POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Machine learning and deep learning algorithms for anomaly detection

Tesi di Laurea Magistrale in
Mechanical Engineering - Ingegneria Meccatronica

Gianluca Bombaci, 10482291

Abstract: This Master's Thesis work is an extension of two research papers that delve into the utilization of artificial intelligence in the field of Structural Health Monitoring. Changes in the nominal configuration of a structure can often indicate the presence of structural defects that require monitoring to prevent them from escalating to critical conditions. Undoubtedly, the capability to automatically detect alterations in a structure holds significant appeal. When there is a lack of prior knowledge about the system, artificial intelligence, and specifically deep learning models, can effectively identify structural changes and enhance the ability to pinpoint the location of damage. However, it's important to note that acquiring data related to scenarios involving damaged structures is not always feasible. Within this Master's Thesis work, a comprehensive overview of Artificial Intelligence is provided, with a specific emphasis on machine learning and deep learning. Furthermore, two deep learning approaches are applied to a test rig featuring a scaled-down four-story building model: a physics-informed autoencoder and a simpler data-driven autoencoder. Subsequently, their performances are compared to conventional methods based on experimental modal analysis. Specifically, modifications to the system are simulated by adjusting the stiffness of the spring. Both of these machine learning algorithms demonstrated their superiority over traditional approaches. Additionally, the physics-informed neural networks exhibited a higher potential for detecting and precisely locating structural damages.

Advisor:
Prof. Simone Cinquemani

Co-advisors:
Ing. Francesco Morgan Bono
Ing. Luca Radicioni
Ing. Claudio Somaschini

Academic year:
2022-2023

Key-words: Artificial Intelligence, Machine Learning, Deep Learning, Structural Health Monitoring, Fault detection, Neural Networks, Convolutional autoencoder, Physic-informed neural network.

1. Introduction

In recent years, there has been a growing interest in the field of Structural Health Monitoring (SHM) for civil structures like buildings and bridges, as evidenced by the references [1–3]. Structures are constantly exposed to various environmental factors that can potentially impact their structural integrity. Some examples of these factors include [4]:

- Structural cracks that can influence the structure's stiffness;
- Changes in balance or weight distribution affecting its mass;
- Wear and loosening in joints altering the boundary conditions for structural dynamics and connections between different sections.

To address these challenges, effective damage detection techniques are crucial. SHM encompasses a range of monitoring strategies that analyze dynamic response measurements, employ feature extraction algorithms, and apply statistical analysis techniques [5].

In a broad sense, damage in structures can be defined as changes that affect their current or future performance. Detecting such damage requires a comparison between two different states of the system, in which one represents the nominal condition, often corresponding to an undamaged state of the structure. Visual inspections are a common method for locating damage. However, they can be imprecise, unreliable, and time-consuming [6]. In contrast, vibration-based techniques have proven to offer a more dependable approach to assessing a structure's health [7–10]. Vibration is considered the most robust indicator of a structure's state compared to other indicators [11].

Given the wealth of data generated by vibration monitoring, deep learning has emerged as a powerful tool. It can identify meaningful features within large datasets using multiple processing layers [12]. Generally, deep-learning models for damage detection rely on supervised learning strategies, where data from both healthy and damaged structural conditions serve as training sets to create functions capable of mapping new input data. However, obtaining the data of the input source that excites the structure can often be prohibitive, leading to issues of robustness and convergence in machine learning techniques [13]. Moreover, collecting data from a damaged state of the structure can be challenging.

To overcome these limitations, Convolutional Autoencoders (CAEs) have been employed to detect damages based solely on raw vibration data from healthy structures [14–16]. Research by Jian et al. [17] demonstrates the usefulness of one-dimensional CNNs for detecting anomalies in bridge vibration signals, while Do et al. [18] showcases the capabilities of autoencoders based on Long Short-Term Memory structures for detecting anomaly vibrations in industrial applications. Finotti et al. [19] assess the structural condition of a viaduct using a sparse autoencoder to learn crucial data features characterizing vibration signals and a support vector machine for damage classification based on these extracted features.

Additionally, the adoption of Physics-Informed Neural Networks (PINNs) allows for the incorporation of physical laws governing the time-dependent dynamics of the structure [13, 20, 21]. Yucesan et al. [22] introduces a novel approach to modeling the fatigue life of wind turbine bearings by blending physics-based and data-driven components into the model. They employ a Recurrent Neural Network (RNN) featuring a physics-informed layer to account for known factors affecting bearing fatigue and a data-driven layer to describe more complex components, resulting in a hybrid model that enhances accuracy and predictive capabilities for real-world bearing fatigue assessments.

The primary motivation for using PINNs in anomaly detection is to mitigate the challenges

of acquiring abnormal data in physical systems and the substantial volume of data required for training neural networks. With PINNs, first principles (physical laws and equations) are integrated with neural networks, narrowing the search space for network parameters and reducing the need for extensive training data. This parameter space compression represents a significant advantage of PINNs over traditional neural network approaches, making them an attractive option for anomaly detection and location [23].

This Master's Thesis work serves as an extension of two research papers that investigate the comparison between an unsupervised deep-learning algorithm and a PINN for structural monitoring, using only vibration data acquired from the healthy state as the training set [24, 25]. Both neural networks are evaluated on a four-story building using acceleration data obtained from accelerometers placed on each floor. The primary objective is to confirm the higher potential of AI methods with respect to conventional methods and to demonstrate the superior capability of PINNs in detecting structural damages compared to conventional unsupervised neural networks.

The organization of this Master's Thesis is as follows:

- Section 2 provides a general overview of Artificial Intelligence (AI) and Machine Learning (ML);
- Section 3 describes the test bench and its mathematical model;
- Section 4 discusses the experimental campaign;
- Section 5 discusses a possibile solution based on control theory;
- Section 6 delves into the ML algorithms used;
- Section 7 presents the results;
- The final section explores conclusions and future trends.

2. Artificial Intelligence and Machine Learning

2.1. Introduction to Artificial Intelligence

Since the main concern of this Master Thesis is the implementation of an AI based algorithm, a general comprehension of the methods we are dealing with is needed and thus, it is the objective of this section. AI is a multidisciplinary field of computer science focused on creating systems and machines capable of performing tasks that typically require human intelligence. These tasks encompass a wide range of activities, including problem-solving, learning, reasoning, perception, understanding natural language, and interacting with the environment. AI systems can be designed to simulate human cognitive functions and adapt to different situations, making them valuable tools for a variety of applications [26].

Artificial intelligence does not comprehends only learning-based approaches, like machine learning and deep learning, but also approaches oriented to replicate the human reasoning process.

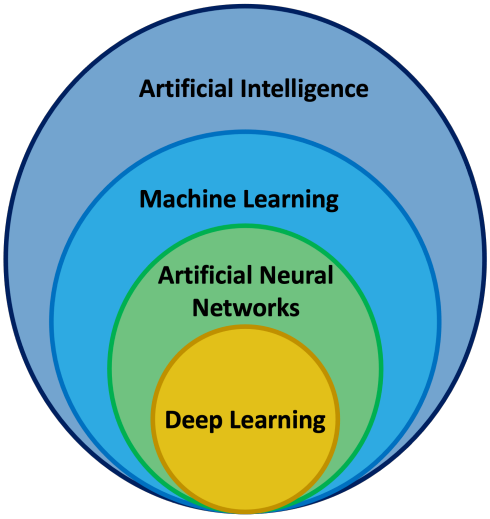


Figure 1: Hierarchical representation of artificial intelligence.

For these reasons, nowadays AI has become an important tool in many aspects of our lives and various industries. Indeed, many are the examples:

- **Automation and Efficiency**. AI technologies can automate repetitive and labor-intensive tasks, leading to higher efficiency and productivity across industries. This allows humans to focus on more creative and complex tasks;
- **Data Analysis**. AI excels at analyzing large amounts of data quickly and accurately. This capability is essential in fields like finance, healthcare, marketing, and scientific research, where data-driven decision-making is crucial;
- **Medical Advancements**. Machine learning algorithms can analyze medical images, predict disease outcomes, and assist healthcare professionals in making informed decisions;
- **Autonomous Systems**. AI powers self-driving cars, drones, and robots with the aim of improving transportation safety and logistics;
- **Scientific Research**. AI assists researchers in analyzing complex scientific data, simulating experiments, and exploring solutions to some of the most challenging problems in fields like physics, chemistry, and climate science;
- **Cybersecurity**. AI helps identify and mitigate cybersecurity threats by detecting anomalies and patterns in network traffic, protecting organizations and individuals from cyberattacks.

In summary, Artificial Intelligence has an high potential to revolutionize industries, improve efficiency, enhance decision-making, and create new opportunities for innovation and advancement. Its impact on society and various domains continues to grow, making it a pivotal field of study and research.

2.2. Historical overview

The development of AI has been marked by periods of optimism, setbacks, and resurgence. Today, AI is a rapidly evolving field with a wide range of practical applications, and it continues to shape the future of technology and society. The concept of AI can be traced back to the 1940s and 1950s when early computer scientists and mathematicians began to explore the possibility

of creating machines that could simulate human intelligence. Alan Turing's work laid the foundation for thinking about machine intelligence and whether machines could exhibit human-like behavior in conversation. However, the term "Artificial Intelligence" was coined at the Dartmouth Workshop in 1956 [27]. In the 1950s and 1960s, AI research was mainly focused on *symbolic AI*, where systems were built using rule-based logic and symbolic representations [28]. Despite early optimism, the field faced significant challenges in achieving its goals, leading to a period known as the "AI Winter." Progress in AI research slowed, and funding became scarce.

During the 1980s, expert systems gained popularity. These systems used knowledge-based approaches to solve specific problems and were widely applied in areas like medicine and finance. In the late 1980s and early 1990s, neural networks and machine learning started to gain attention again. Researchers developed backpropagation algorithms and improved training methods for neural networks, leading to renewed interest in AI.

The 21st century has witnessed significant advancements in AI, driven by breakthroughs in machine learning and deep learning. Large datasets, improved algorithms, and powerful hardware accelerated the development of AI applications in speech recognition, computer vision, natural language processing and robotics. The success of deep learning models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) has revolutionized AI, leading to remarkable achievements in image recognition, language translation, and autonomous vehicles.

2.3. Machine Learning

In classical programming, it is believed that artificial intelligent of human-level can be achieved by programming large set of explicit rules for manipulating knowledge. This approach, known ad *symbolic AI* is suitable to solve well-defined problems and relies on the idea that humans input rules and data to be processed to obtain the required outputs, as shown in Figure 2.

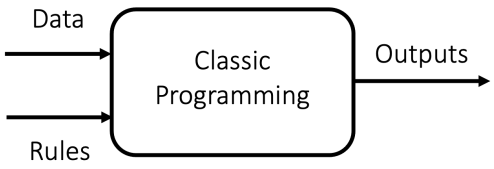


Figure 2: Classical programming paradigm.

However, this paradigm fails in finding explicit rules for complex, fuzzy problems as image classification or speech recognition. For this reason, Machine Learning (ML) was introduced. ML is a sub-method of artificial intelligence that aims to teach computers how to learn from data and make predictions or decisions on future scenarios. Is said that ML algorithms, to perform a task, are not programmed but *trained*. Indeed, the system through a new additional phase, called training, is able to construct the knowledge about the case under study.

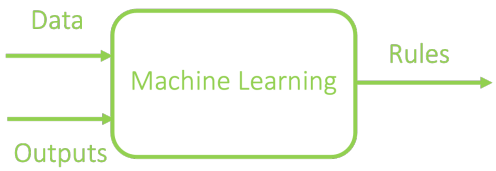


Figure 3: Machine learning programming paradigm.

Structured or unstructured data become a key point. Indeed, they are used for both training and testing ML algorithms. Moreover, ML algorithms are strictly related to statistics. Indeed, they are mathematical models that analyze data to recognize patterns, relationships, and trends especially. However, unlike statistic, ML well performs when dealing with large and complex dataset and the parameters and outputs are not clearly connected, situations in which the research for analytical solution is unpractical or really time consuming. In a more precise way, machine learning models try to transform data in more meaningfully *representations* for the given task. However, these representation are searched in a restricted space of predefined operation, called *hypothesis space*. To summarize, a ML algorithm to work properly needs:

- Input data;
- Examples of the expected outputs;
- A measure of how the algorithm is performing, in order to measure the distance the algorithm's output and the expected one. This index works as a feedback signal, helping the algorithm to adjust itself through the *learning* step.

It must be said that the ability to work with large dataset through the training phase represent also one of the biggest limitation of ML. Indeed, it is impossible to extract generalized rules of data analysis based from information acquired and elaborated in the past. If the input dataset changes a re-training would be necessary.

