Master WAVES

(Waves, Acoustics, Vibrations, Engineering and Sound)

"Artificial Neural Networks for the rapid detection of defects"

Master Thesis

Author: Rodrigo S. Motta

Tutors: Regis Cottereau
Cedric Bellis

Marseille, December 11, 2023

Contents

Abstract

This project investigates the application of neural networks in classifying structures to determine the presence of specific features, such as defects or distinct characteristics. Working on the input data optimization, the study explores whether direct feeding of time series data into the neural network is sufficient or if preprocessing is necessary. The research involves generating 2D computational simulations of defect-free and defective rectangular samples, simulating methods from non-destructive evaluation. An artificial neural network (ANN) is created, and its performance, learning, and robustness are analyzed. The neural network was capable of reaching 100% accuracy for the base model. On the robustness analysis, it was verified that the model can be trained with noise until the amplitude of 20%. Also, a linear relation was observed between the noise amplitude that the model was trained and which test noise amplitude reached the maximum accuracy.

1 Introduction

In various branches of engineering, it is essential to rapidly and non-destructively examine whether a structure possesses a specific characteristic, such as a defect, like an inclusion with distinct properties or a crack. For example, ultrasonic waves are commonly used to inspect gas ducts to detect cracks that could cause failure. In more complex scenarios, such as in petroleum engineering, specific structures like salt domes are actively sought after as they indicate the presence of rock formations that may contain oil reservoirs.

This project aims to explore the potential of neural networks in assisting with classifying structures and determining whether they possess a given feature or not. One of the focuses is on understanding the optimal approach for inputting data into the neural network, exploring whether time series data should be directly fed without preprocessing or if preliminary steps can enhance the process. To achieve this, a collection of 2D computational simulations was generated to enable investigation, utilizing both defect-free and defective rectangular samples. Excitation and measurements were executed on the sample surfaces, similar to methods found in non-destructive evaluation. Subsequently, an ANN was created, having its performance, learning, and robustness analyzed.

This project took place in the *Laboratoire de Mécanique et d'Acoustique* in Marseille, France, and it had financial support from the *Centre National de la Recherche Scientifique (CNRS).* In addition to the building facilities of the laboratory, it was provided access to the Aix-Marseille University *Mésocentre*, a high-performance computing server, which contributed to more efficient calculations in this project.

In this report, the scientific context will be presented initially, followed by the description of dataset construction and the development of the artificial neural network (ANN). In the subsequent section, an analysis of the learning process and robustness will be conducted.

2 Scientific context

This section presents the research context, explaining the problem being dealt with and the proposed solution.

2.1 Defect problems in non-destructive testing

In the field of engineering and materials science, the accurate assessment of structural components and materials stands as a critical endeavor to ensure safety, reliability, and operational efficiency [1]. Non-destructive testing (NDT) emerges as a methodology within this context, enabling the meticulous inspection of components without inducing damage [2]. At the heart of NDT lies the imperative task of identifying defects—imperfections that have the potential to compromise mechanical performance, structural integrity, and operational longevity. These defects manifest in many forms, encompassing anything from microscopic cracks to concealed volumetric anomalies, bearing the latent potential to precipitate catastrophic failures if overlooked[1]. Within the spectrum of NDT techniques, the strategic deployment of wave propagation has been shown to be an effective approach to defect detection[3],[4],[5].

2.2 The use of wave propagation for defect detection

Wave propagation has evolved into a fundamental technique in non-destructive testing (NDT), offering an effective means to unveil defects within materials and structures without causing any damage [5]. This method entails the emission of waves, such as sound or vibrations, through materials, followed by the observation of how these waves interact with the material's internal composition. When waves encounter defects, they undergo specific changes that can be detected, recorded, and subsequently analyzed. This process facilitates the identification of defects' presence, precise location, size, and even the nature of the flaws. In this context, a range of wave types is employed, including ultrasound, electromagnetic waves, and seismic waves, with each one tailored to its most suitable application. For instance, in the aerospace industry, ultrasonic testing employs high-frequency sound waves to inspect aircraft components for cracks, ensuring their structural integrity. In civil engineering, seismic waves are used to study the stability of bridges, buildings, and other infrastructure[5],[6],[7],[8]. Likewise, medical imaging employs ultrasound waves to visualize internal organs and detect potential health issues[9].

Direct problems play a pivotal role in applying wave propagation[10]. For instance, in ultrasonic testing, the direct problem involves sending ultrasound waves into a material and analyzing the echoes to determine if defects exist[8]. In seismic studies, direct problems entail inducing controlled seismic waves and analyzing their reflections to gain insights into subsurface geological formations[10].

Inverse problems, on the other hand, step in when interpreting these measurements becomes

challenging[10]. They involve working backward from observed data to deduce hidden properties of defects or materials. For example, in ultrasonic testing, an inverse problem might entail reconstructing the exact size and shape of a hidden crack based on the echoes received. In seismic studies, inverse problems are applied to decipher the composition of underground layers based on recorded seismic waves. Inverse problems, on the other hand, step in when interpreting these measurements becomes challenging. They involve working backward from observed data to deduce hidden properties of defects or materials. For example, in ultrasonic testing, an inverse problem might entail reconstructing the exact size and shape of a hidden crack based on the echoes received [11]. In seismic studies, inverse problems are applied to decipher the composition of underground layers based on recorded seismic waves.

The integration of direct and inverse problems offers an approach to non-destructive testing. Direct problems provide real-world measurements, while inverse problems unravel the underlying details, enhancing our understanding of materials and structures[10].

In this context, it's important to delve into the challenges associated with both direct and inverse problems. Direct problems involve collecting data from wave interactions and using that data to understand if there are any defects. However, this task can become complex due to the intricate ways in which waves interact with materials. Deciphering these interactions requires specialized knowledge and advanced techniques.

Conversely, inverse problems introduce their own set of difficulties. In these scenarios, with the measurements from wave interactions, but the goal is to uncover hidden details, such as the shape or size of a defect. This process can be demanding because there might be numerous possible solutions that match the collected measurements. Selecting the accurate solution becomes akin to solving an intricate puzzle.

One notable challenge common to direct and inverse problems, especially when dealing with large or complex datasets, is the computational cost. Analyzing wave interactions, whether for direct measurements or inverse reconstruction, can demand substantial computing power and time. The complexity of the mathematical calculations and the need to process a significant amount of data contribute to this computational burden.

Moreover, there's an inherent uncertainty associated with wave-based methods. Noise, variations in material properties, and measurement inaccuracies can all impact the reliability of results. Balancing accuracy and robustness against computational efficiency becomes a critical consideration, especially in real-time applications or scenarios where rapid decisions are essential.

In this scenario, artificial neural networks emerge as a swift and effective solution capable of swiftly solving inverse problems following appropriate training [12],[13]. Moreover, the data generated through direct methods can be harnessed to nourish the network and facilitate its training. To implement this concept, the approach involves utilizing a straightforward seismic wave scenario as the subject of study.

2.3 Introduction to seismic waves

Seismic waves provide a distinct vantage point for defect detection, offering insights into concealed material structures [14]. These waves, stemming from natural occurrences like earthquakes or controlled actions like explosions, exhibit an exceptional capacity to traverse Earth's subsurface layers. Grasping the foundational principles underpinning these waves, along with core mathematical formulas, serves as a cornerstone for harnessing their potential in unveiling defects within diverse materials.

The elastic wave equation, which describes the propagation of elastic waves through a material, can be expressed in terms of the P-wave velocity (v_p) and S-wave velocity (v_s) [15] as follows:

$$\ddot{u} = v_p^2 \nabla\nabla \cdot u - v_s^2 \nabla \times \nabla \times u$$

In this equation: - u represents the displacement of the medium particles as a function of position (x, y, z) and time t, - v_p denotes the P-wave velocity in the material, - v_s stands for the S-wave velocity in the material.

This elastic wave equation captures the behavior of both P-waves and S-waves as they travel through different materials, accounting for the distinct velocities of these wave types.

2.3.1 Wave Types

Seismic waves manifest in two primary forms: body waves and surface waves. Body waves, comprising Primary waves (P-waves) and Secondary waves (S-waves), navigate materials distinctively, enabling differentiation between materials or structures[16]. When encountering defects or material boundaries, these waves respond by reflecting, refracting, and diffracting. This interaction lays the groundwork for defect detection, akin to utilizing underwater sonar for navigation.

In the realm of defect detection, both P-waves (Primary waves) and S-waves (Secondary waves) play pivotal roles in gaining insights into material structures. P-waves are compression waves that traverse materials by causing particles to move in the same direction as the wave itself. They stand as the swiftest seismic waves and are typically the initial ones recorded following an earthquake. Conversely, S-waves are shear waves that induce particle motion perpendicular to the wave's direction. They exhibit a slower pace compared to P-waves and are unable to propagate through liquids[16].

2.3.2 Wave Velocities

The equations for determining the velocities of P-waves (v_p) and S-waves (v_s) are as follows:
For P-waves:

$$v_p = \sqrt{\frac{K + \frac{4}{3}G}{\rho}}$$

For S-waves:

$$v_s = \sqrt{\frac{G}{\rho}}$$

Here, K signifies the bulk modulus of the material, G stands for the shear modulus of the material, and ρ represents the density of the material. These equations empower us to compute the velocities at which P-waves and S-waves traverse a given material. This understanding proves invaluable in comprehending their behavior when encountering defects or material boundaries.

2.3.3 Seismograms

Seismograms and traces play a crucial role in defect detection through seismic methods. In this context, a seismogram is a record of seismic waves that have interacted with a material or structure under investigation. These waves can originate from various sources, including artificial vibrations or natural events like earthquakes[16]. Seismograms display how seismic waves propagate through the medium, including any reflections, refractions, or scattering caused by internal features or defects within the material.

On the other hand, traces are individual recordings of ground motion obtained from specific seismic receivers (e.g., sensors or geophones) placed at different locations. Each trace captures the response of the material to incoming seismic waves, reflecting the characteristics of the subsurface [17]. By analyzing the patterns and features within the traces, seismologists and engineers can infer the presence of defects or anomalies in the material, such as voids, fractures, or structural irregularities. These defects can cause variations in the way seismic waves are transmitted and reflected, leading to distinct signatures in the traces and seismograms.

The interpretation of seismograms and traces involves comparing the observed seismic responses with expected behavior based on known geological and structural information. Machine learning techniques, such as pattern recognition algorithms or neural networks, can aid in automating the detection process by learning from labeled data and identifying subtle patterns associated with defects in seismic recordings[18]. Ultimately, seismograms and traces serve as valuable tools in non-destructive testing for defect detection, enabling the assessment of material integrity and structural health by analyzing the interaction of seismic waves with the material under investigation[19],[20],[21].

2.4 Neural network applied for defect detection

The evolution of deep learning, the rise of convolutional neural networks (CNNs), and the improvement of GPU computing have significantly impacted non-destructive testing, enabling accurate and efficient defect detection, mainly in the computer vision field [22]. Traditional methods require expertly engineered features for object detection, but deep learning autonomously extracts intricate patterns from raw data, revolutionizing the process [22].

The integration of deep learning, especially CNNs, has remarkably improved accuracy and

speed in defect detection [23],[24],[25]. This shift has reduced reliance on manual inspection, which is prone to errors and high costs. The newfound accuracy and efficiency are particularly beneficial for industries where ensuring product quality is crucial.
As mentioned before, neural networks have significantly advanced defect detection within the field of non-destructive testing (NDT) across industries. In the aerospace sector, they analyze ultrasonic wave patterns to identify flaws in critical aircraft components [26]. In manufacturing, these networks inspect welds and materials using radiographic images [27], ensuring structural integrity. Moreover, they enhance surface defect detection in materials such as metals and plastics [28], and automate the evaluation of microchips in semiconductor manufacturing[23]. In the energy industry, they monitor pipelines for corrosion and defects, contributing to the prevention of leaks[29].
Many of these methods are in the field of computer vision, where they are mainly related to inverse problems. Some data in industry are collected by the direct method, and time series is a recurrent domain in acoustics. Several applications using time-related neural networks, like RNN for example, were done for anomaly detection or health monitoring[22].
Recently, 1D convolution neural networks emerged as an efficient and powerful solution for time series data, and are applied in many examples of anomaly detection or health monitory[30],[31].
The 1D convolution is a process in deep learning that involves sliding a filter over a 1D input signal, extracting features by element-wise multiplication and summing. It produces a feature map that captures local patterns. Stride and padding control the movement and size of the output, while an activation function introduces non-linearity. This operation is vital for tasks like time series analysis, speech recognition, and text processing, where recognizing patterns within sequences is essential for accurate predictions [30].
In this project, the idea is to verify the possibility of detecting the presence of a defect as computer vision does (not considering location, shape etc.), but using time series signals, *i.e.*, building a dataset from a direct method to train the neural network to solve an inverse problem, the existence of a defect or not for example.
For the reader unfamiliar with the concepts of artificial neural networks and convolutional neural networks, a summary of this content is in the Appendix. The algorithms applied in this project were totally based on the TensorFlow framerwork[32].

