health Monitoring Reliability," 2011.

[10] C. R. Farrar and K. Worden, "An introduction to structural health monitoring," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1851, pp. 303–315, Feb. 2007, doi: 10.1098/rsta.2006.1928.

[11] B. Culshaw, *Structural health monitoring of civil engineering structures*, vol. 1, no. 3. 1998. doi: 10.1002/pse.2260010313.

[12] L. Rosafalco, A. Manzoni, S. Mariani, and A. Corigliano, “Fully convolutional networks for structural health monitoring through multivariate time series classification,” *Adv Model Simul Eng Sci*, vol. 7, no. 1, Dec. 2020, doi: 10.1186/s40323-020-00174-1.

[13] C. R. Farrar and K. Worden, “An introduction to structural health monitoring,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1851, pp. 303–315, 2007, doi: 10.1098/rsta.2006.1928.

[14] H. Sohn, C. R. Farrar, F. Hemez, and J. Czarnecki, “A Review of Structural Health,” 1996.

[15] T. Pothisiri, K. D. Hjelmstad, and M. Asce, “Structural Damage Detection and Assessment from Modal Response”, doi: 10.1061/ASCE0733-93992003129:2135.

[16] V. M.W., B. J.L., and A. S.K., “Bayesian Probabilistic Approach to Structural Health Monitoring,” vol. 126, no. July, pp. 738–745, 2000.

[17] M. Flah, I. Nunez, W. ben Chaabene, and M. L. Nehdi, “Machine Learning Algorithms in Civil Structural Health Monitoring: A Systematic Review,” *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 2621–2643, Jun. 2021, doi: 10.1007/s11831-020-09471-9.

[18] C. Zhang *et al.*, “A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data,” Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.08055

[19] F. Bakhtiari-Nejad, A. Rahai, and A. Esfandiari, “A structural damage detection method using static noisy data,” *Eng Struct*, vol. 27, no. 12 SPEC. ISS., pp. 1784–1793, 2005, doi: 10.1016/j.engstruct.2005.04.019.

[20] S. Naseer *et al.*, “Enhanced network anomaly detection based on deep neural networks,” *IEEE Access*, vol. 6, pp. 48231–48246, Aug. 2018, doi: 10.1109/ACCESS.2018.2863036.

[21] M. Teng, “Anomaly detection on time series,” in *Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing, PIC 2010*, 2010, vol. 1, pp. 603–608. doi: 10.1109/PIC.2010.5687485.

[22] D. M. Hawkins, *Identification of Outliers*. Springer Netherlands, 1980. doi: 10.1007/978-94-015-3994-4.

[23] S. Omar, A. Ngadi, and H. H. Jebur, “Machine Learning Techniques for Anomaly Detection: An Overview,” *Int J Comput Appl*, vol. 79, no. 2, pp. 33–41, Oct. 2013, doi: 10.5120/13715-1478.

[24] R. P. Finotti, C. Gentile, F. S. Barbosa, and A. A. Cury, “VIBRATION-BASED ANOMALY DETECTION USING SPARSE AUTO-ENCODER AND CONTROL CHARTS.”

[25] A. Berroukham, K. Housni, M. Lahraichi, and I. Boulfrifi, “Deep learning-based methods for anomaly detection in video surveillance: a review,” *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 1, pp. 314–327, Feb. 2023, doi: 10.11591/eei.v12i1.3944.

[26] G. Li and J. J. Jung, “Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges,” *Information Fusion*, vol. 91. Elsevier B.V., pp. 93–102, Mar. 01, 2023. doi: 10.1016/j.inffus.2022.10.008.

[27] O. Avci, O. Abdeljaber, S. Kiranyaz, M. Hussein, M. Gabbouj, and D. J. Inman, “A review of vibration-based damage detection in civil structures: From traditional methods to Machine Learning and Deep Learning applications,” *Mechanical Systems and Signal Processing*, vol. 147. Academic Press, Jan. 15, 2021. doi: 10.1016/j.ymssp.2020.107077.

[28] D. H. Hubel and A. T. N. Wiesel, “54 With 2 plate and 20 text-ftgutre8 Printed in Gret Britain RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT’S VISUAL CORTEX,” 1962.

[29] D. E. Ruineihart, G. E. Hint, and R. J. Williams, “LEARNING INTERNAL REPRESENTATIONS BERROR PROPAGATION two,” 1985.

[30] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, “Gradient Descent Finds Global Minima of Deep Neural Networks.”

[31] K. Fukumizu and S. Amari, “Local minima and plateaus in hierarchical structures of multilayer perceptrons.” [Online]. Available: www.elsevier.com/locate/neunet

[32] N. Qian, “On the momentum term in gradient descent learning algorithms.”

[33] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980

[34] H. Hermansky, “Coding and decoding of messages in human speech communication: Implications for machine recognition of speech,” *Speech Communication*, vol. 106. Elsevier B.V., pp. 112–117, Jan. 01, 2019. doi: 10.1016/j.specom.2018.12.004.

[35] R. v. Donner, Y. Zou, J. F. Donges, N. Marwan, and J. Kurths, “Recurrence networks - A novel paradigm for nonlinear time series analysis,” Aug. 2009, doi: 10.1088/1367-2630/12/3/033025.

[36] A. S. L. O. Campanharo, M. I. Sirer, R. D. de Malmgren, F. M. Ramos, and L. A. N. Amaral, “Duality between time series and networks,” *PLoS One*, vol. 6, no. 8, 2011, doi: 10.1371/journal.pone.0023378.

[37] Z. Wang and T. Oates, “Imaging Time-Series to Improve Classification and Imputation,” May 2015, [Online]. Available: http://arxiv.org/abs/1506.00327

[38] E. Keogh, J. Lin, and A. Fu, “HOT SAX: Efficiently finding the most unusual time series subsequence,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2005, pp. 8–15. doi: 10.1109/ICDM.2005.79.

[39] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing SAX: A novel symbolic representation of time series,” *Data Min Knowl Discov*, vol. 15, no. 2, pp. 107–144, Oct. 2007, doi: 10.1007/s10618-007-0064-z.

[40] “IRIS SP7,” 2009.

[41] M. Döhler, F. Hille, L. Mevel, and W. Rücker, “Structural health monitoring with statistical methods during progressive damage test of S101 Bridge,” *Eng Struct*, vol. 69, pp. 183–193, Jun. 2014, doi: 10.1016/j.engstruct.2014.03.010.

[42] V. Giglioni, I. Venanzi, A. E. Baia, V. Poggioni, A. Milani, and F. Ubertini, “Deep Autoencoders for Unsupervised Damage Detection with Application to the Z24 Benchmark Bridge”, doi: 10.1007/978-3-031-072.

[43] D. M. Siringoringo, T. Nagayama, Y. Fujino, D. Su, C. Tandian, and H. Miura, “Chapter 15-x 15 Vibration Study and Application of Outlier Analysis to the S101 Bridge Full-Scale Destructive Testing Motivation.”