2.4. Different machine learning problems

Focusing only on the ML framework, it is possible to distinguish different methodologies based on the composition of the dataset. In particular, four classes of machine learning are identified:

- **Supervised learning**, each input is fed to the algorithm with the corresponding target output, called *label*. The goal is to predict the label of objects that the machine has never seen, based on the knowledge acquired during the training phase;
- **Unsupervised learning**, as the name suggest any label is fed to the machine for the data-points. The algorithm has to find meaningful patterns by itself through statistic. It is obvious that the solution is not unique. *Dimensionality reduction* and *clustering* are common in this class of ML;
- **Self-supervised learning**. The learning phase doesn't involve any label generated by humans. Labels are still used but they are generated from the input data with a heuristic algorithm. These problems are really useful when for some conditions data are not accessible. In particular, as already said, data about failure condition of structures are difficult to be acquired. Instead, data regarding the nominal condition are available. *Autoencoders*, which are discussed in a more specific manner in Section 6.3, are important examples of self-supervised learning, where the targets are the unmodified inputs;
- **Reinforcement learning**, suitable for all problems for which the correct answer is not known. In those cases, a system of rewards and penalties is implemented and the machine modifies is knowledge trying to maximize the reward.

2.5. Deep learning

In recent years, *Deep Learning* (DL), a sub-field of machine learning, has garnered significant attention due to its remarkable achievements in various domains. On the contrary of what could

be thought, in deep learning, the therm "deep" doesn't refers to a class of ML-algorithms with higher capability of understanding but to a class of artificial neural networks working with several hidden layers of increasingly meaningful representations. Indeed, while the other approaches focus on learning by means one or two layers of representations, and for this reason called *shallow methods*, deep learning relies on tens or hundreds successive layers of data elaboration.

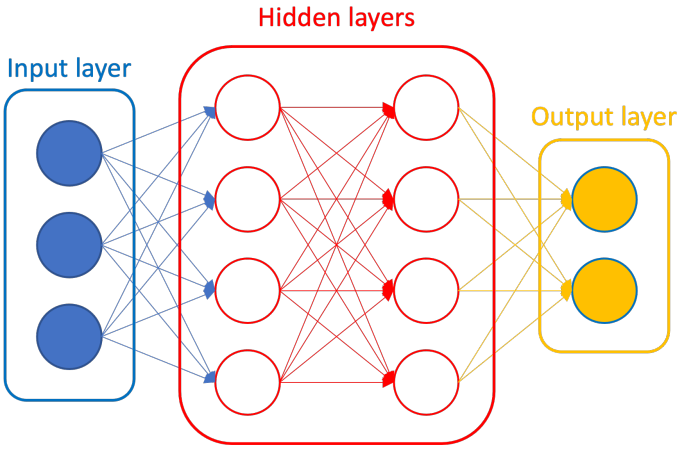


Figure 4: Graphical representation of an artificial neural network.

Being inspired by the human brain's structure and function, each layer used by deep learning algorithms is composed by neurons, linked to all neurons present in the previous and successive layers. Deep learning networks can be thought as multistage information filtering with the objective to map inputs into targets. Each layer, which acts like a filter, is characterized by a weight. These weights are the way in which the algorithm improve itself to correctly associate inputs to targets during the learning phase. The first layer of an *Artificial Neural Network* (ANN) is labeled as *input layer*. The dimension of this layer matches the dimension of the data entering the network. Instead, the last layer of an ANN, labeled as *output layer*, has the dimension of what it is expected from the layer (i.e. the fixed targets). These deep architectures enable the hierarchical extraction of features from raw data. In image processing, for instance, lower layers may detect basic features like edges and corners, while higher layers can recognize complex structures like faces or objects. This hierarchy of features allows deep learning models to capture intricate patterns in data, making them highly effective in tasks such as image recognition and speech recognition. The training phase of deep learning models involves an algorithm called backpropagation, where the model adjusts its internal parameters (weights and biases) to minimize the difference between predicted and actual outputs. This iterative optimization process relies on a mathematical technique known as gradient descent. Both the backpropagation algorithm and the gradient descent are addressed in the following section[29, 30].

Despite its numerous successes, it must be said that deep learning is not without challenges. Indeed, training deep learning models can be computationally intensive and may require large datasets to generalize well. Moreover, one of the key challenges is interpretability, as deep models are often regarded as "black boxes", making it difficult to understand their decision-making processes.

2.6. Input and output data

At this point, it should be clear how important the dataset is for the learning phase of the ML machines. To make predictions, large amount of data-points are required which should contain

relevant information. For this reason the dataset is said to be *considerable* and *significant*. In general, almost the whole ML algorithms use *tensors* as basic data structure [28]. Tensors are a generalization of matrices to an arbitrary number of dimension, also referred as *axis*. For example, an array of numbers (vector) is a 1D tensor and matrices are 2D tensors. Tensors must be always defined by three features:

- Number of axes;
- Shape, which is related to the number of dimensions along each axis;
- Data type, which classifies the type of the data contained in the tensor.

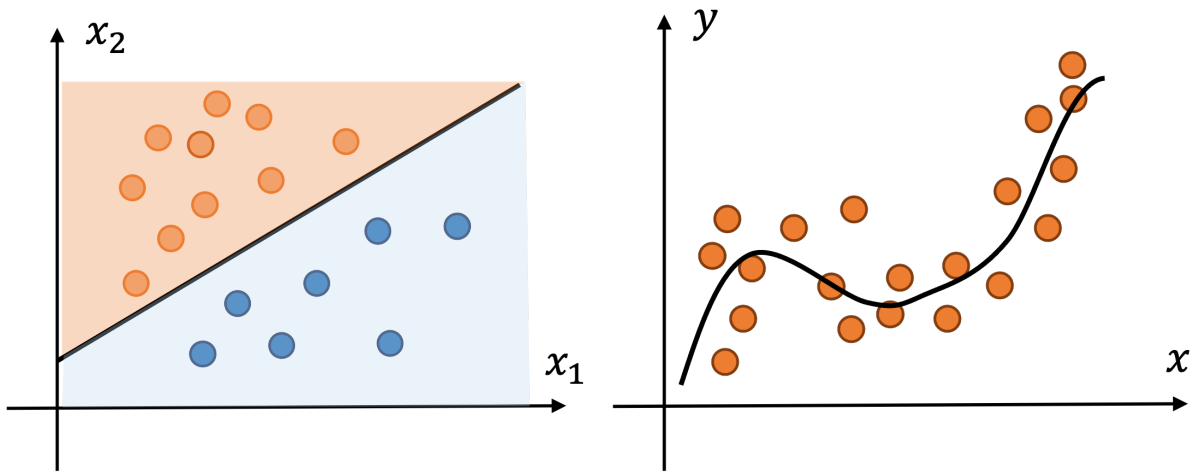
The input to a machine learning algorithms is also named observation x. Thus, the input dataset $\mathcal{D}$ is made up by n observations, which are also characterized by some m peculiar traits named *features*. All the features considered in a ML problem define a m dimensional *feature space* $\mathcal{X}$.

On the other hand, the expected outputs y were introduced as *labels* in the previous sections. Indeed, given the all the observations lying in the *feature space* $\mathcal{X}$, the objective is to predict the corresponding outputs of the $\mathcal{Y}$ set. For example, in *supervised learning* all observations are fed to the algorithm with their corresponding label and the dataset is defined as:

$$\mathcal{D} = \{(x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n)\}$$

To provide a label to a new pair (x,y), a *classifier* function is needed $f : \mathcal{X} \to \mathcal{Y}$. Moreover, according to the content of the label itself is possible to distinguish:

- **Data classification**. In this case, for each data-point there is only one possible output, searched within a fixed space of categories;
- **Data regression**. In this case, even if the output is a single valued variable, it can be influenced by many factors which must be taken into account in the data classification process.



(a) Classification. (b) Regression.

Figure 5: Examples of data classification and regression.

2.7. Data preprocessing

Before starting the training phase of a ML model, it must be ensured that data are fed in a feasible format. To do that, the dataset undergoes a preprocessing phase. *Preprocessing* methods are

used to transform raw data in the desired form. For main classes of data preprocessing techniques are identified [28]:

- **Data Vectorization**. It aims to transform all inputs e targets in tensors of floating-point data;
- **Value Normalization**. In general, it should be avoided to fed into the neural network data with relatively large values or heterogeneous. Indeed, in these cases gradient could not converge. For this reason, a normalization step is usually taken into account to end up with tensors of floating-point values in the 0-1 range;
- **Missing-data**. Incomplete dataset are common for situation in which the data recording is cost intensive, but missing data-point have significant impact on the performances of the model, for this reason many methods have been implemented to come with this problem [31];
- **Feature Extraction.** To decrease the computational cost of the algorithm is possible to reduce the dimension of the feature space, removing all the irrelevant or redundant features present in the dataset [32].

2.8. Practical implementation of a neural network

As already said, the main ingredient of a successful neural network are the layers, that can be thought as filter for data, able to extract representation that, hopefully, would be more meaningful for the problem being studied. So, a deep learning machine is like a series of increasingly finer data filters. However, to properly train a ML algorithm three more elements are needed:

- **Loss function** (also objective function), which represents the quantity to be minimized during the training phase and which defines the feedback signal;
- **Optimizer**, which represents the method chosen for the updating of the network based on the feedback signal coming from the loss function;
- **Metrics for training and testing.**

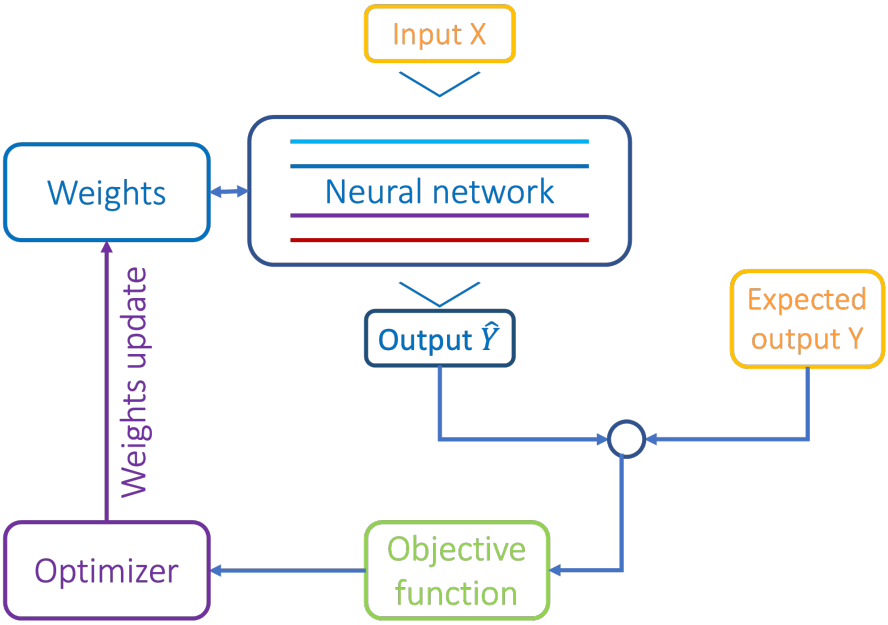


Figure 6: Typical block-diagram of a neural network.

Loss function

The *loss function*, $\mathcal{L} : \mathcal{X} \to \mathcal{Y}$, quantifies how far the predicted label $\hat{y} = f(x)$ is from the true label y [33]. Even if quite similar, *loss function* should not be confused with *cost function*. Indeed, the former is used to optimize the model during training while the latter aggregates the loss values over the entire training set and guides the optimization process.

Optimizer

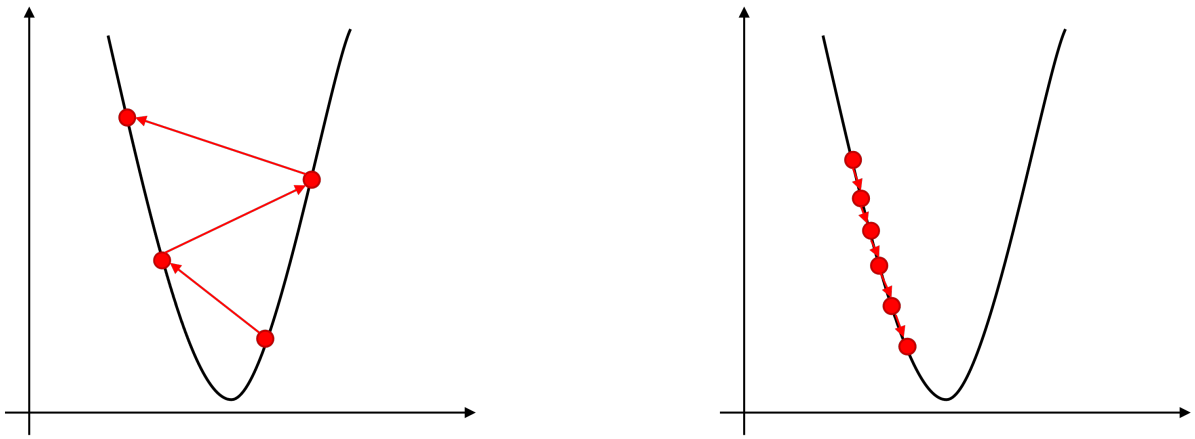
Optimization method have to be taken into account to improve the performances of learning models. These optimization methods rely on optimization algorithms or optimizers which highly affect the accuracy and training speed of the model. During the training phase the optimizer model, modify the weights of each layer and minimize the loss function with the objective of reducing the overall loss and improving accuracy. The choice of the correct optimization model is crucial, indeed, especially for deep learning models, choose the right weights can be a challenging task, as a deep learning model generally consists of millions of parameters. Among the several option available today, the most commonly used are listed below:

- **Gradient descent** is a method based on a convex function, which update its parameters iteratively with the aim of minimizing a given function to its minimum. A gradient is a measures of how much the output of a function changes in relation to changes in the inputs. Thus, in machine learning context it can be thought as an index of the change in all weights with respect to the change in error. You can also think of a gradient as the slope of a function. In mathematical terms, a gradient is the partial derivative of a function with respect to its inputs. As the name suggest, the weights are updated in the direction of the steepest gradient and in a descending versus.

$$w_i = w_{i-1} - \gamma \cdot \nabla f(w_{i-1})$$

  where i represent the current iteration, γ is the learning rate, a parameter regulated by the optimizer and f is the objective function. The length of the steps into the direction of the local minimum is determined by the *learning rate*, which figures out how fast or slow

we will move towards the optimal weights. However, the length of the steps is a crucial parameter to be set. Indeed, if the steps taken are too big, this method may not end up reaching the local minimum because it bounces back and forth in the convex function. On the contrary, setting the learning rate to a very small value, gradient descent will eventually reach the local minimum but in a longer time span.