3 Building the synthetic data

Deep learning heavily relies on an effective training process, where the accuracy and performance of the neural network are determined mainly. One crucial factor in successful training is the availability of representative data that adequately captures the problem's broad scenario. However, obtaining real-world data for specific applications, such as the scope of this thesis, can be extremely challenging. To overcome this limitation, the proposed approach involves generating a synthetic dataset to serve as inputs for the neural network.
This dataset is based on Spectral Element Method(SEM) simulation for seismic wave prop-

agation. In order to do that, the SEMLAB package (Version 2.0) [33], a set of Matlab functions intended for tutorial purposes that solves SH wave propagation for many cases, was used.
The function used for this first approach to the problem applies the Spectral Element Method to solve the 2D Shear-Horizontal wave equation, with stress free boundary conditions, zero initial conditions and a time dependent force source, in a structured undeformed grid. As it is not the focus of this report to discuss the numerical method, just a short introduction about the spatial discretization of the method is presented. However, some adaptations to the code were necessary to generate a proper dataset. Here it follows some of the changes:

- Defect insertion in the global mesh;
- Sensor and source distribution along the boundary;
- Time vector parameterized for a fixed duration;

3.1 Spectral Element Method (SEM)

This study utilizes the Spectral Element Method (SEM) for spatial discretization, which was introduced by Patera in 1984 [34]. The SEM can be viewed as a domain decomposition variant of Spectral Methods or a higher-order extension of the Finite Element Method. It inherits the accuracy of spectral convergence and geometric flexibility from its parent methods. In the SEM, the domain is divided into elements, and each element is equipped with a spectral subgrid that consists of Gauss-Lobatto-Legendre (GLL) nodes. The number of GLL nodes (NGLL=P+1) is determined by the polynomial order P used for spatial approximation. By transforming the weak form of the wave equation into an algebraic system, the SEM operates similarly to the Finite Element Method.

3.2 The influence of the refinement in the result

The accuracy and convergence of simulations in any numerical method rely on the mesh refinement. Therefore, the initial task involved evaluating the error in the results as the polynomial order was increased. The underlying model used for all tasks in this report remained the same, and further details about the model will be provided.
For this particular study, data was generated for various polynomial orders ranging from 4 to 18. The L2-Norm calculation was performed using the highest order (18) as the reference value. As the polynomial order increases, the positions of the nodes within each element change. Since the placement of the sensors and sources is approximated to the nearest node, slight modifications in position may occur with different levels of refinement, potentially influencing the results. To mitigate this effect, a source and a sensor were placed in opposite corners of the boundary, as their positions remain constant regardless of the polynomial order. Figure 1 illustrates the exponential decay of the error, highlighting the impact of increasing the polynomial order on the accuracy of the results.

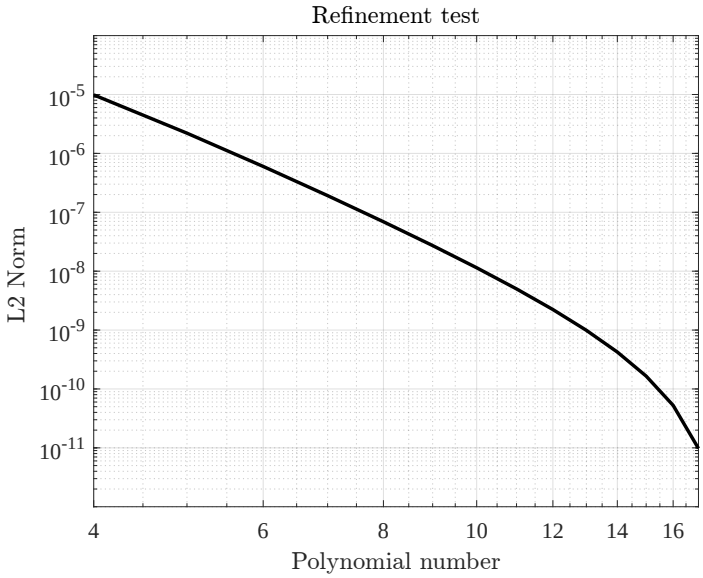


Figure 1: L2 norm of simulations with different polynomial orders using the most refined as the reference value.

3.3 Mesh and defect modelling

After investigating the required refinement for a load of 0.3 Hz, a polynomial order of 6 was selected to generate the initial dataset. For this particular case, a rectangular mesh was constructed with dimensions $L_x = 30$km and $L_y = 15$km. The mesh consisted of 30 elements on one side and 15 on the other, resulting in square elements with 1km sides. Figure 2 illustrates the mesh configuration for the defect-free scenario, showcasing both the larger squares representing individual mesh elements and the smaller squares representing the interconnections between internal nodes of each element.

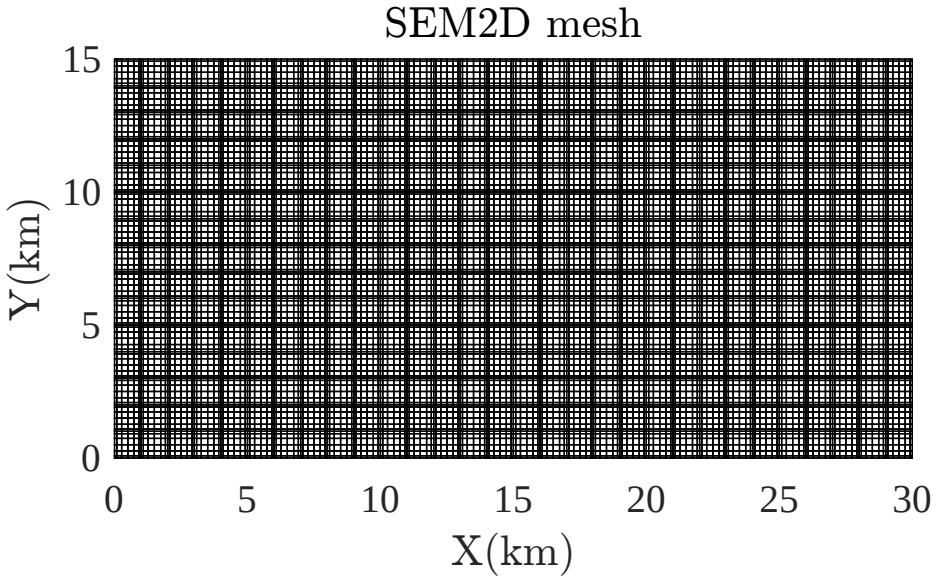


Figure 2: Mesh without defect .

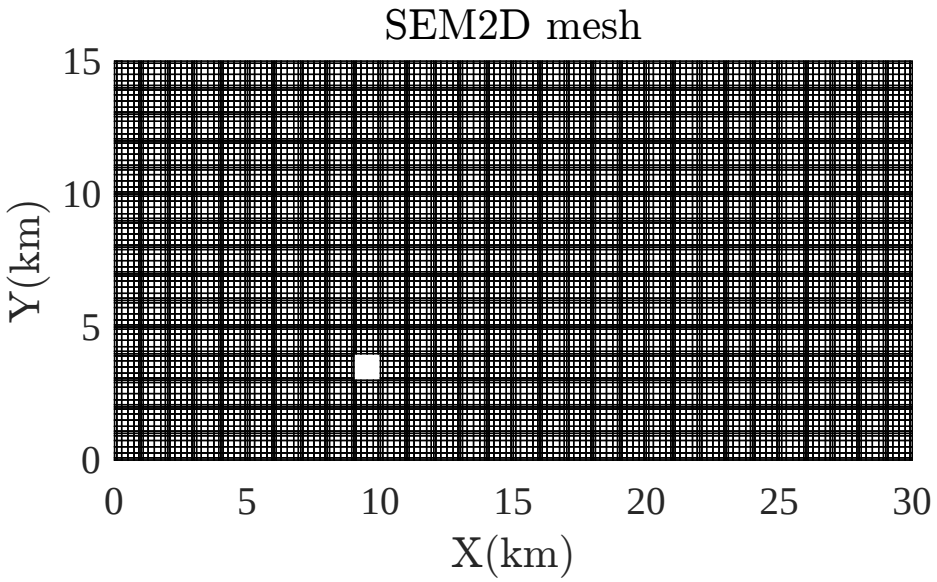


Figure 3: Mesh with defect .

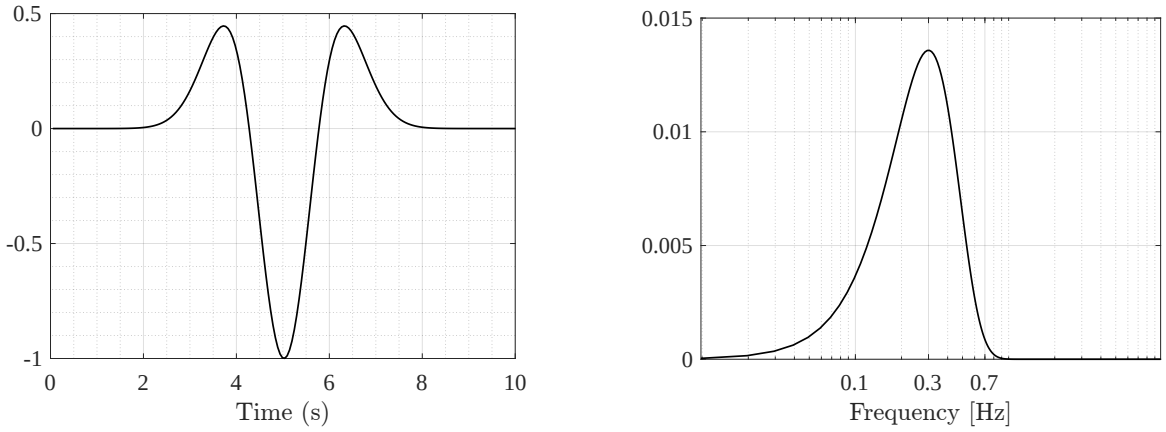
The subsequent task was to determine a method for incorporating a defect into the mesh. Two approaches were evaluated: imposing a Dirichlet condition with zero stiffness in a specific element or implementing a Neumann condition by excluding the nodes of the defective element from the global nodal matrix. The former approach could be advantageous for generating meshes with heterogeneous elements by adjusting their stiffness in the future. However, the latter approach was chosen for the initial dataset due to its straightforward implementation. Figure 3 illustrates the outcome of this approach, where the white area corresponds to the nodes removed from the designated defective element. In this case, element 100 in the mesh was considered the defect.

3.4 Sources and sensors placement

The next step involves configuring the source that will induce excitation in the system, as well as determining the distribution of the source and sensors within the domain. The Ricker wavelet has long been employed as a signal for seismic wave propagation due to its ability to accurately represent seismic phenomena.

In this model, the Ricker wavelet was utilized as the signal for the point source, which was positioned at the boundary of the system. Figure 4a illustrates the time function of the Ricker wavelet, characterized by a kernel duration of approximately 6 s and a delay of 5 s. The fundamental frequency of 0.3 Hz is evident in Figure 4b.

Similar to the source placement, the sensors are positioned along the boundary, following a non-destructive testing approach. Careful consideration was given to the configuration to ensure that a source and a sensor are never located in the same position, thus avoiding any synthetic data unrelated to the problem. Figure 5 illustrates the potential positions of the sources and the corresponding locations where the sensors were placed. Each sensor is consistently spaced at a distance of two elements (2 km) apart, except for the pairs located



(a) Time signal.

(b) Frequency response.

Figure 4: Ricker wavelet: time signal and frequency response.

at the top corners. The simulations were conducted with a single source and all the available sensors.

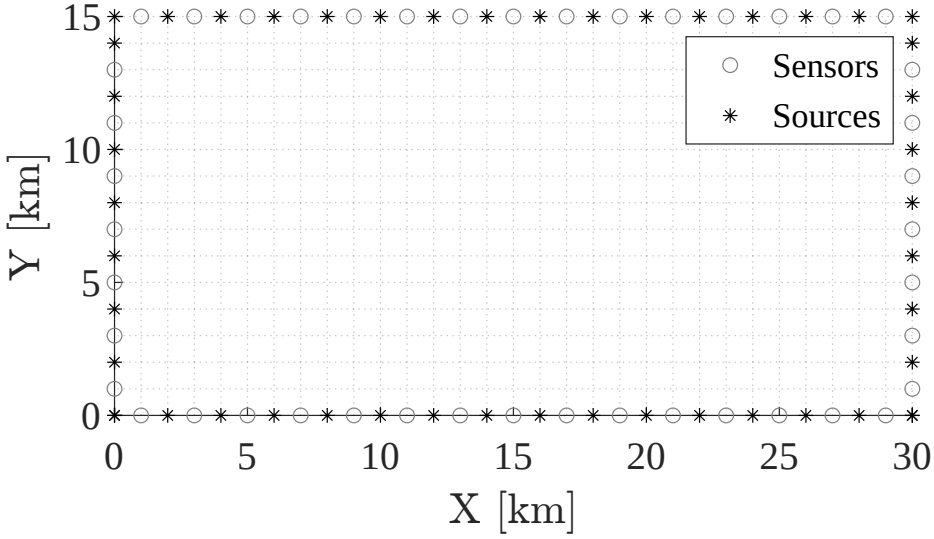


Figure 5: Sensors and Sources location in the model.

3.5 Base model

The base model used for constructing the initial dataset in this study consists of a rectangular non-dissipative homogeneous medium with dimensions $L_x = 30$km and $L_y = 15$ km. The mesh discretization consists of 30 elements along the x-axis and 15 elements along the y-axis, resulting in square elements with sides of 1.0 km. To simulate the presence of defect, a crack for example, the nodes corresponding to a single element are removed. The physical properties of the medium are density equal 1000 $\mathrm{kg\,m^{-3}}$ and the shear-modulus is 10^9 $\mathrm{N\,m^{-2}}$ resulting in a s-waves velocity equal 1.0 $\mathrm{km\,s^{-1}}$.