(a) High learning rate.

(b) Low learning rate.

Figure 7: Effect of different learning rate for the gradient descent method.

Among all the methods which use the gradient descent:

- **Batch gradient descent** firstly calculates the error for each sample within the training dataset and only after updates the model. This whole process happens in a time span which is called *epoch*. Among all the advantages of batch gradient descent the computational efficiency is the most important. Usually, it produces a stable error gradient and convergence. However, this stable error gradient can sometimes result in a model which isn't the best achievable. Moreover, it requires the entire training dataset to be in memory and available to the algorithm.

- **Stochastic gradient descent** (SGD) updates the parameters for each training sample one by one. Depending on the problem, this can make SGD faster than batch gradient descent. Moreover, the frequent updates allow to have a detailed rate of improvement. However, it is clear that those frequent updates are more computationally expensive.

- **Mini-batch gradient descent** is a combination of the concepts of SGD and batch gradient descent. It splits the training dataset into small batches and performs an update for each of those batches. This results in a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. This is the most common type of gradient descent within deep learning.

- **Adam optimizer**, short for Adaptive Moment Estimation optimizer, is a commonly employed tool in the realm of deep learning. It serves as an extension to the Stochastic Gradient Descent (SGD) algorithm, primarily responsible for updating a neural network's weights throughout the training process. Unlike SGD, which maintains a consistent learning rate from start to finish, the Adam optimizer dynamically calculates individual learning rates by considering past gradients and their second moments. This adaptive feature contributes to swifter convergence and enhances the overall performance of the neural network. The Adam optimizer has several advantages. It is notably simple to implement, exhibits faster execution times, places minimal demands on memory resources, and necessitates less

fine-tuning compared to alternative optimization algorithms. However, it leans more towards optimizing convergence speed and may, in some instances, generalize the data-points slightly less effectively than SGD.

Metrics

Performance metrics are an essential element of every machine learning model. They allow to quantify and study the performances of the algorithm. What is important to highlights is that metrics are different from loss functions. Loss functions show a measure of model performance whilst metrics are used to monitor and measure the performance of a model during training and testing. As already said, machine learning tasks are divided in Regression or Classification, in the same way also the performance metrics. In this Master 's Thesis work only the regression metrics are of interest, for this reason classification metrics will not be addressed. The output of regression models is continuous, for this reason the distance between the predicted output and the real output must be evaluated. Among the several possibilities of regression metrics, the main used are:

- **Mean Square Error** (MSE) evaluates the average of the squared difference between the target value and the value predicted by the regression model.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2$$

  Due its nature, it penalizes even small errors by squaring them, which essentially leads to an overestimation of how bad the model is. For this reason, when interpreting the error the squaring factor must be kept in mind.

- **Mean Absolute Error** (MAE) evaluates the average of the difference between the real values and the predicted values.

$$MAE = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y}_j|$$

  On the contrary of MSE, since the absence of the squaring factor, it tends to be more robust towards outliers. It gives us a measure of how far the predictions were from the actual output. However, the absolute value of the residual is not able to show the direction of the error, i.e. whether we're under-predicting or over-predicting the data.

- **Root Mean Squared Error** is evaluated taking the the square root of the average of the squared difference between the target value and the value predicted. Basically, it represent the square root of the MSE.

$$MSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2}$$

Backpropagation algorithm

As it has been pointed out, typically weights of the layers are updated following the direction the gradient in a descending way, according to the type of optimizer chosen. However, this updating process would be really slow and complicated also for small neural networks. To overcome this limitation, the new set of weights is computed through an algorithm called *backpropagation*. The basic idea behind this algorithm is to start from a random set of weights, collect the outputs

of the model for the given inputs and compute the loss. Once the final loss value has been computed, *backpropagation* works backward from the top layers to the bottom layers to compute the contribution of each parameter to the loss. Doing that it make extensive use of the *chain rule*.

2.9. Under-fitting and Over-fitting

In Section 2.3 it has been highlighted how machine learning is related to statistics. For this reason, ML algorithms can be affected by two common problems: *under-fitting* and *over-fitting*.

Under-fitting

Under-fitting can occur when:

- The model is characterized by an insufficient number of parameters;
- It is difficult to find the global optimum of the objective function on the training set, i.e. getting stuck at local minimum;
- There are limitations on the computation resources (not enough training iterations of an iterative optimization procedure)

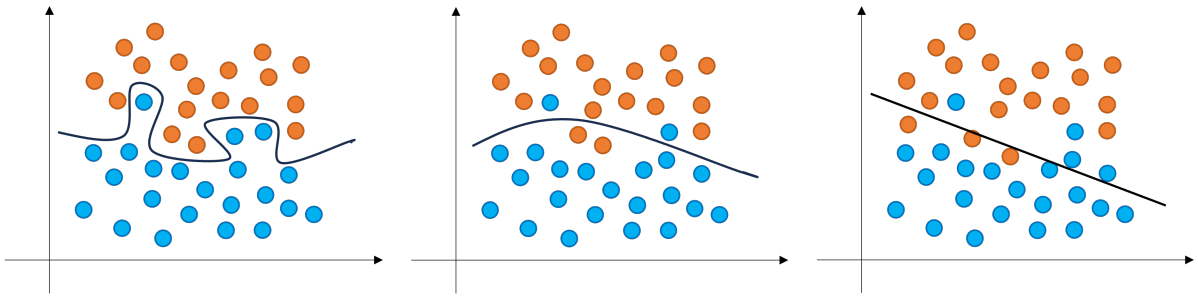
In all of these situations the learner cannot capture some important aspects of the data, therefore not finding a solution that fits the training set in the correct manner. For example, an under-fitting situation could be if the learner uses a linear regression for the pairs $(x_i^{train}, y_i^{train})$ even if y_i^{train} is a quadratic function of x_i^{train}.

Over-fitting

Over-fit can occur when:

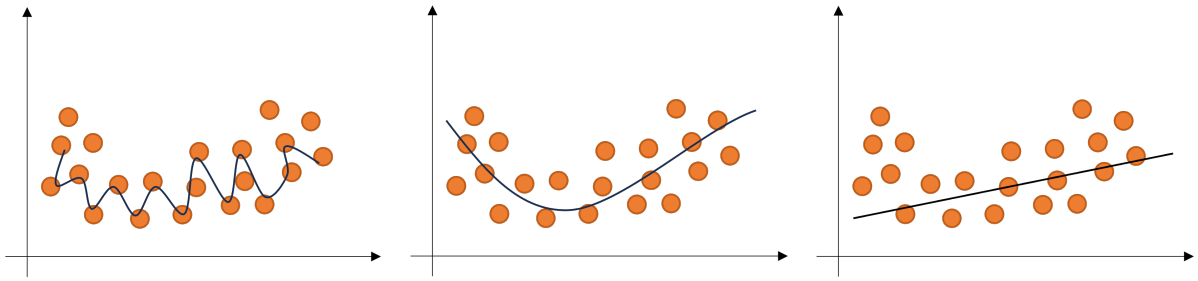
- The model feature set is too large compared with the size of the training data;
- The data at disposal are not sufficient;

In these cases, the learner fits the training set in a very good way but loses the ability of generalization, i.e. small training error but larger generalization error. As results, the performances with an unknown dataset will drop drastically.



(a) Over-fitting. (b) Right-fitting. (c) Under-fitting.

Figure 8: Examples of over-fitting, right-fitting and under-fitting for data classification.



(a) Over-fitting. (b) Right-fitting. (c) Under-fitting.

Figure 9: Examples of over-fitting, right-fitting and under-fitting for data regression.

At this point, could be useful to define the characteristics of *capacity* of a ML model. Informally, a model's capacity is its ability to fit a wide variety of functions. A model with higher capacity can model several different types of functions and so be able to learn a function which sufficiently map inputs to outputs in the training dataset. However, a model with too much capacity may memorize the training dataset and its associated noise and fail to generalize. The term "generalization" refers to the capability of the model to adapt and properly react to unseen and new data, but coming from the same distribution as the one used to train the model. It's clear that a trade-off between generalization and capacity must be found to obtain the correct solution from a ML algorithm.

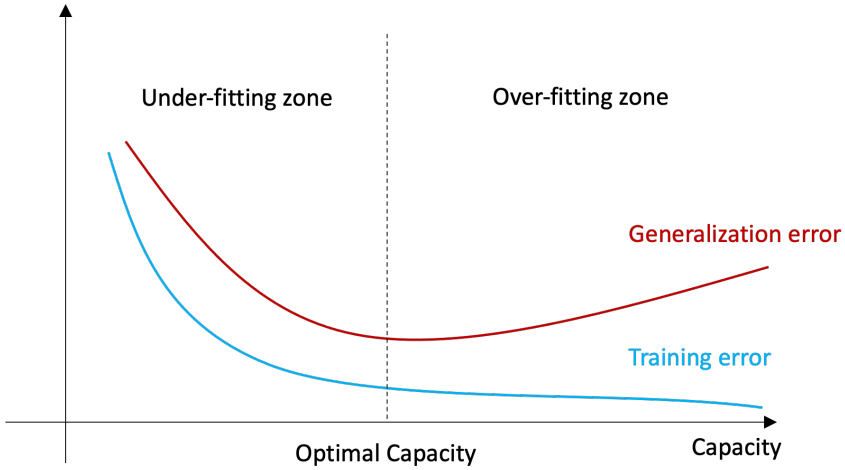


Figure 10: Generalization and training errors as function of the capacity of the model.

As shown in Figure 10, as model's capacity increases, training error is reduced, but the difference between training and generalization error increases. At some point, the increase in this difference is larger than the decrease in training error (typically when the training error is low and cannot go much lower), and we enter the over-fitting region, where capacity is too large, above the optimal capacity [12].

2.10. Model validation

The main goal of a machine learning model is to generalize, in other words to perform in a good way on unseen data. To achieve that, typically the initial dataset is split into three subsets: training, validation and test sets. As the name suggest, they will be used for three different

phases: training, validation and test. Splitting the dataset allow the model to be evaluated on different data with respect to the one used for training, trying to avoid the model to begin to *over-fit*. With *validation* is intended the process for assessing the performances of the ML model. In principle one can think that only training and test phases are needed to obtain a performing model. However, the configuration of the model needs to be tuned, i.e. number of layers or size of layers. Indeed, as it has been pointed out over-fitting and under-fitting problems are very common after the training process. For this reason, a previously separated portion of the dataset is used to check how the model would perform for real-world predictions. There are several validation techniques, the most used are listed below [28]:

- **Holdout Validation** is the simplest validation method. The available dataset is split into training and validation sets. The model, trained on the training set, is then applied to the validation set and the loss is evaluated. It must be pointed out that the result of the validation process are strictly dependent on the way the dataset have been divided;

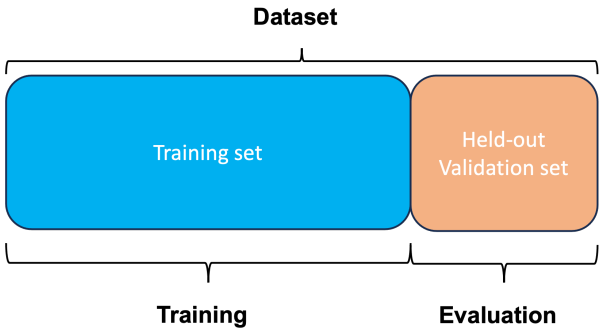


Figure 11: Holdout validation split.

- **Cross Validation** represent a more robust alternative to holdout validation. It performs well in all the cases in which the dataset is limited. In this case, the model is trained and validated on different set of the initial dataset at each iteration. Then, the overall loss is evaluated as average of the single losses;
- **Iterate Cross Validation with shuffling** is an extension of the previous method, used when the dataset is extremely limited. In this case, a cross-validation methods is applied; then, the dataset is shuffled and the same cross-validation method is applied. At the end an averaging procedure is done taking into account each loss obtained. It's clear how this method requires huge amount of resources. Indeed, the model must be trained several times.