In terms of boundary conditions, a fixed Neumann condition is applied to the boundaries of

the model. This condition ensures that the displacements along the boundary remain fixed, representing a stress-free condition.

The simulation duration is set to 102 seconds, with a time discretization of approximately 0.05 seconds. Within this simulation timeframe, selected snapshots from the first 23 s were captured to illustrate specific stages of the wave propagation, the interaction with the defect, and the behavior at the boundaries. These snapshots are strategically chosen to provide insights into the wave dynamics and the effects of the defect and boundaries on the propagation.

Figure 6 presents a series of snapshots obtained from a simulation in the previously described model. The excitation load in the form of a Ricker wavelet with a frequency of 0.3 Hz is emitted from a point source located at the coordinate (0,0) (Figure 6a). The wave propagates through the non-dissipative medium until it encounters the defect (Figures 6c,6d,6e), leading to scattering and redistribution of the energy around the defect region. The primary wave continues to propagate, reflecting off the boundaries as it travels, while the scattered wave interacts with the defect and undergoes further reflections (Figures 6f,6g,6h).

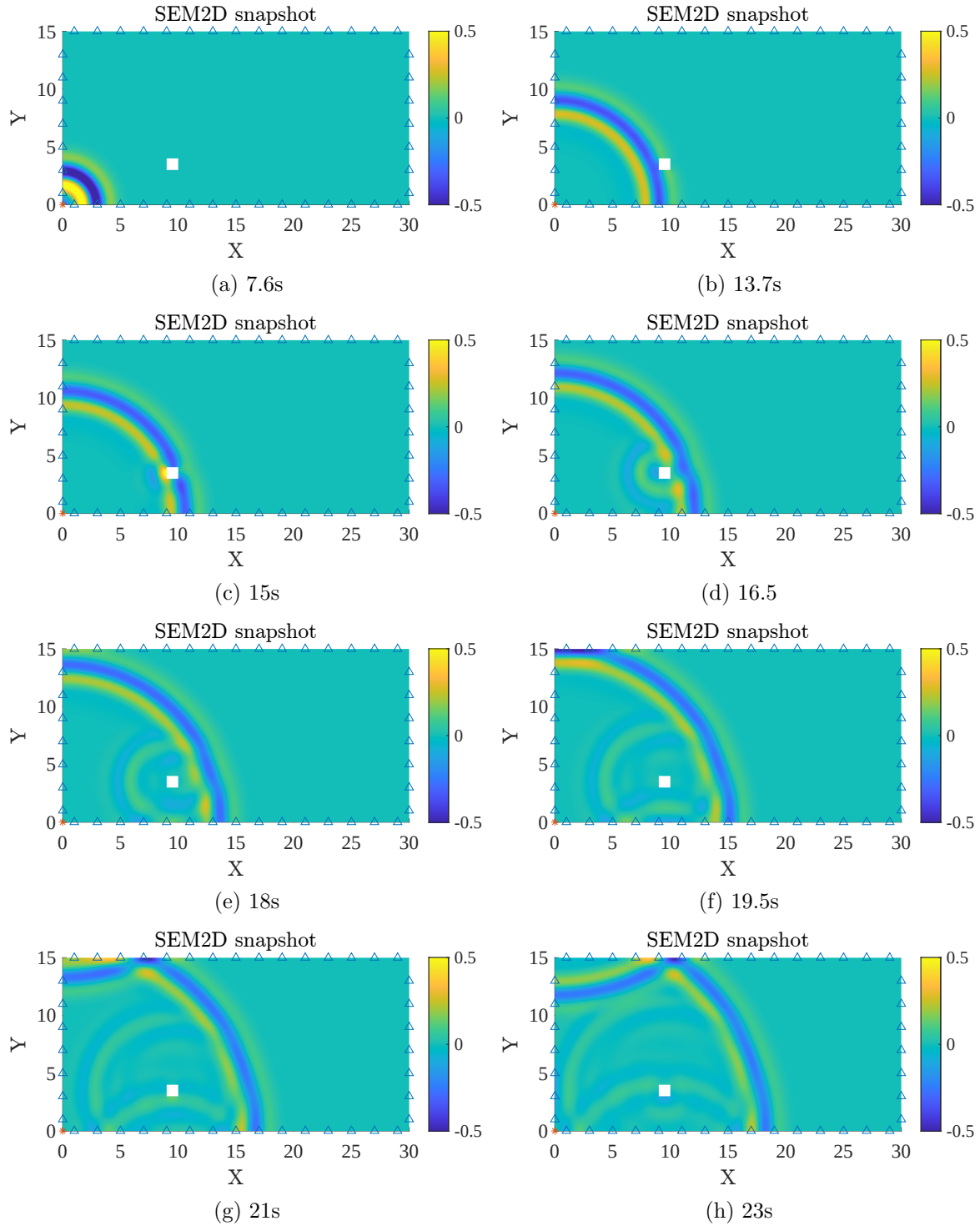


Figure 6: Selected snapshots of the first 23 s of the simulation in the presence of a defects

3.6 The dataset

As the objective is to use time series data, the input data is a seismogram, in other words, a signal of wave velocity per time, related to one source and sensor location. Figure 7 shows two examples of a set of seismograms, for models with and without defect. Both graphs were resulted from a sound source in the location (0,0) and the first 10 sensors on the x-axis for y=0;

Since the objective is to analyze time series data, the input data takes the form of seismograms, which represent the wave velocity signal over time at specific source and sensor locations. Figure 7 depicts two examples of seismogram sets, one for models without a defect and the other for models with a defect. Both graphs were generated using a sound source located at (0,0) and the first 10 sensors positioned along the x-axis at y=0.

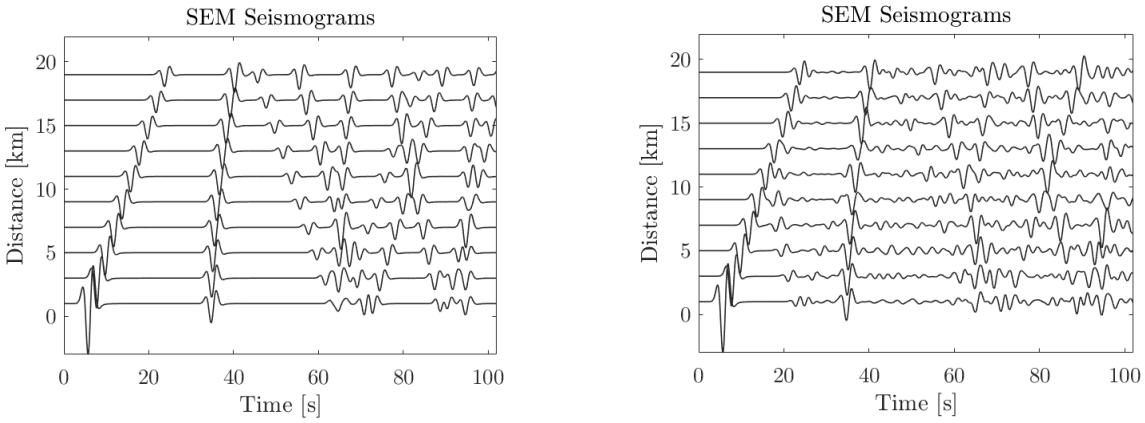
In Figure 7a, the signals correspond to the scenario without defects. The initial set of peaks observed in the sensor readings represents the direct wave, propagating with an s-wave velocity of 1.0 $\mathrm{km\,s^{-1}}$. For instance, the first sensor located 1.0 km away from the source detects the wave approximately 1.0 s after its emission. It is important to note that the source signal itself has a delay of 5.0 s. The subsequent sensors are placed at 2 km intervals, resulting in time differences of 2.0 s between the detection of peaks in adjacent sensors. The second set of peaks corresponds to the reflection of the direct wave on the top hard boundary. For the first sensor, this peak is detected at approximately 32 s, which represents the time it takes for the wave to travel back across the width of the domain, approximately 30 km. The third set of peaks is related to the wave traveling back after reflecting on the right-side boundary. As a result of multiple reflections, numerous peaks are detected thereafter. This observation suggests that the simulation duration could be based on these three sets of peaks, ensuring sufficient time for the wave to travel back and forth to the source location, as subsequent reflections occur.

In Figure 7b, the same sets of peaks can be observed, but additional peaks emerge due to the wave scattering caused by the defect. It is this characteristic feature that the neural network should be trained to detect.

As previously mentioned, the dataset must be tailored to suit the nature of the problem at hand. In the described model, there are a total of 450 possible defect locations. However, computing results for every single case would be computationally intensive and time-consuming. To address this challenge, a script was devised to randomly select defect locations. The approach involved dividing the medium into subgrids and selecting one element randomly from each subgrid. Figure 8 illustrates this process, with 18 defects depicted in black and the subgrids shown in gray.

By implementing this solution, the defects are more evenly distributed throughout the medium, resulting in a dataset that is more representative of real-world scenarios. Following these procedures, the dataset comprises a total of 2112 samples without defects and 38,016 samples with defects.

The final step in building the dataset involves organizing the data while ensuring its representativeness. In the case of a dataset for neural networks, it is crucial to divide it into



(a) without defect (b) with defect

Figure 7: Spectral element method seismogram for a sound source located on (0,0) coordinate, and the sensors at bottom boundary (y=0).

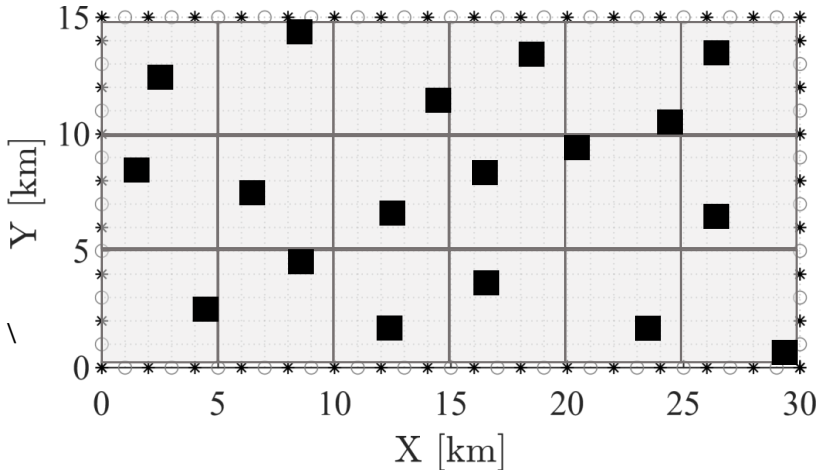


Figure 8: Defect selection process example to provide more reliable data .

training and testing sets, each with corresponding input and label datasets. Furthermore, it is important to have an equal number of samples for each case (with and without defects). The following processes were undertaken to achieve these objectives:

- First copy the data without defect until it reaches the same amount of samples, resulting in a total of 76032 samples.
- Training dataset: Random selection of 60000 samples, 30000 of each type.
- Test dataset: The 16032 remaining after the random selection of the training dataset.
- Dataset shuffling.
- Linking the dataset with the shuffle index to enable retrieval of information from each signal, such as defect location.

Following these steps, the dataset is properly organized and ready for further analysis and utilization.

As some of the samples were copied and there is also a symmetry in our simulation, some of the samples will repeated. To avoid problems with this configuration, the application of noise to provide data randomness and augmentation was studied in the section robustness analysis. Besides that, a small control dataset was built, following the same procedure, but with different sources, sensors, and defect positions. Thus, the generalization of the model can be tested without any concern.

4 Building a Neural Network

In tackling this particular challenge, the approach involves integrating two types of architectures. One focuses on finding key characteristics in the data, while the other one handles making decisions based on what was found. In this project, the 1D convolutional neural network is used to uncover the important features in the data. Then, a group of connected layers is used to decide between two categories, which is known as binary classification, where 0 represents no defect and 1 indicates the presence of a defect. The base structure was defined as the one in Figure 9.

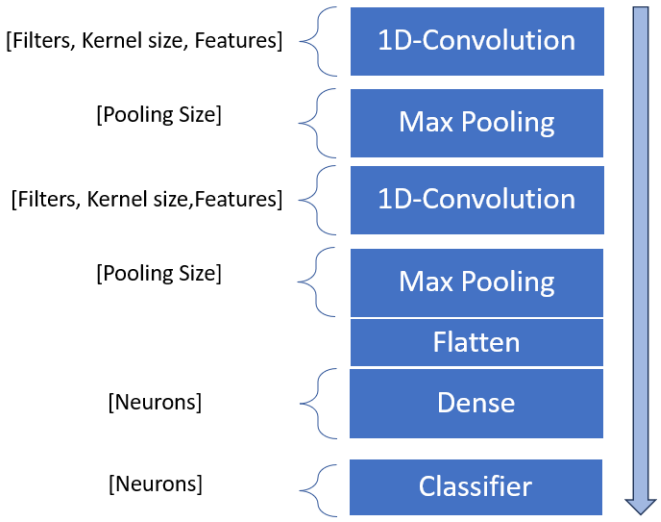


Figure 9: The base architecture of the neural network model to detect the presence of a defect.

Once the architecture is established, the focus shifts towards selecting hyper-parameters. These parameters function as settings that influence the performance of the frameworks. The objective, however, encompasses not only achieving favorable outcomes but also comprehending the learning processes embedded within the frameworks. Consequently, these parameters are adjusted to not only optimize results but also to offer insights into the intricacies of the learning mechanism.
Nevertheless, accomplishing this objective can pose challenges, often due to factors such as the multitude of parameters or the intricate nature of the data. For example, in scenarios involving images of cats and a 2D convolutional layer, it becomes perceptible that certain

filters identify the cat's edges. These filters subsequently piece together these edges to ascertain the formation of a complete cat image. In the context of this study, the complexity lies in the utilization of non-periodic time-based signals, which deviate from a regular pattern. In order to achieve the objectives, certain decisions were made concerning the model:

- **Simplicity:** The intention is to maintain simplicity to facilitate enhanced analysis of filters and their outputs.
- **Kernel Size:** A wider kernel size is chosen to allow more effective observation of shapes.
- **Balancing Speed and Accuracy:** The model should ensure quick training while upholding a satisfactory level of accuracy.

Across all convolutional and dense layers, the activation function employed is ReLU. However, within the classifier, the sigmoid activation function was adopted. Additionally, padding was configured to maintain consistent sizes as the input moved through each layer, and a pooling size of 4 was applied. Notably, biases were disregarded in the convolutional layers, while they were retained in the dense layer. In all the architectural designs, a single input feature was utilized, namely, the time domain signal.
The training process can be outlined through the following steps:

- Loading the dataset (training and test data)
- Segmenting a portion of the training data for validation
- Configuring and compiling the model
- Establishing functions to govern the training process
- Setting up and executing the fit function
- Analyzing the learning curves
- Assessing the model's generalization using the test data

This fundamental framework guided the majority of processes within this project, with minor adaptations based on specific study requirements.

4.1 Training process

The training process of the model utilized TensorFlow [32], an open-source platform in Python designed for machine learning tasks. This platform enables building, training, validating, and testing machine learning models. Although TensorFlow offers a broad range of applications, this project focuses primarily on the steps mentioned earlier.
The dataset creation was conducted using Matlab, resulting in a '.mat' file format. To import this data into Python, a specialized module was required due to the specific type and size

of the data. Following the extraction of training and test sets, the model was configured and compiled. The cross-binary entropy function was adopted as the loss function, and the Adam algorithm was employed as the optimizer. The metric used for evaluation was accuracy, which gauges how frequently the predicted labels align with the actual data labels. The output accuracy value ranges between 0 and 1, with 1 indicating all labels were correctly predicted.

Two functions were developed to regulate the training process. One function halts the model when accuracy remains relatively unchanged, indicating it has reached a plateau in the loss process. The other function assists in identifying this plateau by reducing the learning rate by a factor of 0.1 after the loss remains constant for 5 epochs. The learning rate starts at the default value of 0.001 and gradually decreases to 0.00001.

During the training process, the fit function incorporates these callbacks and employs a batch size of 32. Since the training is guided by the maximum accuracy and minimum loss, the specific number of epochs is not crucial; it simply needs to be sufficiently large to cover the necessary computations. Notably, the fit function segregates validation data from training data. In this project, a majority of this split was set at 20% of the training data.

Subsequently, the training started, while simultaneously recording the history of loss and accuracy. These recorded values, plotted against epochs, are referred to as learning curves. Analyzing these curves allow researchers to ascertain instances of overfitting or underfitting.

4.2 Tunning hyperparameters

In order to increase accuracy, decrease loss and avoid underfitting and overfitting the hyperparameters must be properly tuned. They are kernel size, number of filters, number of neurons, the use of dropout or not, as well as, the dropout value. Following the base model, the initial model has the hyperparameters as shown in Figure 10.

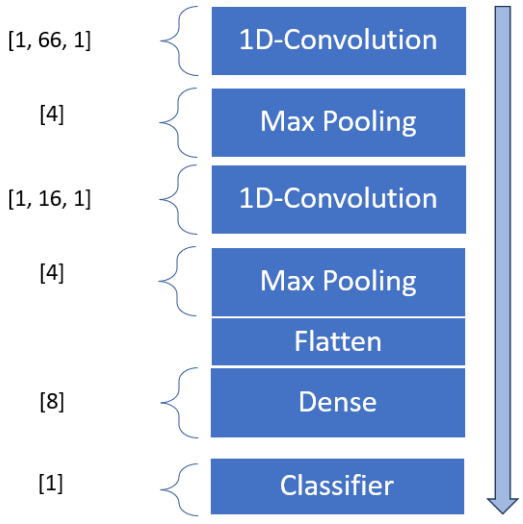


Figure 10: The initial neural network model to detect the presence of a defect.

As a start, a simple model was implemented, with just one filter in each convolutional layer. The highlight here is the kernel size compatible with the size of the signal generated by the

source in the simulations. Also with a pooling of size 4, the kernel size for the second layer is adjusted to 16, approximately the quarter of the original kernel size.
In Figure 11 it is possible to visualize the learning curves from the training of the initial model.
As expected, the lack of parameters, *i.e.* layers or filter makes the model take a lot of epochs to train it. The accuracy is very high, almost 100%. However, the ripples in the curves and the space between them indicate overfitting.

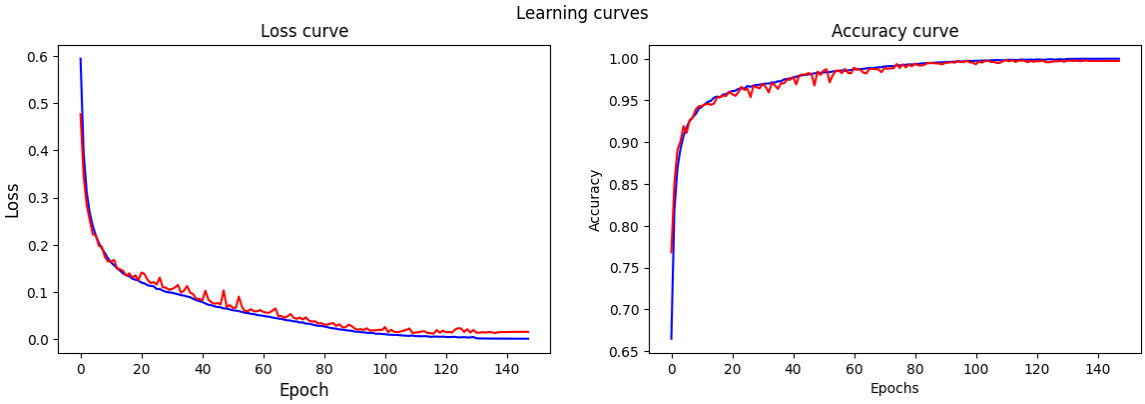


Figure 11: Learning curves from the initial model .

Thus, in order to correct this a hyperparameter tuning process was made, focusing on the kernel size, number of filters, and the possibility of application of a dropout.
After several runs, the final neural network model for this data has the configuration indicated in Figure 12. A slight increase in the kernel size and an increase in the number of filters, with 4 filters in the first layer and 16 in the second layer. Also, a dropout of 0.2 was added after the dense layer.

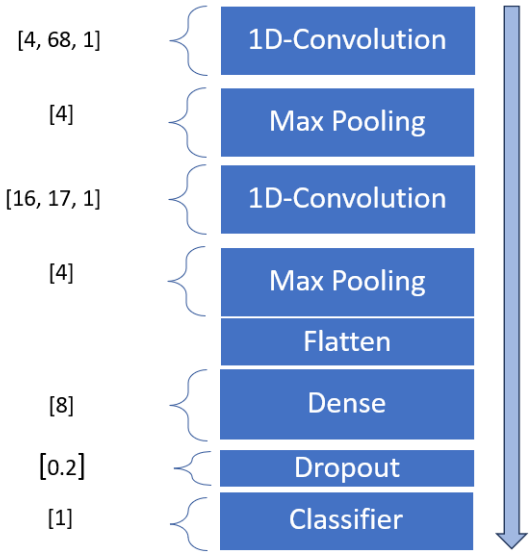


Figure 12: The final neural network model to detect the presence of a defect.

Taking a look at the learning curves in Figure 13, it is possible to see no ripples, the curves are approximately aligned and it took only 18 epochs to train the model and reach the accuracy of 100%. Thus, obtaining a model efficient to train, high accuracy and with a good fit.

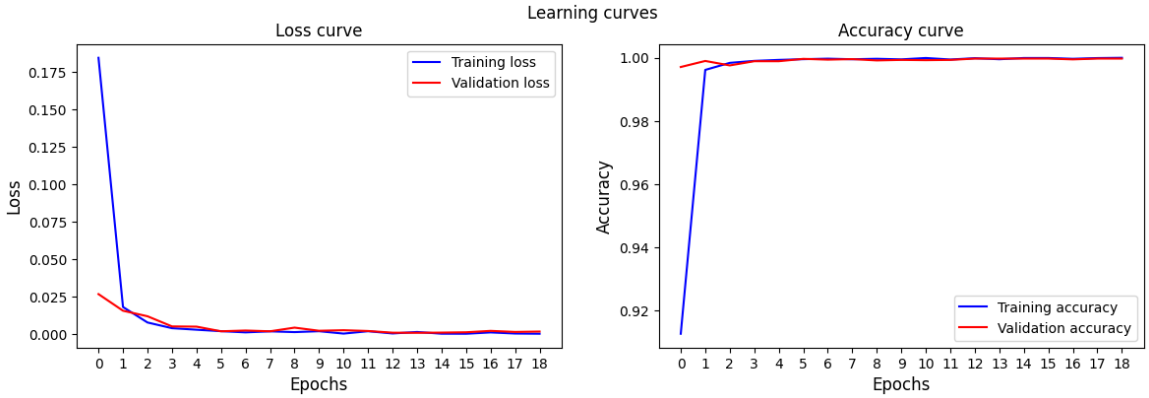


Figure 13: Learning curves from the final model after the hyperparameter tuning process .

The last part of the training, is the verification of the generalization of the model. To do that, the model is tested with data that was never presented to it. Also, the model was tested with data generated from a different source, sensor, and defect positions, as control data.
In both, the performance was very high, 100% accuracy for the test data, and 99,34% for the control data. This indicates that the model generalized very well. Important to highlight that generalization occurs in data with similar distribution. If the test data is generated with another source velocity or a different domain geometry, the model is not going to perform well, as both of these parameters would affect the position of the peaks in the generated signal.

5 Results and Analysis

In this section, the neural network model will be analyzed according to its learning process and robustness.

5.1 Learning Analysis

To understand the learning process, many analyses can be made related to the weights, outputs, and learning curves. Two approaches were chosen, first, to analyze in the first convolutional layer, the weights and outputs variation in 4 defined checkpoints along the learning curves. Second, analyze the output change in the second layer and the Flatten layer for samples with and without defects with the same source and sensor position. These approaches were chosen due to the importance of understanding how the features are extracted from the input signal and presented to the classification layers.

In Figure 14 are the samples that were used to analyze the output of the mentioned layers. Both samples have the same source and sensor position, the only difference is that on the left there is no defect and on the right there is.

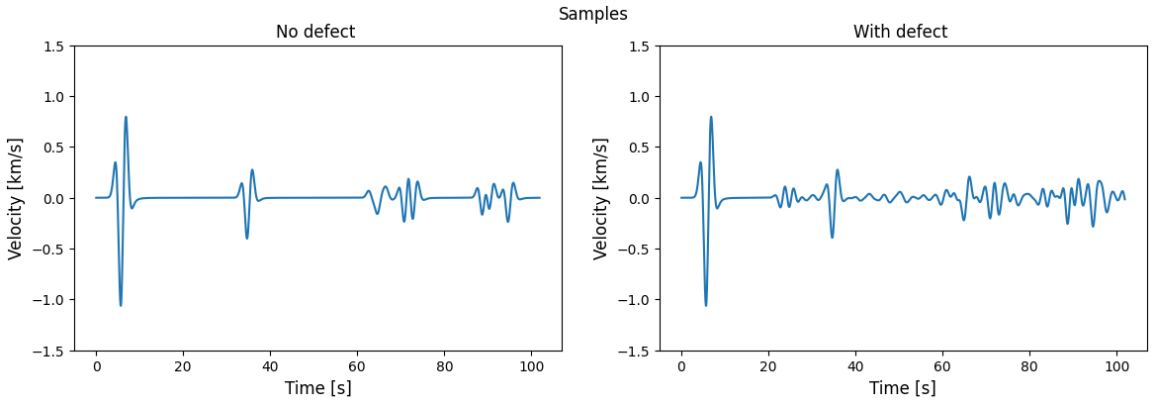


Figure 14: Samples selected to visualize the output of the layers.

5.1.1 Checkpoint analysis

The model underwent a reset and subsequent training, with the added step of saving its results after each epoch where accuracy evolved. Four distinct checkpoints were selected to capture pivotal moments. The first checkpoint was set for the initial epoch, the second marked the first notable accuracy improvement, the third was designated upon reaching a 99.90% accuracy threshold, and the fourth was positioned at the conclusion of the model's training, which achieved its maximum accuracy.
In Figure 15 the checkpoints are marked in the accuracy curve.