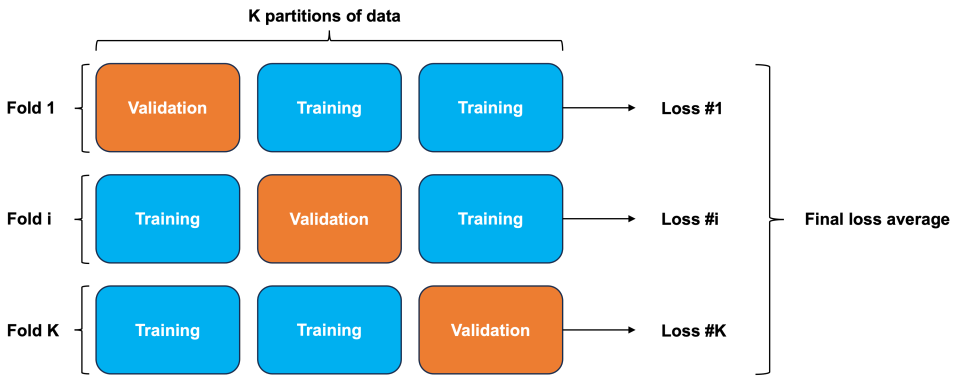
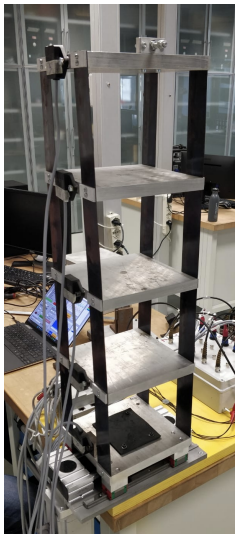
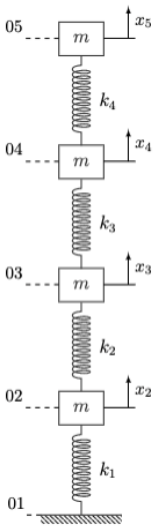


Figure 12: Cross validation split.

3. System description



(a) A photo of the real system.



(b) Lumped mass model of the system.

Figure 13: Real system and lumped mass model.

The system object of this work is a multi-storey building shown in Figure 13. Five aluminum plates, connected by steel laminas, respectively model the storeys and the pillars of the building; the physical data of the system are reported in Table 1. [34]

Table 1: Data of the system.

Storey	
Area	$200 \times 200\ mm^2$
Thickness	$20\ mm$
Mass	$2.26\ kg$
Pillars	
Area	$0.5 \times 50\ mm^2$
Length	$180\ mm$
Thickness	negligible
Mass	negligible

3.1. Mathematical model

The reason behind using a lumped mass approach in modeling the system is the substantial difference in mass between each storey and the laminas. This approach simplifies the system into four degrees of freedom, represented by four masses connected by springs, as depicted in Figure 13 (b). In the upcoming sections, we will introduce a physics-informed neural network (PINN). PINNs stand out due to their custom loss function, which incorporates information

related to the governing physical laws of the system. Therefore, deriving the equations of motion that describe the dynamics of the tested building, as presented in equation 1, becomes a critical aspect of training the machine learning algorithm we are implementing.

$$[M]\ \underline{\ddot{x}} + [C]\ \underline{\dot{x}} + [K]\ \underline{x} = \underline{0} \tag{1}$$

In particular, the column vector $\underline{x}$ represents the absolute horizontal displacements of the storeys:

$$\underline{x} = [x_1\ x_2\ x_3\ x_4]^T \tag{2}$$

The mass matrix of the model is diagonal:

$$[M] = \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & m \end{bmatrix} \tag{3}$$

While it's possible that future studies may relax this assumption, by including displacement and degree of rotation as boundary conditions, in this Master's Thesis the hypothesis of a clamped-clamped beam is used to calculate the stiffness matrix, as outlined in [35]. Specifically, since one end experiences transversal displacement, the stiffness of the equivalent spring can be determined as follows:

$$k_{eq} = 4 \cdot k = 4 \cdot \frac{12EJ}{L^3} \tag{4}$$

Furthermore, it's essential to consider the influence of the weight of each storey on the transversal stiffness, as emphasized in [36]. To empirically validate this point, Section 4 will investigate the inverted building. In this regard, the term $T = m \cdot g/L$ is taken into consideration, and the resulting stiffness matrix is presented as follows:

$$[K] = \begin{bmatrix} 2k_{eq} - 7T & -k_{eq} + 3T & 0 & 0 \\ -k_{eq} + 3T & 2k_{eq} - 5T & -k_{eq} + 2T & 0 \\ 0 & -k_{eq} + 2T & 2k_{eq} - 3T & -k_{eq} + T \\ 0 & 0 & -k_{eq} + T & k_{eq} - T \end{bmatrix} \tag{5}$$

Regarding damping, the analysis incorporates Rayleigh's damping model, which permits the representation of the damping matrix as a linear combination of both the mass matrix and the stiffness matrix, as outlined in [37]. This relationship is expressed as follows:

$$[C] = \alpha \cdot [M] + \beta \cdot [K] \tag{6}$$

The coefficients α and β are determined through a process of least square minimization. When modal coordinates are used, the mass, stiffness, and damping matrices become diagonal matrices, allowing us to express for each vibration mode i within a specific set:

$$c_i = \alpha \cdot m_i + \beta \cdot k_i \Rightarrow \xi_i = \frac{c_i}{2m_i\omega_i} = \frac{\alpha}{2\omega_i} + \frac{\beta\omega_i}{2} \tag{7}$$

In this context, ξ_i represents the modal damping, and ω_i stands for the natural frequency linked to the i-th vibration mode. Furthermore, typically, ξ_i and ω_i are determined by examining the structural vibration response through modal analysis, which leads to the formulation of the following over-determined system of equations:

$$\begin{bmatrix} \frac{1}{2\omega_1} & \frac{\omega_1}{2} \\ \dots & \dots \\ \frac{1}{2\omega_i} & \frac{\omega_i}{2} \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \xi_1 \\ \dots \\ \xi_i \end{bmatrix} \tag{8}$$

which is then resolved through a least square minimization process. In this thesis, during the experimental investigation discussed in subsequent sections, we determine the modal damping ξ_i and the natural frequency ω_i for the four vibration modes exhibited by the structure. The resulting coefficients for the Rayleigh's damping model are provided in Table 2.

It's worth highlighting that the methodology outlined in this study can be expanded to encompass more intricate continuous structures, such as bridges. In such cases, the model can be approximated by either estimating and incorporating the modal parameters of the first N vibration modes using experimental modal analysis, as detailed in [35], or by developing a finite element model.

Table 2: Results of the least square minimization process.

α	0.03
β	0.028

4. Experimental campaign

To identify structural damage, both conventional methods and machine learning algorithms depend on detecting alterations in the structural behavior. The techniques explored in this thesis employ variations in vibration measurements between the structure in its standard state, referred to as "healthy," and a state with "damage" as an indicator of potential damage.

Hence, the experimental campaign carried out on the tested structure is designed to collect raw data for both the "healthy" and "damaged" states. The experimental setup for both scenarios includes:

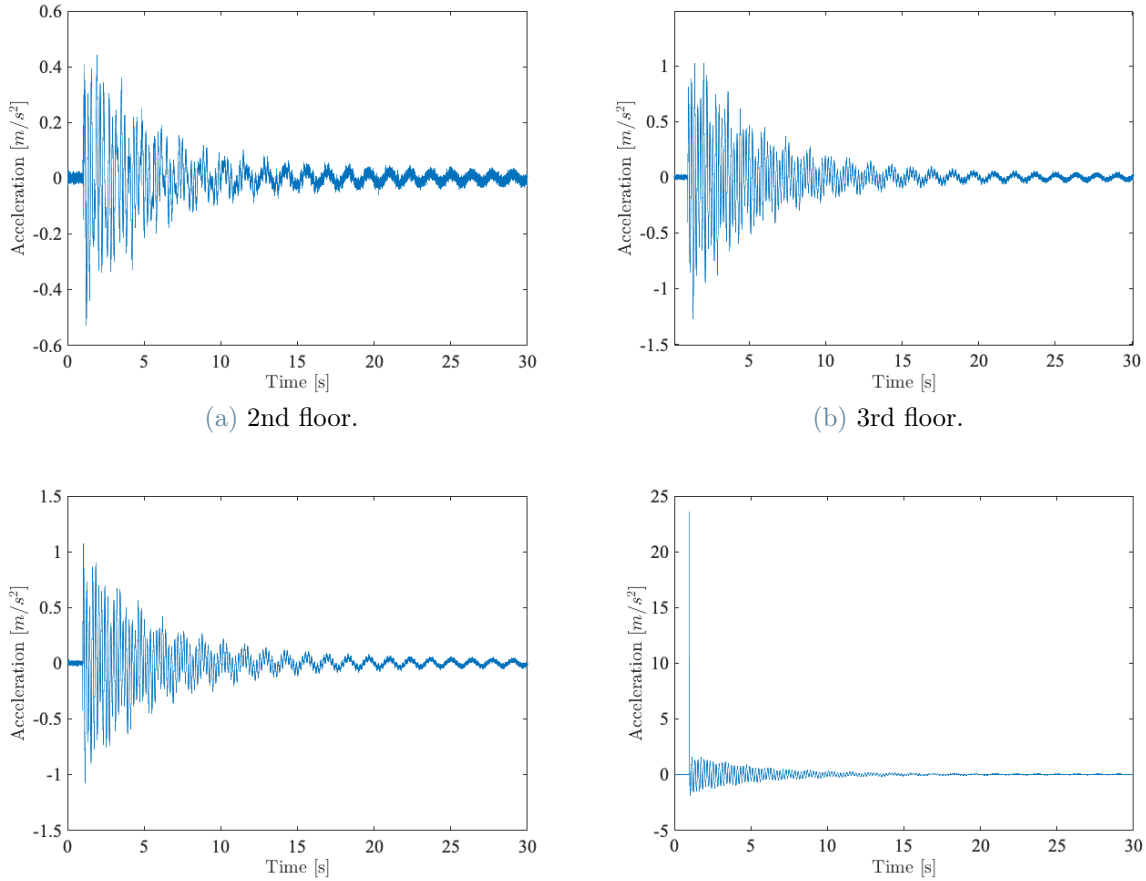
- four TE triaxial capacitive MEMS accelerometers, one per each storey;
- a PCB Piezotronics impact hammer;
- a National Instruments c-DAQ.

The structure is stimulated using an impact hammer, and the transverse vibrations are captured. In the "healthy" scenario, a total of 1000 data records, each lasting 70 seconds and sampled at a rate of 128Hz, are recorded. An illustration of the responses, specifically for the initial 30 seconds, is displayed in Figure 14.

Since the vibration signals are initially recorded in Volts, they require preprocessing before undergoing analysis. This preprocessing involves accounting for the sensitivity of each accelerometer and the impact hammer, thereby correctly restoring the measurements to their respective units, such as m/s^2 and N.

Additionally, it's crucial to recognize that the measured data are digitized signals acquired within a finite time window. Consequently, they may be susceptible to various issues, including (i) noise, (ii) aliasing due to the sampling process, and (iii) leakage, owing to the finite acquisition window. To address these challenges, an averaging procedure is employed. Two distinct estimators, as outlined in equation 9, are employed to assess the Frequency Response Function (FRF) for inertance (acceleration/force) for each accelerometer.

$$\begin{aligned} H_1(f) &= \frac{X(f)}{F(f)} = \frac{G_{XF}(f)}{G_{FF}(f)} \\ H_2(f) &= \frac{X(f)}{F(f)} = \frac{G_{XX}(f)}{G_{FX}(f)} \end{aligned} \tag{9}$$



(a) 2nd floor. (b) 3rd floor.