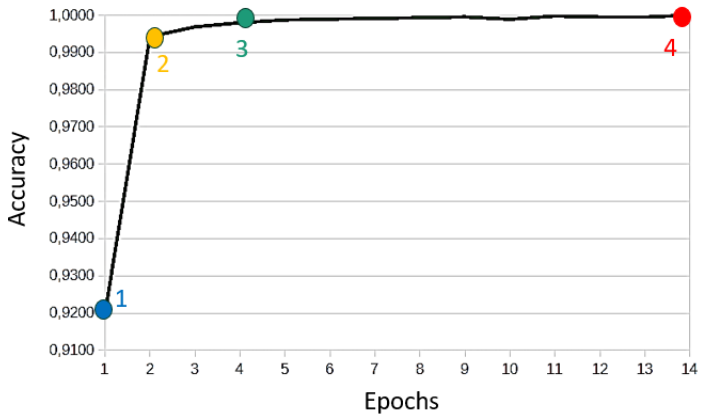


Figure 15: Checkpoints selected in the accuracy curve .

Figure 16 showcases the weights of the four filters from the initial layer across each checkpoint. Notably, the weights corresponding to the final model are highlighted in red. A distinctive waveform pattern emerges within the weights, signifying that the learning process predominantly emphasizes augmenting the distinction between positive and negative weights.

Filters 1 and 3 exhibit remarkable similarity, albeit with a slight phase shift. An intriguing observation is that these filters possess prominent peaks with a temporal offset of roughly 1.5 seconds. This temporal difference aligns with the time gap between the peaks of the wave generated by the source, offering a tangible link between these filter features and the source-generated wave patterns.

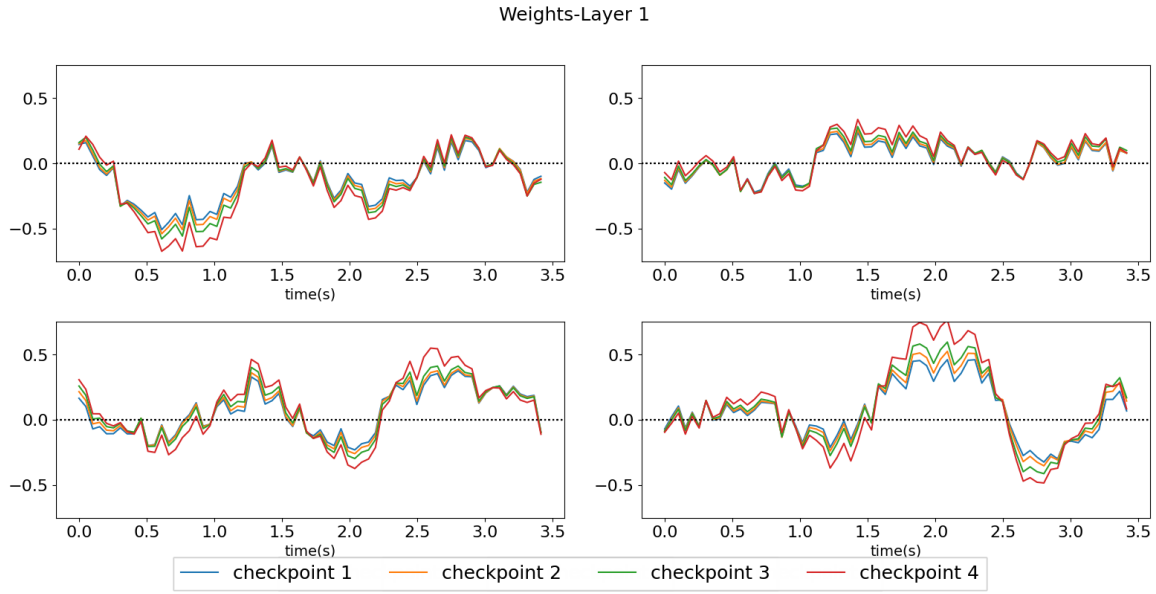


Figure 16: Learning evolution of the first layer weights according to the selected checkpoints .

In an effort to comprehend the role of these weights, a non-defective sample was introduced to the network, and the resulting output was scrutinized.
As illustrated in Figure 17, this output reveals a discernible pattern. It becomes evident that the four filters play a role in segmenting the sample into distinct components, effectively isolating the peaks of the waves as they traverse the domain. The observed phase disparity in filters 1 and 3 is mirrored in the output. Both filters capture the primary peak, yet they diverge in identifying the side peaks on opposite sides.
Moreover, the ongoing learning process is causing the peaks to become narrower. This phenomenon signifies that the weights are endeavoring to achieve heightened precision in peak extraction. This refinement contributes to the progressive enhancement of accuracy throughout the learning process.

5.1.2 Output analysis

The preceding sections revealed that the initial convolutional layer is tasked with capturing distinctive peak patterns from the input.
Turning to the subsequent layer, the examination becomes intricate due to the interplay of 4 filters in the first layer and 16 filters in the second, culminating in a total of 64 filters that warrant scrutiny. In this context, the focus shifts towards scrutinizing the output of the second layer, particularly for samples with and without defects. This illustration serves as a preliminary insight, offering potential pathways for future projects aiming to unravel

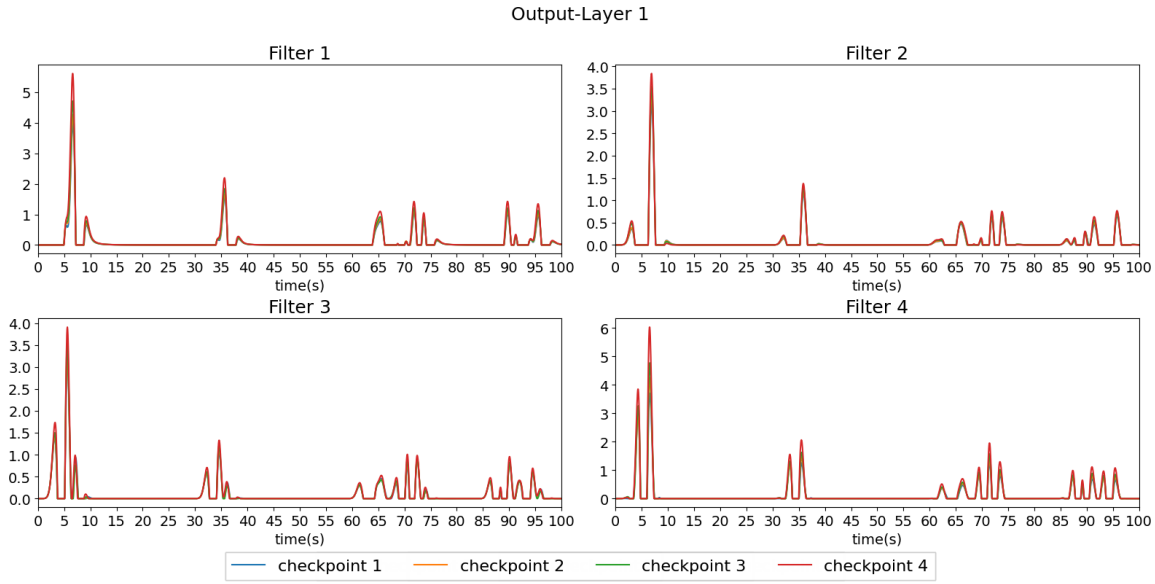


Figure 17: Learning evolution of the first layer output according to the selected checkpoints for a sample without defect.

the nuances embedded within this layer. Figure 18 illustrates the output of the second convolutional layer for two input signals: one with a defect and one without.

Broadly, this layer appears to be engaged in detecting signal patterns like reflections, interferences, and peak identifications. Filter 16, for instance, seems to concentrate on extracting the fundamental peak detections. Notably, it portrays a overview of when the source signal, the dispersed wave from defects, and the waves rebounded off boundaries were detected. Meanwhile, Filter 15 appears to center its attention on recognizing interference caused by reflected waves. Intriguingly, Filter 14 consistently maintains a flat line at 0, suggesting a possible reference to a DC component. The hypothesis proved false after testing the input with a forced DC component and the filter kept the same response.

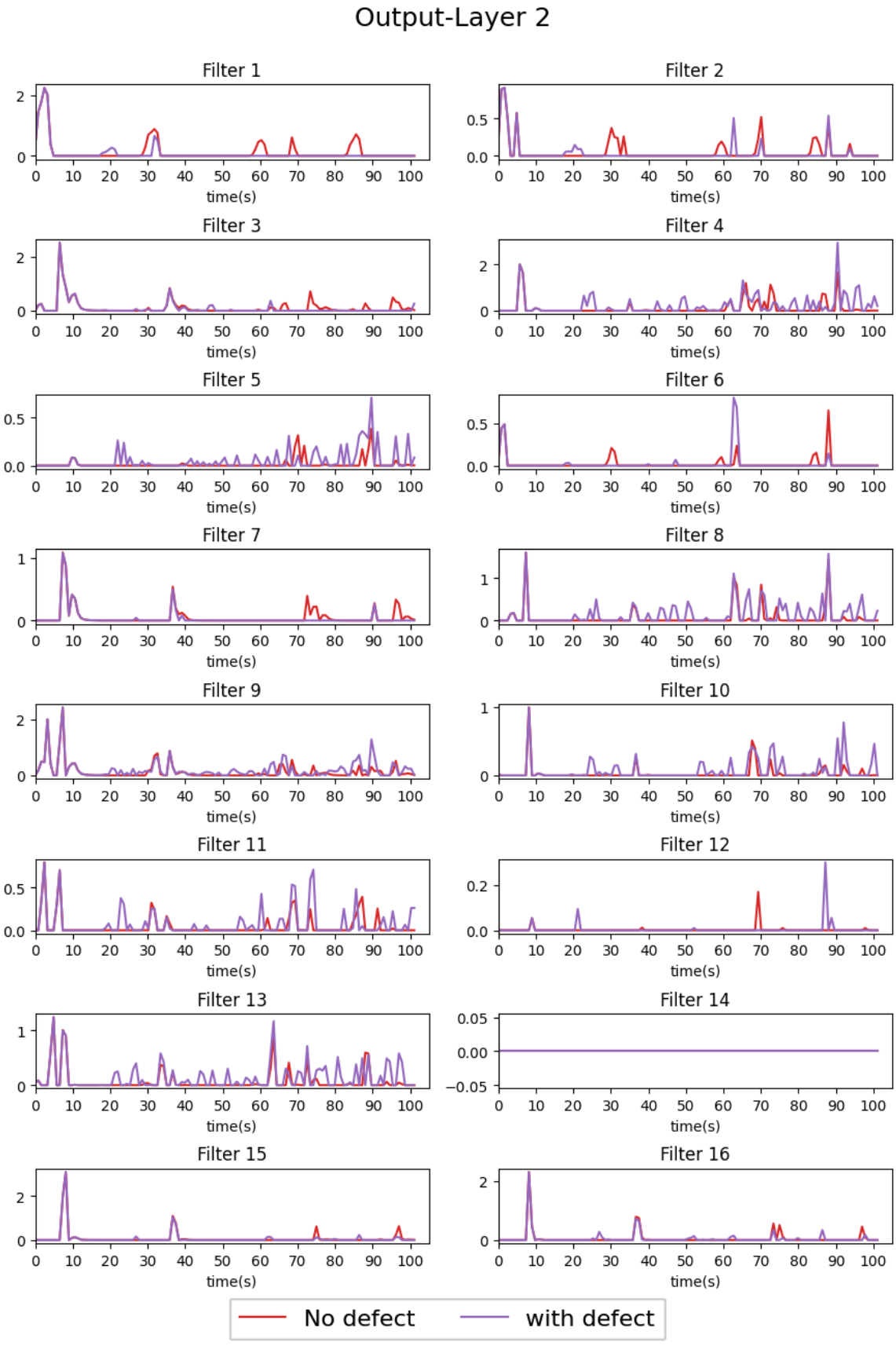


Figure 18: Output of the second convolutional layer related to the samples with and without defect .

After the convolutional layer 2 and its max pooling, the output of this block has to pass through the Flatten layer. This is necessary to unroll all the outputs of the filters in one dimension tensor, which will fed as input to the dense layer.

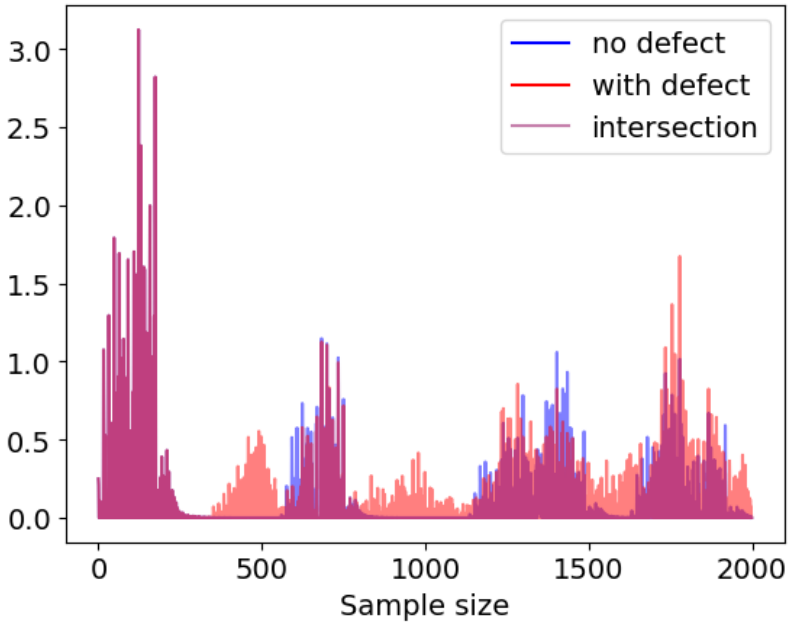


Figure 19: Output of the Flatten layer for the samples with and without defect .