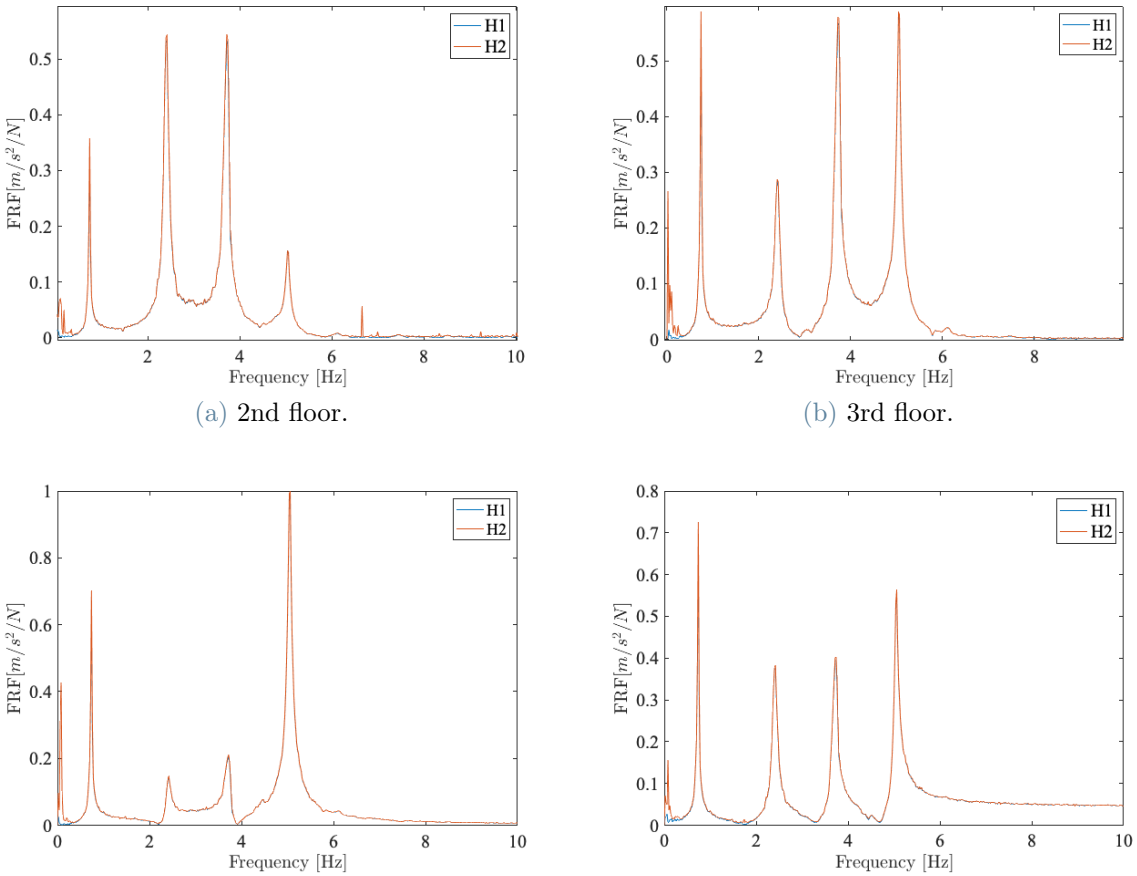
(c) 4th floor. (d) 5th floor.

Figure 14: Acceleration of each floor with input applied on 5th floor.

In this context, F and X represent the Fourier transforms of the input force and the output vibration, while G_{FF} and G_{XX} denote the estimated auto spectra. Additionally, G_{XF} and G_{FX} represent the estimated cross spectra, which are typically computed as follows:

$$\begin{aligned} G_{XX}(f) &= \frac{1}{N}\sum_{j=1}^{N} X_j^*(f)\cdot X_j(f) \\ G_{XY}(f) &= \frac{1}{N}\sum_{j=1}^{N} X_j^*(f)\cdot Y_j(f) \end{aligned} \tag{10}$$

where X and Y are two general sampled signals and N is their length. Figure 15 shows an example of FRFs for the case in which the structure was excited by an input force acting on the 5th floor. After calculating the Frequency Response Functions (FRFs) for each accelerometer, the natural frequencies and mode shapes are deduced using the Experimental Modal Analysis (EMA) technique, as described in references [38] and [39]. Specifically, the natural frequencies of the system are presented in Table 3 for both the numerical and experimental models. Additionally, the mode shapes are illustrated in Figure 16.



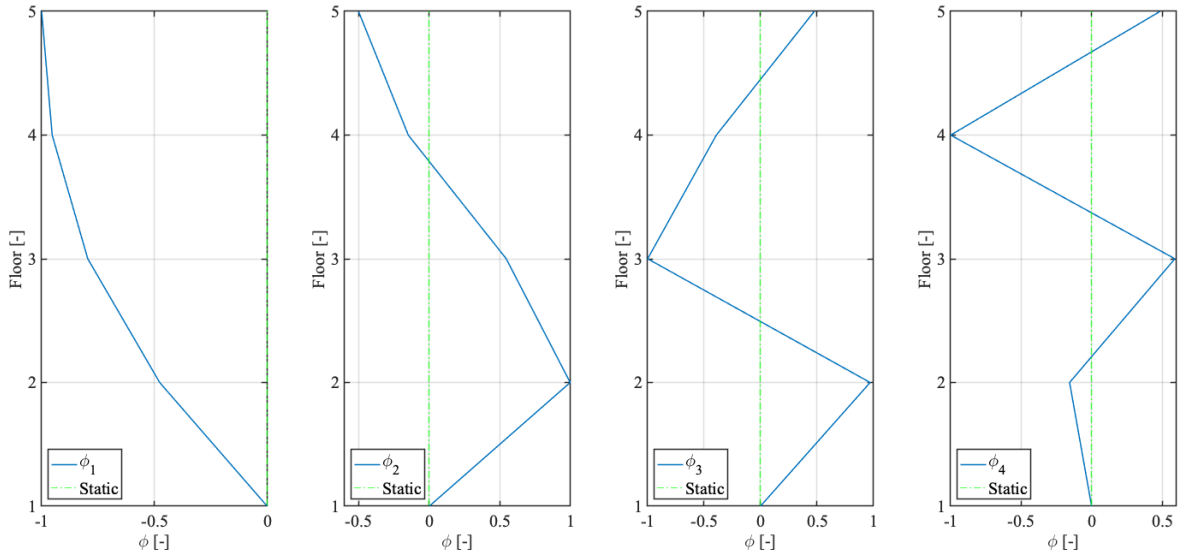
(a) 2nd floor. (b) 3rd floor.

(c) 4th floor. (d) 5th floor.

Figure 15: FRFs evaluated for each accelerometer, one for each floor, with input applied on 5th floor.

Table 3: Natural frequencies for both the numerical and the experimental model.

Mode	Numerical model [Hz]	Experimental model [Hz]
1	0.79	0.75
2	2.51	2.41
3	3.88	3.74
4	5.01	5.04



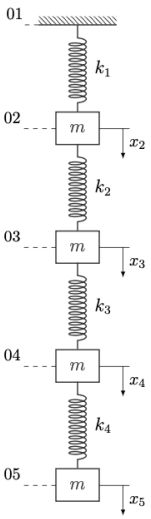
(a) 1st mode. (b) 2nd mode. (c) 3rd mode. (d) 4th mode.

Figure 16: Vibration modes of structure for the healthy scenario.

It's crucial to emphasize that the natural frequencies and mode shapes obtained for the "healthy" structure, as presented earlier, will serve as a reference for the vibration-based damage detection method. This method, in turn, will be used as a benchmark to evaluate the performance of the two machine learning algorithms.

To empirically validate the hypotheses proposed in Section 3 concerning the impact of weight on stiffness, another test was conducted. In this test, the structure was placed in an inverted configuration, without the addition of any extra mass, as depicted in Figure 17.

(a) A photo of the flipped real system.



(b) Lumped mass model of the system.

Figure 17: Flipped real system and lumped mass model.

Moreover, the contribution of this tension beam effect on the modal stiffness of the first four modes of vibration was investigated through the influence coefficient α_T, evaluated as follow:

$$\alpha_T = \frac{k_T}{k_{fl}}$$

where:

- k_T is evaluated through the difference between the modal stiffness of the system in the default configuration and the one in the flipped configuration;
- k_{fl} is the structural part of the modal stiffness in the flipped configuration: $k_{fl} = k_i - k_T$.

The values of α_T are reported in Table 4.

Table 4: Flipped system: stiffness and influence coefficient.

		Mode of vibration			
		1st	2nd	3rd	4th
$\omega_{i,def}$	[Hz]	0,75	2,41	3,72	5,06
$\omega_{i,flip}$	[Hz]	1,47	3,84	5,88	7,34
k_T	[N/m]	71	399	925	1261
k_s	[N/m]	122	9171	2160	3546
$k_{i,def}$	[N/m]	51	518	1235	2284
$k_{i,flip}$	[N/m]	193	1316	3085	4807
α_T	[-]	0,59	0,43	0,43	0,36

The resulting natural frequencies are reported in Table 5.

Table 5: Natural frequencies for both the numerical and the experimental model.

Mode	Numerical model [Hz]	Experimental model [Hz]
1	1,39	1,46
2	3,72	3,84
3	5,48	5,88
4	6,75	7,32

As is possibile to notice, the effect of the weight on the transversal stiffness of the system must be taken into account. Indeed, this effect is not negligible.

When examining scenarios involving structural damage, it's essential to bear in mind that internal structural issues are typically not caused by material loss and, consequently, do not result in changes in mass. Instead, they often stem from alterations in geometry or material properties that impact one or more elements within the stiffness matrix, as discussed in [40]. Therefore, in the "damaged" condition, the time histories are acquired by altering only the stiffness values of the laminas. Specifically, six distinct sets of laminas, all featuring the same cross-sectional dimensions but varying in length as indicated in Table 6, are employed to reduce the stiffness of the springs connecting consecutive floors. These adjustments result in stiffness values ranging from 10% to 60% of the nominal value.

Table 6: Lengths of the set of laminas used to reproduce a damage in the structure.

Damage Percentage	Length
0%	180.0 mm
−10%	186.5 mm
−20%	194.0 mm
−30%	203.0 mm
−40%	213.5 mm
−50%	227.0 mm
−60%	244.0 mm

In total, a dataset comprising 240 time records, each lasting 70 seconds and sampled at a rate of 128Hz, is collected. This dataset encompasses 10 records for every possible combination of damage extent (represented by the type of lamina) and damage location (across four floors).

5. Pole Placement for SHM and model updating

In control theory *Pole Placement* technique allows to change the position of the closed-loop poles of the system with respect to the open-loop ones [41]. Considering a Linear Time Invariant (LTI) system in state-space representation:

$$\dot{\vec{X}} = [A]\vec{X} + [B]\vec{u} \tag{11}$$

the Pole Placement method determines the feedback gain matrix such that the closed-loop poles are placed at desired location, applying a full state feedback to system motion:

$$\vec{u} = [K](\vec{X}_{ref} - \vec{X}) \tag{12}$$

where the matrix $[K]$ represents the feedback gain matrix and $\vec{X}_{ref}$ the reference motion. Writing the equation of motion of the controlled system:

$$\begin{aligned} \dot{\vec{X}} &= [A]\vec{X} + [B][K](\vec{X}_{ref} - \vec{X}) && (13)\\ &= ([A] - [B][K])\vec{X} + [B][K]\vec{X}_{ref} && (14)\\ &= [A_c]\vec{X} + [B_c]\vec{X}_{ref} && (15) \end{aligned}$$

where $[A_c] = [A] - [B][K]$ is the state matrix of the feedback control system. The eigenvalues of the open-loop system are strictly related to the dynamic properties of the system. In principle, a structural damage will modify those properties through changes in mass, stiffness and damping of the system. With this in mind the application of a feedback control system via pole placement to detect damages and update the matrices related to the system seems a viable option [42–45]. Indeed, once identified the new poles of the system, they could be imposed to the system in the nominal conditions, thus evaluating the feedback gain matrix and updating the matrices representing the system in the new status. However, as already said in Section 4 in this work the hypothesis that damages affect only the stiffness matrix is used. Moreover, before applying the pole placement, being the stiffness and the damping matrices of the system not diagonal resulting in inertially and dynamically coupled, the following *modal coordinate* transformation is applied to the system:

$$\vec{x} = [\Phi] \cdot \vec{q} \tag{16}$$

where the matrix [Φ] is the modal matrix of nxn dimension (4x4 in this work), containing the identified mode vectors of the system, while $\vec{q}$ are the new coordinates, called *modal* or *principal coordinates*. Those coordinates are able to diagonalize the mass, the damping and the stiffness matrices of the system, allowing so to write decoupled equation of motion.

$$[\hat{M}] = [\Phi]^T[M][\Phi] \tag{17}$$

$$[\hat{K}] = [\Phi]^T[K][\Phi] \tag{18}$$

$$[\hat{C}] = [\hat{M}] + [\hat{K}] \tag{19}$$

Once the system in modal coordinates is obtained and its new state-space representation is written, the pole placement technique is applied and the new state matrix of the controlled system is computed $[A_c]$. Recalling that in the state space the state matrix is composed as follows:

$$[A] = \begin{bmatrix} 0 & 1 \\ -\frac{[K]}{[M]} & -\frac{[R]}{[M]} \end{bmatrix} \tag{20}$$

To obtain the new, updated, stiffness matrix:

$$[\hat{K}]_{upd} = -[A_{21}] * [\hat{M}]; \tag{21}$$

However, it should be remembered that the obtained matrix represents the stiffness matrix of the system in modal coordinates. So, the inverse transformation should be applied, obtaining:

$$[K]_{upd} = [\Phi][\hat{K}][\Phi]^T \tag{22}$$

This method showed great capabilities in detecting damages and updating the stiffness matrix. However, it should pointed out that the results were obtained studying a fully *observable* and fully *controllable* system, as required by the hypothesis of the pole placement methods. However, in general, for complex structures this represent a big limitation. Indeed, even approximating the number of degrees of freedom of the system, it would be not so straight-forward and expensive to use a very high number of accelerometers. To overcome this limitation, Model Order Reduction methods could be used. However, the description and the application of this techniques is out of the scope of this work but they can be found in dedicated books, as [46].

6. Network architectures and training

The core objective of this Master's Thesis work is to compare the ability to detect structural damage between a physics-informed neural network and a purely data-driven neural network. Furthermore, conventional vibration-based techniques, which rely on the analysis of changes in both natural frequencies and vibration modes of the structure, are used as a reference to evaluate the advantages of employing machine learning algorithms over traditional methods.

6.1. Pre-processing

In the case of the undamaged structure, 1000 time records, each lasting 70 seconds, are recorded. Specifically, transverse accelerations are captured by four accelerometers at a sampling frequency of 128 Hz and organized into a 4-column matrix. The dataset is then normalized, as described in [47]. Normalization involves calculating the maximum absolute value for each channel and storing these values in a vector denoted as G. This normalization process ensures that each data point within the signals falls within the range of -1 to 1. Finally, the dataset is divided into three subsets: training, validation, and test, comprising 800, 100, and 100 records, respectively.

6.2. Training and test

The training dataset is employed to train the autoencoder model illustrated in Figure 18. Notably, the training phase for the PINN-CAE and DD-CAE models differs due to the custom loss function implemented for the former. Details of this custom loss function will be discussed in the subsequent subsection. During the CAE training process, 200 epochs are considered, with Mean Absolute Error (MAE) serving as the loss function. Additionally, a callback is applied to monitor the validation loss. MAE is evaluated separately for each accelerometer to assess the reconstruction error generated by the trained model when predicting the test set. The maximum MAE values observed across the entire test set are established as thresholds for anomaly detection. As mentioned earlier, MAE values exceeding these thresholds are indicative of time histories representing the damaged structure.

6.3. Autoencoder

Both machine learning algorithms, referred to as PINN-CAE and DD-CAE, share a common neural network architecture based on a convolutional autoencoder. Autoencoders are self-supervised learning techniques that aim to reconstruct input data at the output with minimal distortion after a series of transformations and data compression steps. They are widely used for denoising, data compression, and high-dimensional data visualization [48]. Convolutional Neural Networks (CNNs), upon which Convolutional Autoencoders (CAEs) are built, are a subset of Artificial Neural Networks (ANNs) that leverage convolution operations, offering advantages such as reduced parameter connections and faster convergence due to dimension reduction. This architecture is employed to remove irrelevant features while preserving essential information [49, 50].

In this Master's Thesis work, the autoencoder is constructed using the TensorFlow framework and assembled using the Keras API. As reported in Figure 18, the model counts 11 layers of the

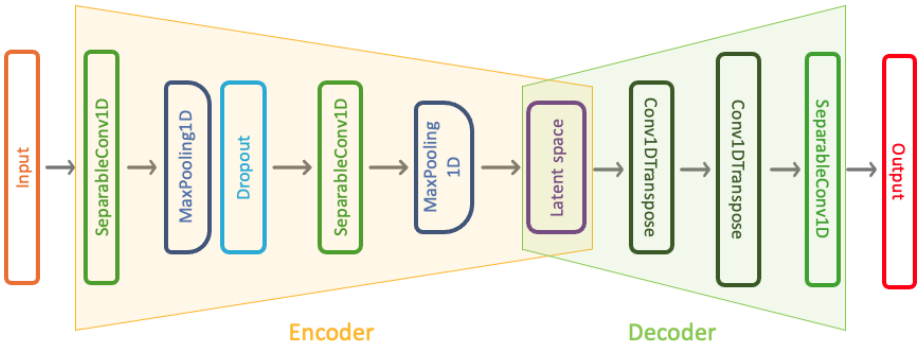


Figure 18: Autoencoder model.

following types:

1. **Separable Convolutional 1D layer**. This layer performs 1D convolutions separately on each channel of the input data. Then, it combines the channels through point-wise multiplication. This approach is highly effective for analyzing time series data, as demonstrated in [51];

2. **MaxPooling layer**. Convolutional layers tend to increase the number of parameters in the output tensors significantly compared to the input tensors, especially when numerous

filters are applied. To address this, a pooling layer is typically used after a convolutional layer. The purpose of the pooling layer is to downsample the feature map by retaining only the most significant information extracted by the preceding convolutional layer. Various pooling methods exist, but this work adopts MaxPooling, which selects the maximum value from a predefined sub-matrix;

3. **Dropout layer**. As already pointed out in Section 2, overfitting is a common challenge in neural network development, where a model becomes overly specialized in the training data, potentially leading to poor generalization to new data. Dropout is a regularization technique that enhances the network's robustness during training by encouraging it to learn general and recurring patterns. During training, if a tensor passes through a dropout layer, some of its values are randomly set to zero based on a dropout probability, which indicates the fraction of input elements to be zeroed out. During testing, no values are zeroed out, but the output is scaled by a factor equal to the dropout probability. Sometimes, dropout is applied only during training, leaving the test and prediction phases unchanged. Additional guidance on managing the dropout layer can be found in [52].

4. **Dense layer**. This layer is the simplest and most straightforward type. It is used to define the latent space of the neural network. Every neuron in a dense layer is connected to every neuron in the preceding layer. Each connection is associated with a weight that multiplies the input value. The dense layer also includes a bias term, denoted as "b," and an activation function, represented as "f." If "x" denotes the input tensor, "z" is the output tensor, and "W" represents the weights tensor, the mathematical equation for a dense layer is given as follows:

$$z = f(\tilde{z}) = f(Wx + b) \tag{23}$$

   Where $\tilde{z}$ is called weighted sum of the input.

5. **Transposed Convolutional 1D layer**. The layer in question is a specific type of convolutional layer designed to upscale the spatial resolution of an input tensor while preserving a connectivity pattern compatible with certain convolutional layers. Essentially, it functions as an operation that transforms an input tensor into an output tensor with a higher spatial resolution. This operation is commonly referred to as "deconvolution".

The training process for the autoencoder involves utilizing the Mean Absolute Error (MAE) as the loss function. Ass already introduced, MAE, expressed in Equation 24, quantifies the average magnitude of the absolute discrepancies between the predicted values (which are the outputs generated by the autoencoder) denoted as y_i and the input values x_i [53].

$$MAE = \frac{\sum_{i=1}^{N} |y_i - x_i|}{N} = \frac{\sum_{i=1}^{N} |e_i|}{N} \tag{24}$$

In this equation, N represents the number of signal samples. Furthermore, upon completing the training phase, the Mean Absolute Error (MAE) is utilized as an indicator for detecting anomalous time histories within the tested structure. The MAE loss essentially quantifies the level of error in the reconstruction carried out by the autoencoder. Therefore, it's reasonable to assume that a higher reconstruction error corresponds to more significant damage, as outlined in [54].

6.4. Physic-informed neural network

Training deep neural networks often necessitates access to extensive datasets, which can be challenging to obtain, especially for pre-existing structures or damaged scenarios. Physics-informed

neural networks offer a potential solution to this limitation. These networks can be trained using additional information derived from the underlying physical principles governing the dynamic behavior of the system. This approach seamlessly integrates both data and mathematical models, even in situations where the models may not be fully understood, are subject to uncertainty, or involve high-dimensional parameters, as highlighted in [55] and [56].

To facilitate the training of PINNs, conventional loss functions are not suitable. Instead, custom loss functions are essential. Specifically, these custom loss functions should incorporate the physical laws that govern the dynamic response of the system, thereby constraining the space of allowable solutions during autoencoder training. The outline of the proposed custom loss function is depicted in Figure 19.

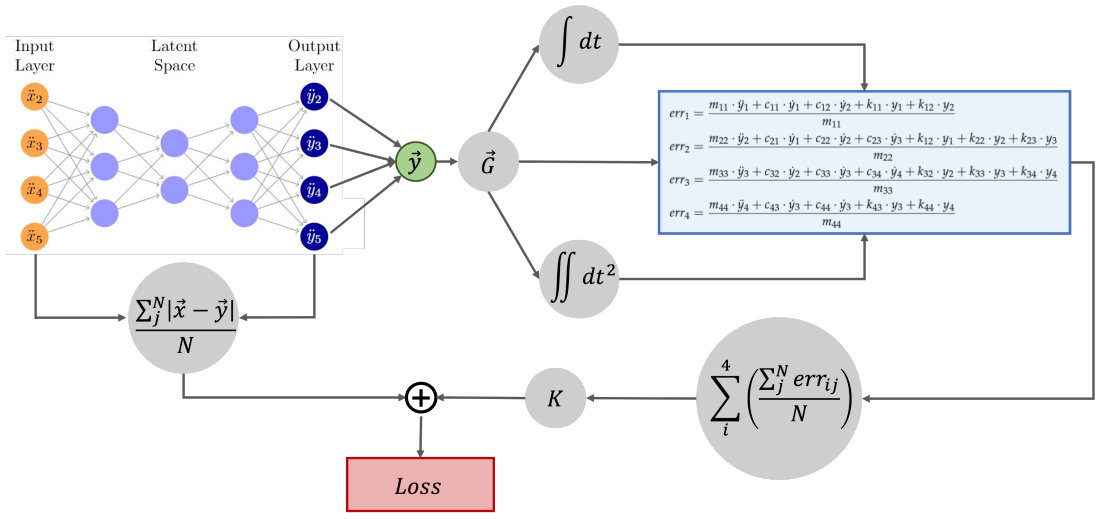


Figure 19: Custom loss scheme.

The signals denoted as y_i, which are the outcomes of the autoencoder, represent the reconstructed time histories of the scaled vibration data acquired during the healthy scenario. These data, in turn, depict the system's response to the input force applied by the hammer. However, it can be assumed that after a certain period, set at 10 seconds in this case, the transient behavior resulting from the forced motion of the system completely diminishes. Consequently, following the inverse scaling operation using the vector G (containing the scaler values applied during the preprocessing stage), the remaining portion of each reconstructed time history should exhibit low error when compared to the previously presented set of Ordinary Differential Equations (ODEs) in Equation 1.

In particular, to maintain consistency with measurement units, the ODEs are divided by the mass of the corresponding floor. By doing so, the error functions are expressed in acceleration units. This adjustment is made to ensure that the physical loss shares the same measurement units as the data-driven loss. Moreover, it's reasonable to anticipate a greater error when considering time histories originating from the damaged scenario. As a result, after appropriate time-domain integration, following methods such as those described in [57] and [58], the following error functions

are computed for each time instant:

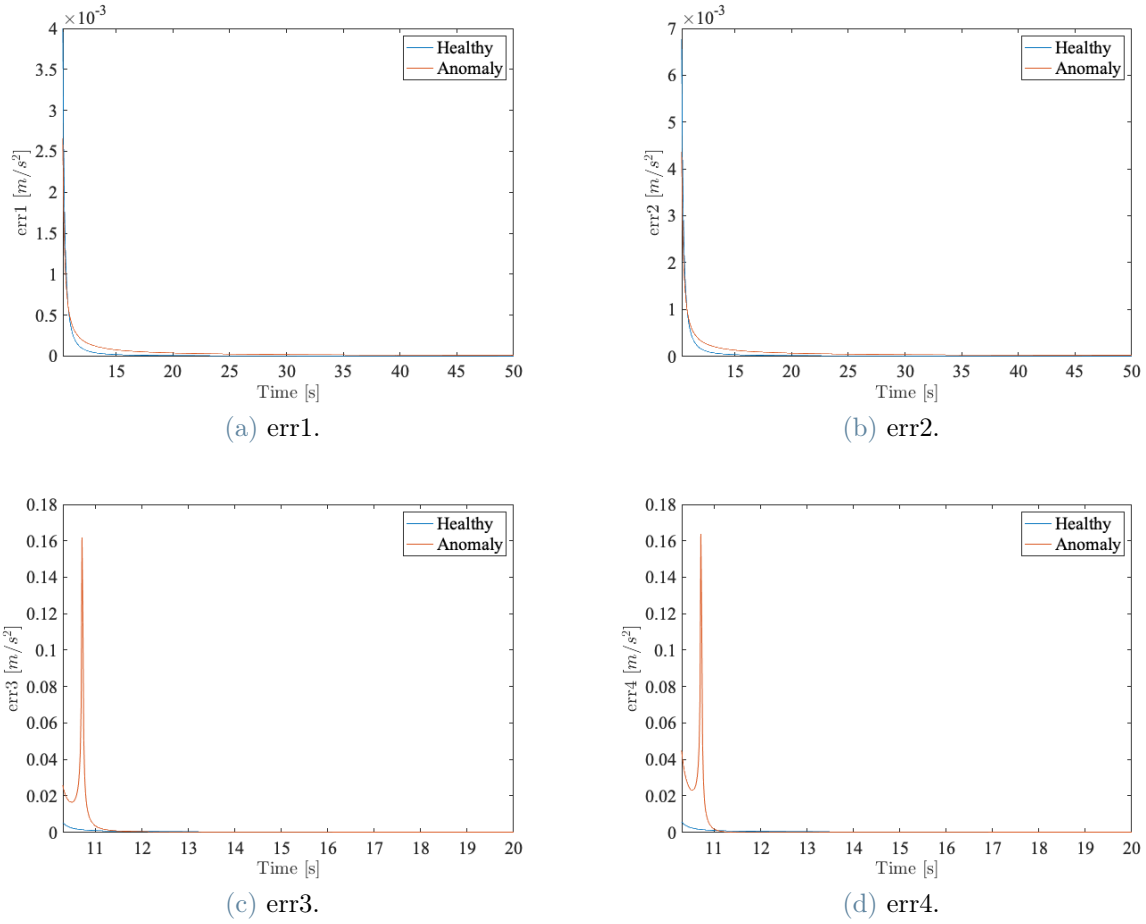
$$
\begin{aligned}
err_1 &= \frac{m_{11} \cdot \ddot{y}_1 + c_{11} \cdot \dot{y}_1 + c_{12} \cdot \dot{y}_2 + k_{11} \cdot y_1 + k_{12} \cdot y_2}{m_{11}} \\
err_2 &= \frac{m_{22} \cdot \ddot{y}_2 + c_{21} \cdot \dot{y}_1 + c_{22} \cdot \dot{y}_2 + c_{23} \cdot \dot{y}_3 + k_{12} \cdot y_1 + k_{22} \cdot y_2 + k_{23} \cdot y_3}{m_{22}} \\
err_3 &= \frac{m_{33} \cdot \ddot{y}_3 + c_{32} \cdot \dot{y}_2 + c_{33} \cdot \dot{y}_3 + c_{34} \cdot \dot{y}_4 + k_{32} \cdot y_2 + k_{33} \cdot y_3 + k_{34} \cdot y_4}{m_{33}} \\
err_4 &= \frac{m_{44} \cdot \ddot{y}_4 + c_{43} \cdot \dot{y}_3 + c_{44} \cdot \dot{y}_3 + k_{43} \cdot y_3 + k_{44} \cdot y_4}{m_{44}}
\end{aligned}
\tag{25}
$$