Figure 19 shows the output of the Flatten layer, in blue for the non-defective sample and in orange for the one with a defect. The interesting part here is the intersection between them in red. It shows what they have in common and at the same time shows the features that the dense layers will use to differentiate them in the classification process.

5.2 Robustness Analysis

Robustness is essential to a model, and a good way to test it is by applying noise to data. If the model is robust, it can be very beneficial not only because of the accuracy and generalization but also because it opens the possibility to introduce randomness in the data, in other words, providing an easy way to augment the data.

5.2.1 Inserting noise into the data

To investigate the model robustness, the training and test data were fed with noise, in two manners, named relative and destructive. The first is correlated to the signal itself, which could represent measurement errors. The second, on the other hand, signifies errors uncorrelated to the data, for instance, due to background noise. The Equations 1 e 2 define respectively the relative and destructive noise.

$$Xr_n = x_n(1 + A.u_n) \tag{1}$$

$$Xd_n = x_n + \max(x_n).A.u_n \tag{2}$$

Where x_n is the data,A is the amplitude of the noise, u_n is the noise generated from a uniform distribution, between -1 and 1, and the index n indicates that the noise is applied for each sample individually and independently.
Figure 20 showcases both instances. On the left, an example of a sample is presented, with a relative noise of 60%. On the right, the same sample is depicted, but this time with a destructive noise of 60%. Notably, in the first instance, the information remains discernible despite minor deviations. In contrast, the information in the second case appears to be almost entirely destroyed.

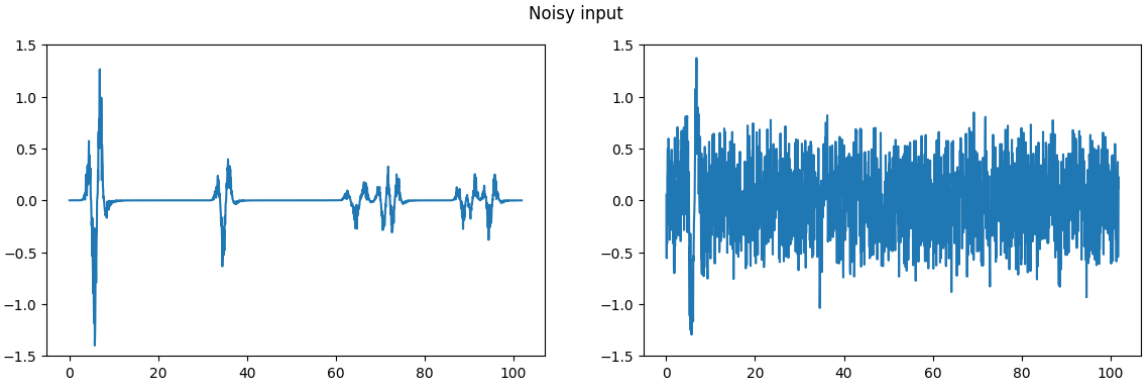


Figure 20: Noisy samples with no defect, on the left with relative noise, and on the right with destructive noise .

Interesting to observe the network's response to these noises. The observation is directed towards the output of the initial convolutional layer. Figure 21 illustrates the four outputs associated with each filter within the context of the relative noise scenario. Evidently, it is discernible that the signal remains substantially unaltered, closely resembling the input signal. Consequently, an analogous level of accuracy to that observed with clean data is anticipated. The preservation of the signal's integrity can be ascribed to the convolutional layer's behavior—its utilization of both the stride and convolution operation is an effective denoising mechanism or regularization.

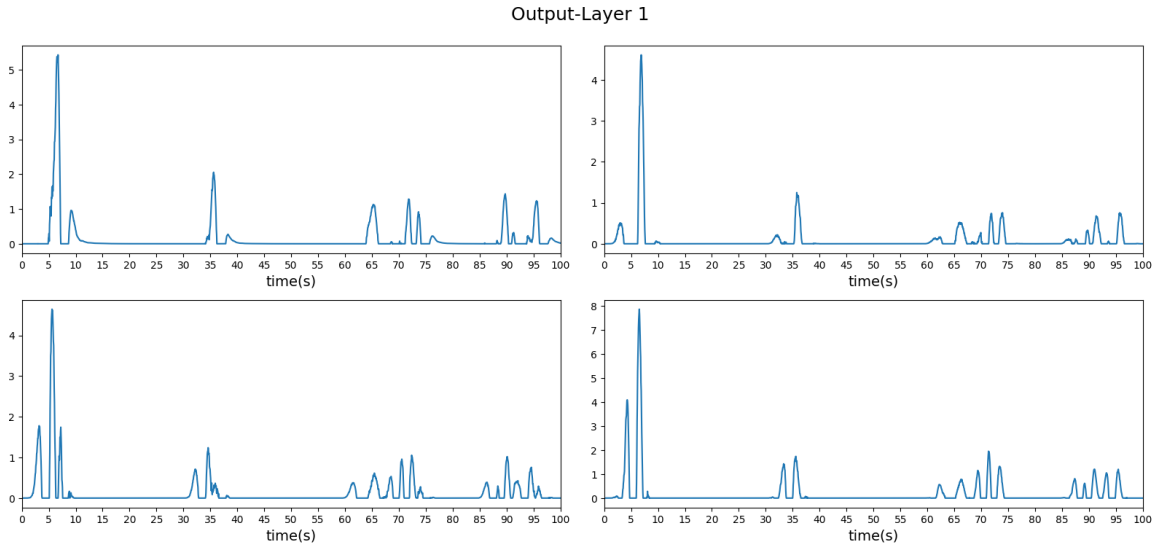


Figure 21: Output of the first convolutional layer from an input sample fed with relative noise.

On the other hand, Figure 22 shows that the sample subjected to destructive noise exhibits a loss of information, notwithstanding the denoising capability inherent to the convolutional neural network. Nevertheless, it is observable that a portion of the information has been recovered, exemplified by the prominence of certain peaks that previously were masked by the noise.

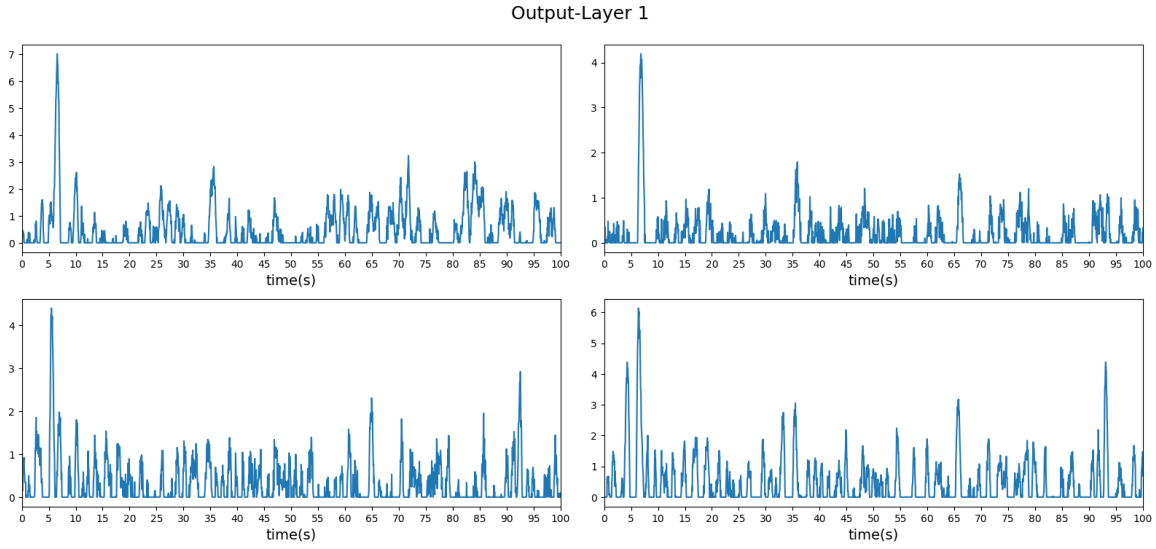


Figure 22: Output of the first convolutional layer from an input sample fed with destructive noise.

5.2.2 Training the model with noisy data

Training the model with noisy data can easily lead to overfitting if it is not controlled. In order to do that, an analysis of the amount of noise and which type can be inserted in the data was made.

Basically, the model was trained and tested separately with both noises, varying the amplitude of the noise in the training and testing.
Figure 23 shows on the x-axis the noise amplitude applied to the test data, while the colormap shows the noise amplitude that the model was trained.
As expected, the model has very good performance even in very high noises, in most of the trained amplitudes the accuracy keeps at 100%, with just a small decrease of almost 5% in models trained with low or no noise and tested with the noise amplitutes higher than 80%.

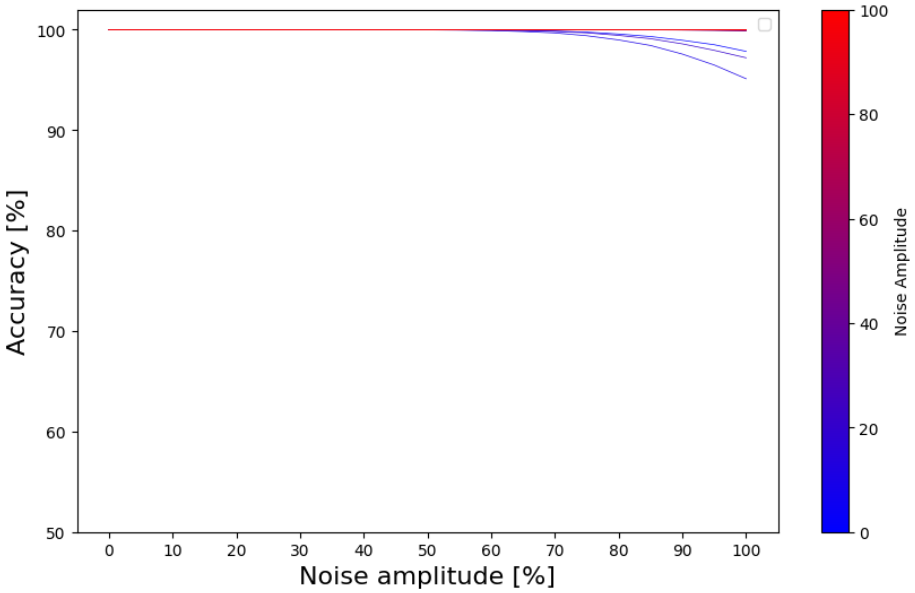


Figure 23: Accuracy according to the training and test noise amplitude .

This noise can be a very good implementation to data augmentation since I can add randomness to the data, that can be repeated due to duplication or symmetry, and keep high accuracy.

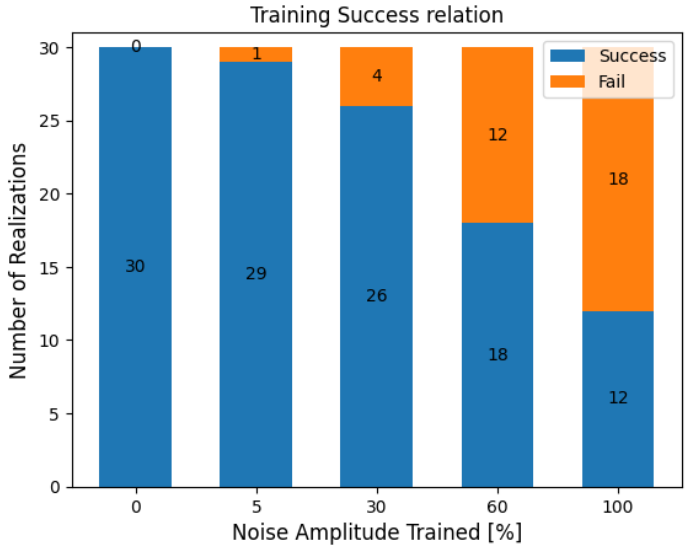


Figure 24: The relation between the increase of the destructive noise amplitude and the failed models .

Conversely, the destructive noise is more complex to deal with. In a pre-training test, it was observed that sometimes the model did not converge, in the sense that the training had failed to find a local minimum that properly satisfied the problem, in other words, the accuracy kept constant around 50%. In a binary classification, this means that the model failed to classify the data correctly. To show that 30 realizations of training were made for each 5 different noise amplitudes (0%, 5%, 30%, 60%, 100%). Each time a model was trained the accuracy of the last epoch was collected. The threshold of the convergence was set as an accuracy higher than 60%. At this point, using the *Mésocentre* was very necessary due to the number of data and iterations in the process. Having the possibility to run this in parallel in a high-performance computing server made this analysis possible. Figure 24 shows the relation between the training noise and the failing of the model. It can be observed that the higher the noise, the higher the chance of not converging to a local minimum with a good fit.