As previously mentioned, within these error functions, y_i, $\dot{y}_i$, and $\ddot{y}_i$ correspond to the displacement, velocity, and acceleration signals obtained through the autoencoder's reconstruction. Subsequently, the absolute values of their mean values are calculated and collectively form the physical component of the custom loss function:

$$
\mathcal{L}^{physic} = \frac{\sum_{i=1}^{N} |err_{1,i}|}{N} + \frac{\sum_{i=1}^{N} |err_{2,i}|}{N} + \frac{\sum_{i=1}^{N} |err_{3,i}|}{N} + \frac{\sum_{i=1}^{N} |err_{4,i}|}{N} \tag{26}
$$

In this context, where N represents the number of time steps for which the error functions are assessed.

To demonstrate the effectiveness of the information derived from the physical component of the loss, the error functions described in Equation 25 are computed for both "healthy" and "damaged" time histories. In the "damaged" scenario, a defect is introduced by reducing the stiffness by 28% of the nominal value, specifically in the third floor. The results, presented in Figure 20, affirm the equation of motion for the system's value as part of the custom loss. It is evident that for an anomalous time history, the error related to the equations of motion is greater compared to a healthy one, particularly for the degrees of freedom near the location of the damage.



(a) err1. (b) err2.

(c) err3. (d) err4.

Figure 20: Error functions evaluated for two different scenarios: "healthy" and "damaged".

On the other hand, for the data driven portion of the custom loss again the MAE loss function is taken into account.

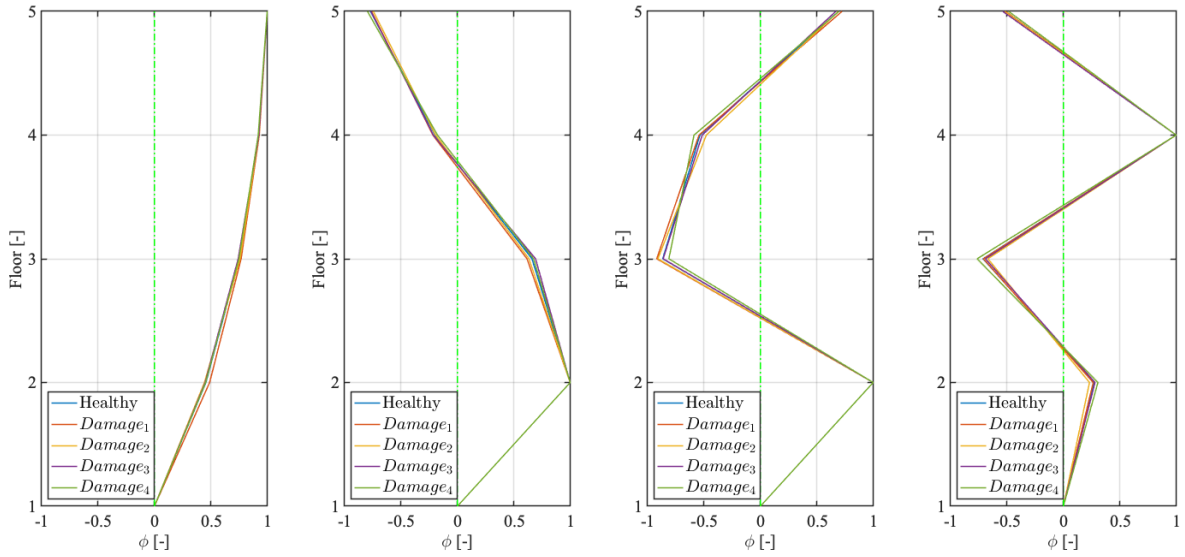
At the end, the obtained custom loss function is:

$$\mathcal{L} = K \cdot \left[\frac{\sum_{i=1}^{n} |err_{1,i}|}{n} + \frac{\sum_{i=1}^{n} |err_{2,i}|}{n} + \frac{\sum_{i=1}^{n} |err_{3,i}|}{n} + \frac{\sum_{i=1}^{n} |err_{4,i}|}{n} \right] + MAE \tag{27}$$

where K is a constant to express the physical part of the custom loss in an adimensional form, as the data driven portion of the loss function.

7. Results

The dataset comprising 240 damaged records is initially subjected to analysis through Experimental Modal Analysis (EMA). However, it is observed that natural frequencies and mode shapes exhibit only slight variations in response to damages of 10%, as depicted in Figure 21.



(a) 1st mode. (b) 2nd mode. (c) 3rd mode. (d) 4th mode.

Figure 21: Vibration modes of structure for 10% reduction of the stiffness value and for different positions of the damage.

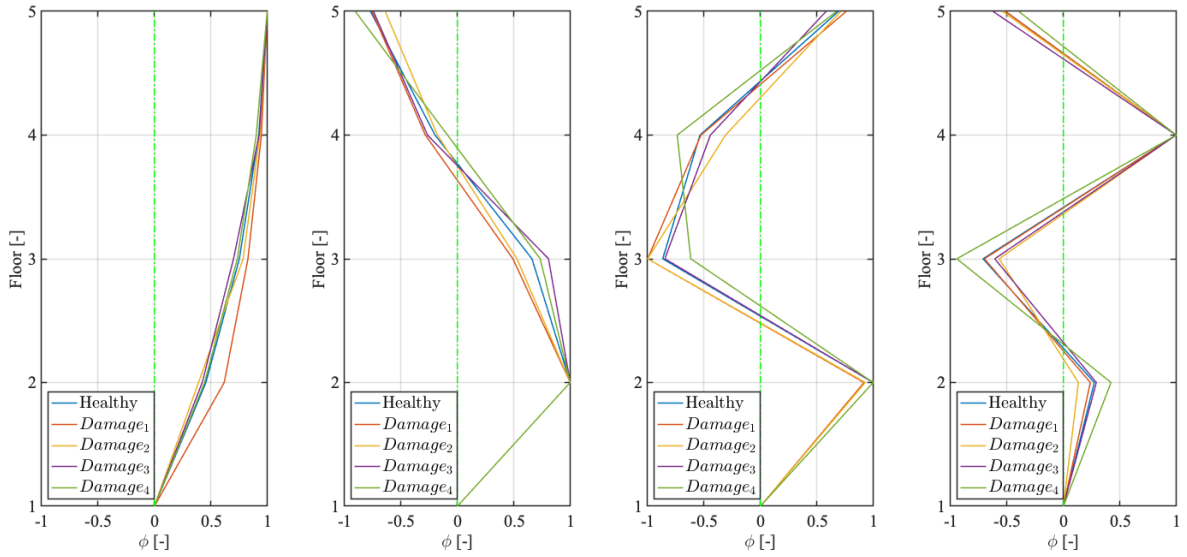
In the case of more significant damages, as illustrated in Figure 22, noticeable differences in natural frequencies and mode shapes become evident. However, it's worth noting that pinpointing the precise location of the damage is not straightforward. This observation aligns with a common challenge outlined in the literature, which highlights that vibration signals are reliable indicators of damage presence but may pose difficulties in precisely identifying its location, as discussed in [59]. With this approach in mind, the anomaly dataset is pre-processed using the same scaling parameters obtained during the training phase. Subsequently, this pre-processed data is input into both the PINN-CAE and DD-CAE models. The Mean Absolute Error (MAE) values are computed for each anomaly record and for each channel (corresponding to different accelerometers). These MAE values are then compared to the previously established MAE test thresholds. Any record with a loss exceeding the threshold is classified as an anomaly.

Both architectures successfully identify all the considered time histories as anomalies. This outcome confirms the anticipated higher precision of data-driven algorithms in detecting structural damage compared to conventional methods. Notably, this difference becomes particularly evident when the extent of the damage is relatively low.

Moreover, for each detected anomaly, the channel (corresponding to the accelerometer position, i.e., the floor) with the highest MAE loss is selected as the predicted damage location. This predicted position is then compared to the known real damage location. Subsequently, an accuracy indicator is evaluated for both of the considered algorithms, with damages ranging from 10% to 60% serving as the reference point.

$$\mathcal{A} = \frac{n_d}{n_{tot}} \times 100 \tag{28}$$

In this equation, where n_d represents the count of anomalies with a damage extent equal to or greater than the reference value, and where the model accurately identifies their positions, while n_{tot} stands for the total number of anomalies with a damage extent equal to or greater than the reference value. The outcomes of these evaluations are presented in Table 7.



(a) 1st mode. (b) 2nd mode. (c) 3rd mode. (d) 4th mode.

Figure 22: Vibration modes of structure for 50% reduction of the stiffness value and for different positions of the damage.

Table 7: Anomalies detection rates as function of the damage percentage for both PINN-CAE and DD-CAE.

Damage Percentage	**Accuracy $\mathcal{A}$** DD-CAE	PINN-CAE
-10%	33.19%	79.43%
-20%	40.20%	82.81%
-30%	52.24%	87.22%
-40%	65.11%	92.03%
-50%	84.61%	100%
-60%	100%	100%

It is possible to conclude that the PINN outperforms the results obtained with the purely data-driven approach, as expected.

8. Conclusion

A general introduction to Artificial Intelligence and Machine learning is given.

The main focus of this thesis work is to assess the accuracy of detecting structural damages using two distinct machine learning algorithms: a physics-informed convolutional autoencoder (PINN-CAE) and a purely data-driven convolutional autoencoder (DD-CAE). The evaluation is based on raw data obtained from experimental acquisitions conducted on a four-storey building, and both algorithms share the same structural architecture. The Mean Absolute Error (MAE) of the reconstruction is employed as an indicator to identify anomalous records.

Both the PINN-CAE and the DD-CAE outperform conventional vibration-based methods in their ability to detect structural damages and pinpoint their locations. They successfully identify all anomalous time histories and exhibit a high level of precision in detecting structural changes. Notably, the physics-informed network demonstrates greater accuracy in locating damage compared to the data-driven approach, especially for lower levels of damage severity. This emphasizes the significant potential of combining a data-driven architecture with information derived from the physical model of the studied system.

Future developments of this research will involve altering the mass of the system and utilizing the model in conjunction with neural networks to detect anomalies in more complex structures such as bridges or viaducts. Analytically representing such structures can be challenging, but a numerical model based on simulation methods like the Finite Element Method can be employed. The expansion of this work by considering a numerical model of more complex system will be developed in future by the authors.

Abbreviations

The following abbreviations are used in this manuscript:

SHM	Structural Health Monitoring
AI	Artificial Intelligence
ML	Machine Learning
NN	Neural network
ANN	Artificial Neural Networks
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
PINN	Physic-Informed Neural Network
CAE	Convolutional Autoencoder
MSE	Mean Square Error
MAE	Mean Absolute Error
LTI	Linear Time Invariant
EMA	Experimental Modal Analysis
FRF	Frequency Response Function
PINN-CAE	Physic-informed Convolutional Autoencoder
DD-CAE	Data-driven Convolutional Autoencoder

References

[1] Gangbing Song, Chuji Wang, and Bo Wang. Structural health monitoring (shm) of civil structures, 2017.

[2] Hong-Nan Li, Liang Ren, Zi-Guang Jia, Ting-Hua Yi, and Dong-Sheng Li. State-of-the-art in structural health monitoring of large and complex civil infrastructures. *Journal of Civil Structural Health Monitoring*, 6:3–16, 2016.

[3] Francisco J Pallarés, Michele Betti, Gianni Bartoli, and Luis Pallarés. Structural health monitoring (shm) and nondestructive testing (ndt) of slender masonry structures: A practical review. *Construction and Building Materials*, 297:123768, 2021.

[4] Diogo Montalvao, Nuno Manuel Mendes Maia, and António Manuel Relógio Ribeiro. A

review of vibration-based structural health monitoring with special emphasis on composite materials. *Shock and vibration digest*, 38(4):295–324, 2006.

[5] Charles R Farrar and Keith Worden. *Structural health monitoring: a machine learning perspective*. John Wiley & Sons, 2012.

[6] X Wu, J Ghaboussi, and JH Garrett Jr. Use of neural networks in detection of structural damage. *Computers & structures*, 42(4):649–659, 1992.

[7] OS Salawu. Detection of structural damage through changes in frequency: a review. *Engineering structures*, 19(9):718–723, 1997.

[8] D Goyal and BS Pabla. The vibration monitoring methods and signal processing techniques for structural health monitoring: a review. *Archives of Computational Methods in Engineering*, 23:585–594, 2016.

[9] Claus Peter Fritzen. Vibration-based structural health monitoring–concepts and applications. *Key Engineering Materials*, 293:3–20, 2005.

[10] Filipe Magalhães, Álvaro Cunha, and Elsa Caetano. Vibration based structural health monitoring of an arch bridge: From automated oma to damage detection. *Mechanical Systems and signal processing*, 28:212–228, 2012.

[11] Arnaud Deraemaeker, Edwin Reynders, Guido De Roeck, and Jyrki Kullaa. Vibration-based structural health monitoring using output-only measurements under changing environment. *Mechanical systems and signal processing*, 22(1):34–56, 2008.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[13] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[14] Zahra Rastin, Gholamreza Ghodrati Amiri, and Ehsan Darvishan. Unsupervised structural damage detection technique based on a deep convolutional autoencoder. *Shock and Vibration*, 2021:1–11, 2021.

[15] Luca Rosafalco, Andrea Manzoni, Alberto Corigliano, and Stefano Mariani. A time series autoencoder for load identification via dimensionality reduction of sensor recordings. *Engineering Proceedings*, 2(1):34, 2021.

[16] Xirui Ma, Yizhou Lin, Zhenhua Nie, and Hongwei Ma. Structural damage identification based on unsupervised feature-extraction via variational auto-encoder. *Measurement*, 160: 107811, 2020.

[17] Xudong Jian, Huaqiang Zhong, Ye Xia, and Limin Sun. Faulty data detection and classification for bridge structural health monitoring via statistical and deep-learning approach. *Structural Control and Health Monitoring*, 28(11):e2824, 2021.

[18] Jae Seok Do, Akeem Bayo Kareem, and Jang-Wook Hur. Lstm-autoencoder for vibration anomaly detection in vertical carousel storage and retrieval system (vcsrs). *Sensors*, 23(2), 2023. ISSN 1424-8220. doi: 10.3390/s23021009. URL `https://www.mdpi.com/1424-8220/23/2/1009`.

[19] Rafaelle Piazzaroli Finotti, Flavio de Souza Barbosa, Alexandre Abrahão Cury, and Roberto Leal Pimentel. Novelty detection using sparse auto-encoders to characterize structural vibration responses. *Arabian Journal for Science and Engineering*, pages 1–14, 2022.

[20] Maziar Raissi, Zhicheng Wang, Michael S Triantafyllou, and George Em Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, 2019.

[21] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[22] Yigit Yucesan and Felipe Viana. A physics-informed neural network for wind turbine main bearing fatigue. *International Journal of Prognostics and Health Management*, 11:17, 05 2020.

[23] Bin Huang and Jianhui Wang. Applications of physics-informed neural networks in power systems - a review. *IEEE Transactions on Power Systems*, 38(1):572–588, 2023. doi: 10.1109/TPWRS.2022.3162473.

[24] Francesco Morgan Bono, Luca Radicioni, Simone Cinquemani, and Gianluca Bombaci. A comparison of deep learning algorithms for anomaly detection in discrete mechanical systems. *Applied Sciences*, 13(9):5683, 2023.

[25] Francesco Morgan Bono, Luca Radicioni, Gianluca Bombaci, Claudio Somaschini, and Simone Cinquemani. An approach based on convolutional autoencoder for detecting damage location in a mechanical system. In *NDE 4.0, Predictive Maintenance, Communication, and Energy Systems: The Digital Transformation of NDE*, volume 12489, pages 93–100. SPIE, 2023.

[26] Dalvinder Singh Grewal. A critical conceptual analysis of definitions of artificial intelligence as applicable to computer engineering. *IOSR Journal of Computer Engineering*, 16(2):9–13, 2014.

[27] Michael Haenlein and Andreas Kaplan. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California management review*, 61(4): 5–14, 2019.

[28] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[29] Raul Rojas and Raúl Rojas. The backpropagation algorithm. *Neural networks: a systematic introduction*, pages 149–182, 1996.

[30] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[31] Jae-On Kim and James Curry. The treatment of missing data in multivariate analysis. *Sociological Methods & Research*, 6(2):215–240, 1977.

[32] Rizgar Zebari, Adnan Abdulazeez, Diyar Zeebaree, Dilovan Zebari, and Jwan Saeed. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1(2):56–70, 2020.

[33] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, pages 1–26, 2020.

[34] S. Di Carlo, L. Benedetti, and E. Di Gialleonardo. Teaching by active learning: A laboratory experience on fundamentals of vibrations. *International Journal of Mechanical Engineering Education*, 50(4):869–882, 2022. doi: 10.1177/03064190221082033. URL `https://doi.org/10.1177/03064190221082033`.

[35] Federico Cheli and Giorgio Diana. *Advanced dynamics of mechanical systems*, volume 2020. Springer, 2015.

[36] A. Ghali, A. M. Neville, and T. G. Brown. *Structural Analysis*. CRC Press, 12 2017. ISBN 9781315273006. doi: 10.1201/9781315273006.

[37] Man Liu and Dadiv G Gorman. Formulation of rayleigh damping and its extensions. *Computers & structures*, 57(2):277–285, 1995.

[38] Brian J Schwarz and Mark H Richardson. Experimental modal analysis. *CSI Reliability week*, 35(1):1–12, 1999.

[39] Álvaro Cunha and Elsa Caetano. Experimental modal analysis of civil engineering structures. 2006.

[40] P Hajela and FJ Soeiro. Structural damage detection based on static and modal analysis. *AIAA journal*, 28(6):1110–1115, 1990.

[41] Paraskevas N Paraskevopoulos. *Modern control engineering*. CRC Press, 2017.

[42] K Ramar and KK Appukuttan. Pole assignment for multi-input multi-output systems using output feedback. *Automatica*, 27(6):1061–1062, 1991.

[43] Fereidoun Amini and Farzaneh Modiri. Restoring dynamic properties of damaged buildings using pole assignment control method. *Journal of Vibroengineering*, 16(6):2920–2932, 2014.

[44] W Wonham. On pole assignment in multi-input controllable linear systems. *IEEE transactions on automatic control*, 12(6):660–665, 1967.

[45] YM Ram and S Elhay. Pole assignment in vibratory systems by multi-input control. *Journal of Sound and Vibration*, 230(2):309–321, 2000.

[46] Wilhelmus HA Schilders, Henk A Van der Vorst, and Joost Rommes. *Model order reduction: theory, research aspects and applications*, volume 13. Springer, 2008.

[47] F.M. Bono, L. Radicioni, and S. Cinquemani. A novel approach for quality control of automated production lines working under highly inconsistent conditions. *Engineering Applications of Artificial Intelligence*, 122:106149, 2023. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2023.106149. URL `https://www.sciencedirect.com/science/article/pii/S0952197623003330`.

[48] Yifei Zhang. A better autoencoder for image: Convolutional autoencoder. In *ICONIP17-DCEC. Available online: http://users. cecs. anu. edu. au/Tom. Gedeon/conf/ABCs2018/paper/ABCs2018_ paper_ 58. pdf (accessed on 23 March 2017)*, 2018.

[49] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[50] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[51] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1409–1416, 2019.

[52] Nitish Srivastava. Improving neural networks with dropout. *University of Toronto*, 182 (566):7, 2013.

[53] Jun Qi, Jun Du, Sabato Marco Siniscalchi, Xiaoli Ma, and Chin-Hui Lee. On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Letters*, 27:1485–1489, 2020. doi: 10.1109/lsp.2020.3016837. URL https://doi.org/10.1109%2Flsp.2020.3016837.

[54] Francesco Morgan Bono, Luca Radicioni, Simone Cinquemani, Lorenzo Benedetti, Gabriele Cazzulani, Claudio Somaschini, and Marco Belloli. A deep learning approach to detect failures in bridges based on the coherence of signals. *Future Internet*, 15(4), 2023. ISSN 1999-5903. doi: 10.3390/fi15040119. URL https://www.mdpi.com/1999-5903/15/4/119.

[55] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[56] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[57] Philip J Davis and Philip Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

[58] Anders Brandt and Rune Brincker. Integrating time signals in frequency domain – comparison with time domain integration. *Measurement*, 58:511–519, 2014. ISSN 0263-2241. doi: https://doi.org/10.1016/j.measurement.2014.09.004. URL https://www.sciencedirect.com/science/article/pii/S0263224114003832.

[59] Scott W Doebling, Charles R Farrar, Michael B Prime, and Daniel W Shevitz. Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: a literature review. 1996.

Abstract in lingua italiana

Questo lavoro di Tesi Magistrale rappresenta l'estensione di due articoli di ricerca incentrati sull'applicazione dell'intelligenza artificiale per il monitoraggio della condizione delle strutture. Infatti, una variazione della configurazione nominale può essere correlata ad un difetto strutturale che deve essere attenzionato prima che raggiunga condizioni critiche. E' evidente come la capacità di rilevare automaticamente i cambiamenti in una struttura abbia riscosso molto interesse negli ultimi anni. Inoltre, senza una conoscenza pregressa del sistema, l'intelligenza artificiale e, in modo più preciso, i modelli di *deep learning* potrebbero rivelarsi efficaci nella rilevazione di cambiamenti struttura e migliorare inoltre la capacità di identificarne la posizione. Tuttavia, l'acquisizione di dati relativi a scenari di strutture danneggiate non è sempre praticabile. In questo articolo viene fornita una panoramica generale dell'Intelligenza Artificiale, con particolare attenzione al *machine learning* e al *deep learning*. Inoltre, due approcci basati sul *deep learning* sono applicati a un modello di un edificio di quattro piani: un *autoencoder physic-based* e *autoencoder* semplicemente basato sui dati disponibili. Le prestazioni sono poi confrontate con quelle ottenute con approcci convenzionali basati sulla classica analisi modale. In particolare, i possibili danni strutturali sono simulati variando la rigidezza della dei pilastri, modellati attraverso molle. Entrambi gli algoritmi di *machine learning* hanno dimostrato migliori prestazione rispetto all'approccio tradizionale. Inoltre, viene confermato il maggiore potenziale nel rilevare e localizzare i danni delle reti neurali *physic-informed*.

Parole chiave: Intelligenza artificiale, *Machine Learning*, *Deep Learning*, Monitoraggio Strutturale, Rilevamento Danni, Reti Neurali, *Autoencoder* convoluzionale, Reti neurali *Physic-informed*.

Acknowledgements

Il percorso al Politecnico di Milano è stato tortuoso, pieno di alti e bassi e momenti complicati ma al contempo è stata, senza dubbio, l'esperienza più temprante e formativa della mia vita.

Vorrei ringraziare il prof. Cinquemani e tutto il team di ricerca impegnato in questo progetto. In particolare, vorrei ringraziare Morgan, Luca e Claudio, persone splendide e tutor dalle capacità incredibili. Lavorare con voi è stato veramente un piacere ed un periodo di grande appredimento.

Un ringrazimento speciale a tutti i miei amici, compagni di una vita che mi hanno sempre supportato e senza cui non avrei mai superato così tante difficoltà. In particolare, vorrei ringraziare Giuseppe T. che considero letteralamente un fratello, compagno di mille avventure e spero di altrettante tante in futuro.

Un grazie ad Antonio M., amico di una vita e oggi compagno di una nuova esperienza lavorativa che spero ci porti verso vette inesplorate.

Un grazie speciale ai due colleghi che hanno segnato questo percorso di Laurea Magistrale: Michele e Alberto. Con assoluta certezza posso dire che senza voi due, il risultato non sarebbe stato lo stesso.

Un ringrazimento a tutta la mia famiglia, ai miei zii Gianfranco e Stefania e i miei cugini Manuel e Gaia per essere stati sempre presenti con una parola di conforto e di incitamento. Un enorme grazie a mia sorella Claudia, ma sopratutto ai miei genitori Maria Rita e Riccardo. Conosco il peso dei vostri sacrifici e questo rende quello che avete fatto per me ancora più speciale.

In ultimo, ma non per importanza, ringrazio chi non c'è più: i miei nonni Vincenzo, Santina, Nellina e Franco e mia cugina Chiara, questo traguardo è anche per voi.