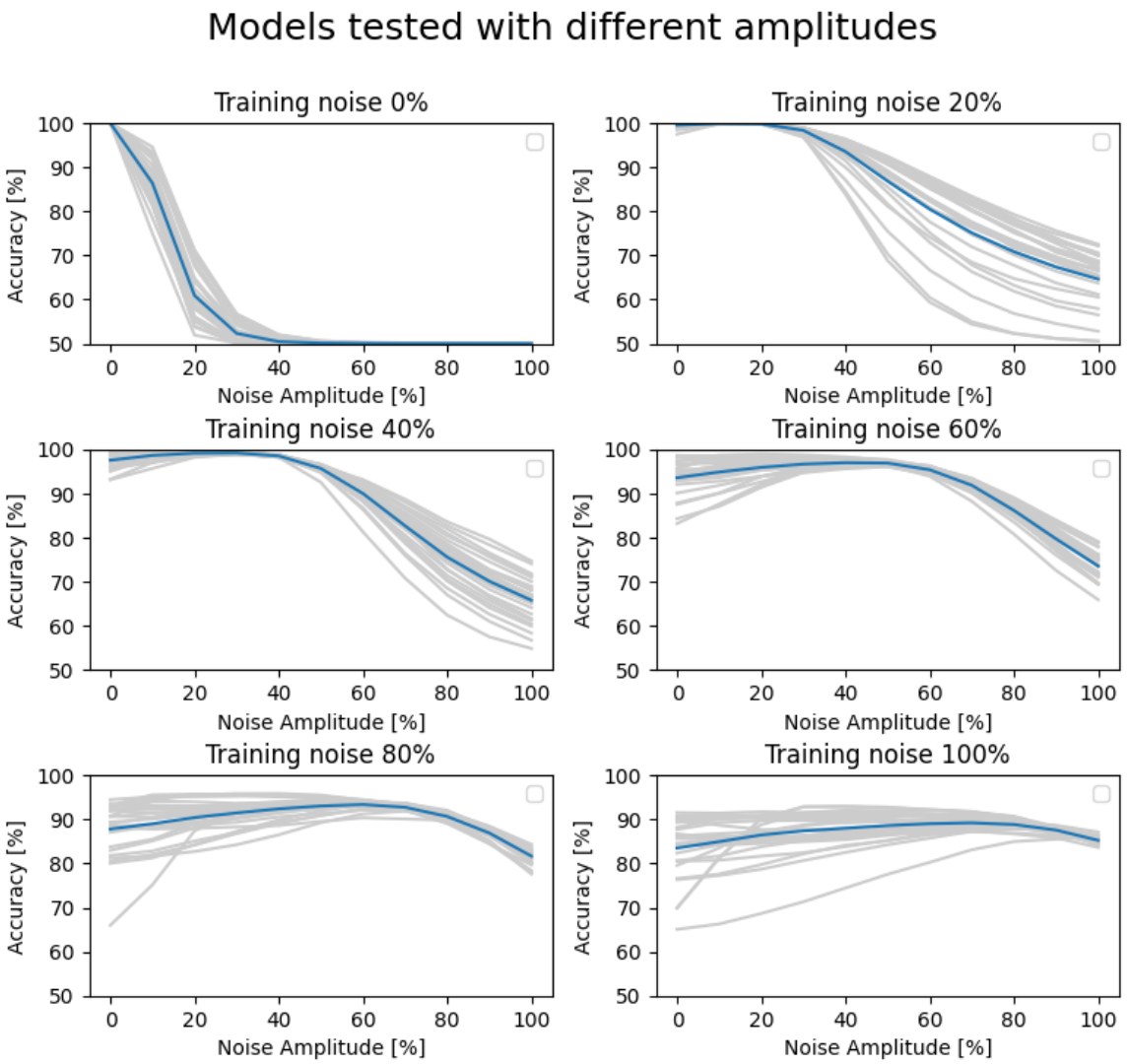


Figure 25: Defect selection process example to provide more reliable data .

Also, it was observed in the pre-test that the models, due to the randomness applied to the test data, could have different accuracy, although with low variance, however do to the

randomness of the training data and the ill-posed property of the neural network, different realizations of models trained with the same amplitude noise could have different accuracy responses for the same test sample.

Thus, in order to analyze more precisely, several realizations of training were made, with different amplitudes(0%, 20%, 40%, 60%, 80%, 100%), and 25 of them that had success in the training were stored. These models were tested with destructive noisy data, the same way the relative ones were tested.

Figure 25 shows in gray the realizations for each amplitude-trained model and in blue the average of them. Thus, the confidence interval of all training can be observed. The clean model (0%) is very sensible to the destructive noise, since increasing the amplitude the accuracy drops fast. In the models trained with noise, it can be observed a region close to the trained amplitude where the accuracy is maximum. Also, the confidence interval seems smaller, with the realizations squeezing around this maximum accuracy. At the same time, the models trained with higher noise lose their accuracy for low or no-noise samples.

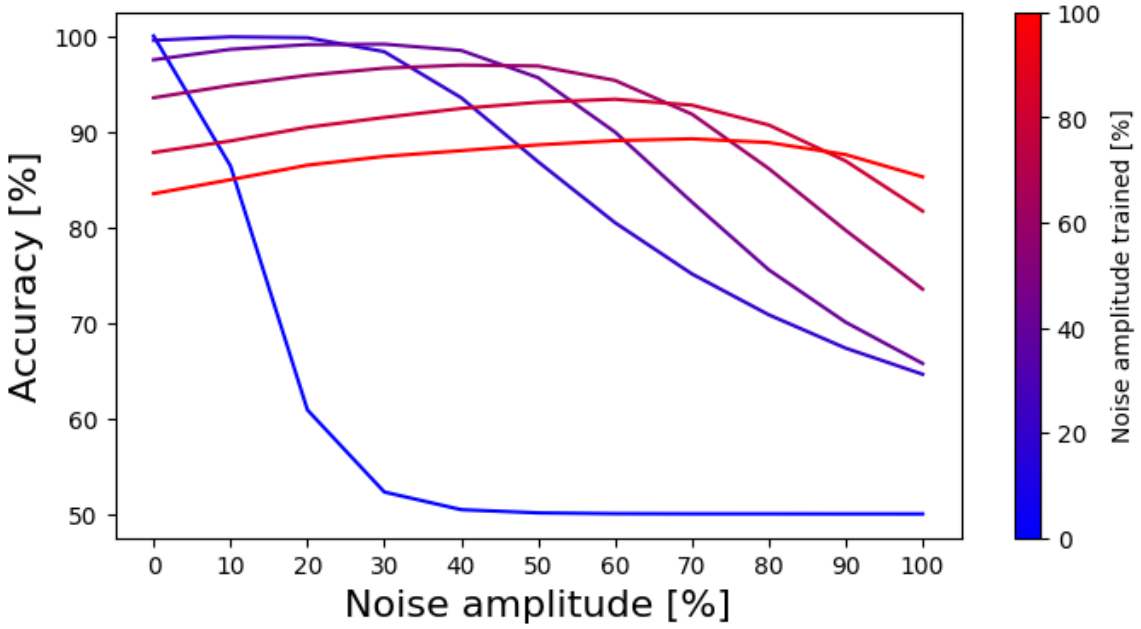


Figure 26: Accuracy average of the realizations trained with different destructive noise.

It can be observed in Figure 26 the maximum accuracy moving along the test noise amplitude(x-axis), as well as, the loss of accuracy in low noise tests for the models trained with high noises. Besides that, the relation between the maximum accuracy in the test and the noise amplitude trained seems to be very linear.

To evaluate that the noise amplitude where the maximum accuracy was obtained was plotted related to the noise amplitude the model was trained. In Figure 27 this linear relation can be noticed, with the maximum accuracy being around 70% of the trained noise.

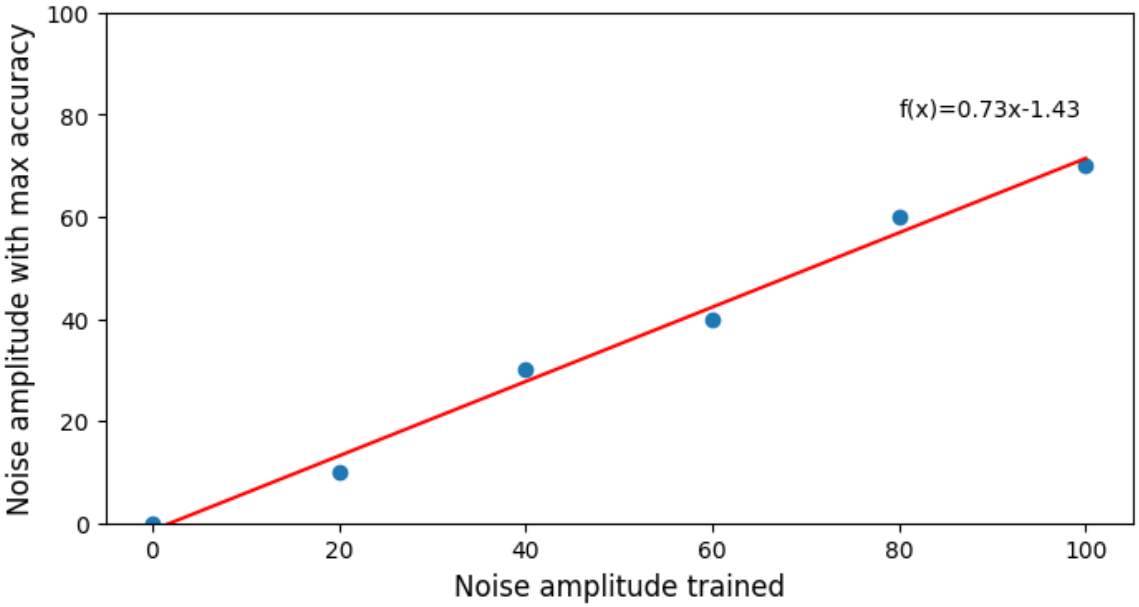


Figure 27: Noise amplitude with maximum accuracy related to the noise amplitude the model was trained .

Also, it is important to observe how the models trained with destructive noise perform in a test with clean data. Thus, Figure 28 shows that models trained until 20% of destructive noise amplitude have very high accuracy. Therefore, it is safe to state that the model can be trained with a maximum 20% destructive noise to improve robustness and the data can be augmented safely using relative noise amplitude until 50%.

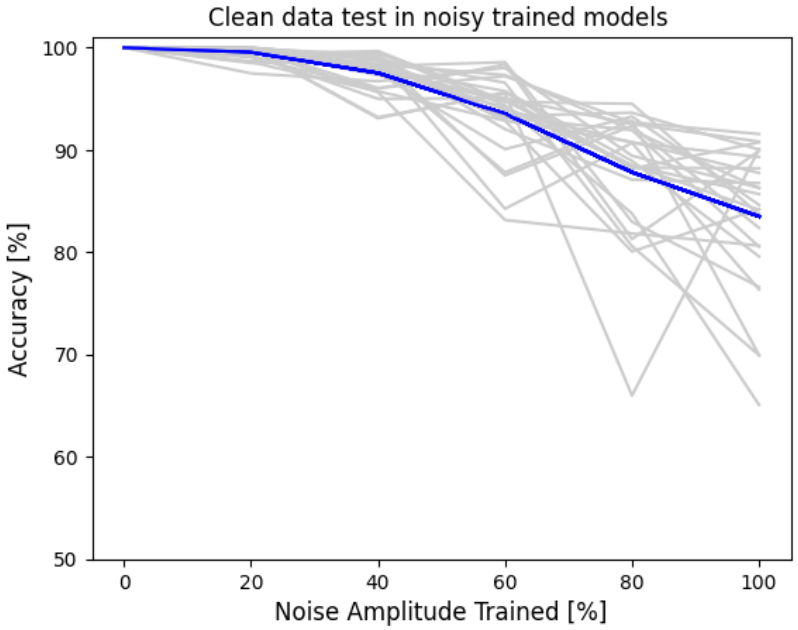


Figure 28: Models trained with destructive noise tested with clean data.

5.3 Multi-defect case

This project had the intention to investigate the use of ANNs to detect a defect in a specific domain. To extend this study, a trial to detect multiple defects was done, aiming if not solve the case to bring some insights and hypothesis for future works. The idea was to work with the same architecture and evaluate its performance. Two approaches came up to tackle this problem; the first one was to use the same architecture and train individually for each case as a binary classification, for instance, the pair no defect, two defects, and so on, until a limit. Then, write a code to process the test data to evaluate the samples in parallel in each model, returning the highest value predicted as the prediction. The second option was just to change the binary classification, for a multi-classification using the function softmax.
Both of the approaches required new datasets to be generated and labeled. Thus, the second option was chosen due to the time necessary to generate the data and the opportunity to work with a new method of classification.
The dataset was generated almost the same way as described in Section 2. The only change was in the defect generation; instead of just one defect, multiple defects were randomly generated. A total of 5 independent defects could be created. It was observed that in some cases the defect could allocate in a sensor position, and generate a "NaN" response. In these cases, the sample was removed in the data post-processing.
Another observation to keep in mind is that each time a defect is added to the total number of defects, the problem gets more complex and needs more samples to be representative. For instance, in the rectangular model from this project, one defect has 450 possible locations, while for 2 defects it would be 101025 possible location combinations. This can make a dataset very complex to work with, and that can underfit or overfit during the training, and have low accuracy in the test phase.
To tackle this problem and consider the time necessary to generate the data, the simulation was executed with fewer sources and sensors(12 and 16, respectively) but more realizations, each producing a new set of defects. In total, 20 realizations were made for each defect case, producing a dataset of 23040 samples, or 3840 for each defect configuration. From that, 18342 were separated randomly and shuffled to be the training dataset, resting 4698 samples for the test dataset.
Thus, in the ANN mode, the binary classifier using sigmoid was substituted by a softmax multiclassifier layer with 6 neurons. The model was compiled using sparse categorical cross-entropy as a loss function and sparse categorical accuracy as the metric, which works like the accuracy method used before, but counting the categories matched.

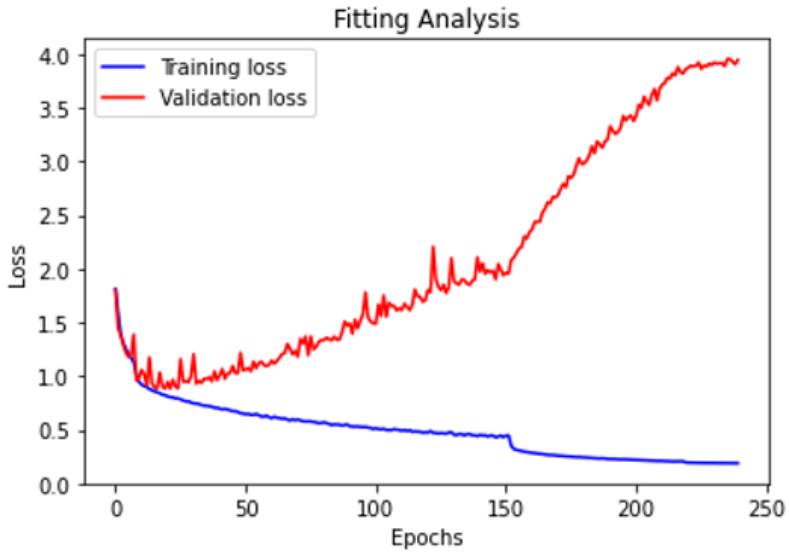


Figure 29: Loss curve of the training of the multi defect case.

Figure 29 depicts the loss in the training process. As the validation case goes up after a certain epoch, an overfit can be considered. Several reasons can lead to that, for example, too many parameters, lack of data, training process, etc. It is necessary to have a look in the test phase to have more hints about the problem.

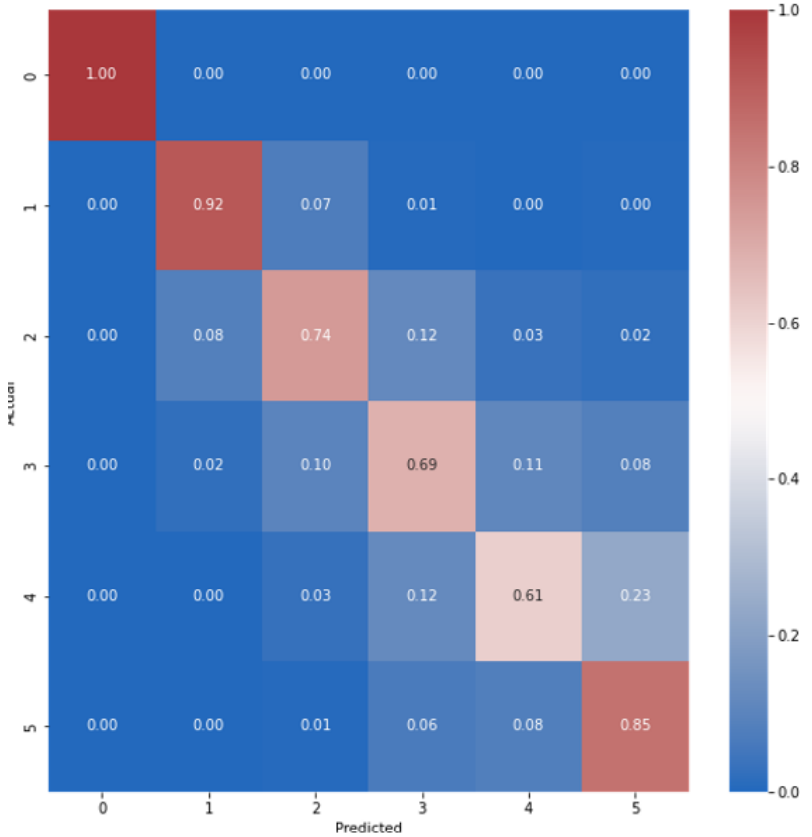


Figure 30: Confusion matrix for the classification of the number of defects in a rectangular domain.

Figure 30 shows the confusion matrix, having on the x-axis the predicted classification and on the y-axis the labeled ones. For the cases with no defect and 1 defect, the model has high accuracy, while for 2,3,4 the accuracy drops a lot. As mentioned before, the accuracy could be low due to the number of defects increasing. However, for the case with 5 defects, the accuracy was very high. Two hypotheses came up. First, there was more data for 5 defects in the training data than for 2,3 and 4. Second, although the data can get more complex with the increase in the number of defects, this can lead to an increase of reflections in the simulations that can go faster to a kind of diffuseness in which the sensors' responses will tend to be statistically more similar.

Thus, to try to improve these results, more data was produced, and the model was changed with some tools that help avoid overfitting, inserting batch normalization and dropout(0.2) in each layer, and L2 regularization (0.001) in the kernel of the convolutional layers. Also, 20 more realizations were made for each defect case, producing a dataset of 46080 samples, or 7680 for each defect configuration. From that, 36684 were separated randomly and shuffled to be the training dataset, resting 9216 samples for the test dataset.

Figure 31 shows the improvement in the validation loss, with but still overfitting.

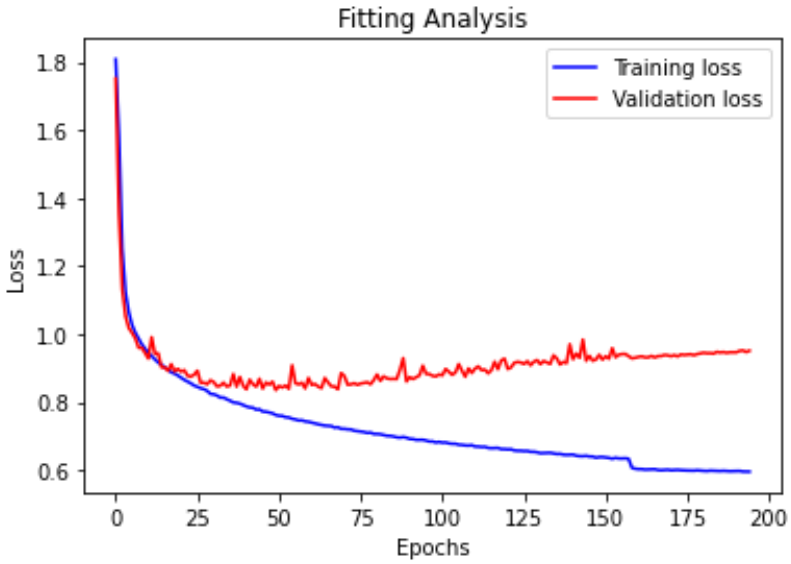


Figure 31: Loss curve of the training of the multi defect case after changing the model and producing more data.

In Figure 32 is the confusion matrix for the new configuration. Some accuracies dropped a bit, but this can happen between the same model training many times. However, it can be due to new data, that added more complexity, and as still overfitting the generalization is still not good. As we had less data in the previous model, that can be the reason for higher accuracy.

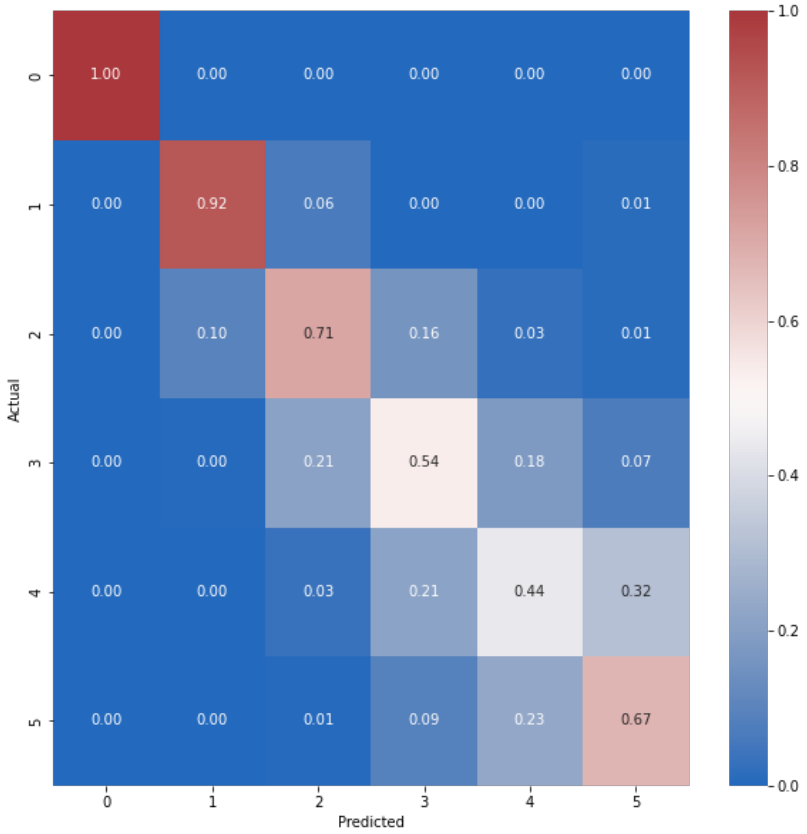


Figure 32: Confusion matrix for the classification of the number of defects in a rectangular domain after changing the model and producing more data.

The case with 5 defects still has better accuracy than the ones with 3 and 4 defects, but now it is lower than 2 defects. There was an unfair distribution of the training data, with a different number of samples for each case. This led to the situation that the higher the number of samples, the higher the accuracy.
So, the alternative here is to test with fair data distribution and generate more data, then evaluate if there is any improvement.

6 Conclusion

This project aimed to explore the feasibility of defect detection using time series signals. The goal was to create a synthetic dataset through direct methods in order to train a neural network to tackle an inverse problem: detect the presence of defects.
The ANN model created was able to detect the presence of a single square defect in a rectangular and homogeneous domain. The accuracy reached 100% for this specific case. Alteration in the geometry or the fundamental frequency of the source in the simulation made the accuracy drop to less than 60%. This was expected because 1d convolutional neural networks can generalize that very well, but they need to be in the same distribution, and any of these changes can alter it. Thus, it is important to understand these limitations, and, in future works try to overcome them. Another limitation of 1D convolutional neural networks is that the sample size is fixed, which can cause many problems if data resampling is needed to input them into the network. Also, this project tried to understand the learning process of the designed ANN. It is a very complex task, that many researchers skip, focusing on the result. This can be a wise decision because seeking to comprehend this can be very time-consuming for a small reward. From this part, some insights came up, but they need further studies to verify them. For instance, the second convolutional layer is possibly responsible for extracting the features that will provide information to the classifier to detect the presence or not of the defect. The most exciting outcome of this project came from the robustness analysis, where it was observed that 1d convolutional layers can act as regularizers and denoise the input signal, depending if the noise is correlated or uncorrelated. If it is uncorrelated, the model can be trained with noise until the amplitude of 20% and improve the robustness and generalization without influence significantly the accuracy. Besides that, a linear relation was observed between the noise amplitude that the model was trained and which test noise amplitude reached the maximum accuracy. This proportion is approximately 70% of the trained noise amplitude. An outcome that comes from the robustness is that it may be possible for the neural network to learn uncorrelated noise without overfitting. This can be very interesting for non-destructive testing using seismic waves, due to the number of uncorrelated noise that can be found in their measurements. The last part of the project was related to testing the possibility of detecting multiple defects and classifying them by the number of defects. Sadly, this part was impossible to develop completely due to the end of the internship. However, despite the overfit, there is some indication that it is possible to detect multiple defects but the data necessary can be very expensive to generate. This project started with a simple problem of defect detection and brought some insights for future works. The first, is a deeper study of noise robustness, expanding to uncorrelated noises that can be found in real measurements. Another research line is multiple defect detection, where another architecture can be explored to get more satisfactory results.

Appendix

.

Fundamental concepts in deep learning

.

Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) consist of fundamental elements that collectively enable their ability to learn patterns and make predictions from data. These elements work in harmony to process information and adjust parameters during the training process. Understanding these elements is essential for grasping how ANNs function and produce meaningful outcomes:

1. **Neurons or Nodes:** Neurons are the basic computational units of an ANN. They receive input, perform calculations, and generate output. Neurons mimic the behavior of biological neurons, aggregating inputs, applying an activation function, and passing the result to the next layer.

2. **Layers:** ANNs are structured into layers, each serving a distinct role. The input layer receives raw data, hidden layers process intermediate representations, and the output layer generates predictions or classifications. The number of hidden layers and their neurons define the network's depth and complexity.

3. **Weights:** Connections between neurons are represented by weights. These weights determine the strength of influence one neuron has on another. During training, the network adjusts these weights to minimize errors and improve performance.

4. **Activation Functions:** Activation functions introduce non-linearity to the neural network, allowing it to model complex relationships. Common activation functions include sigmoid, ReLU (Rectified Linear Unit), and tanh (hyperbolic tangent).

5. **Bias:** Bias terms are added to each neuron's input before the activation function is applied. Bias enables the network to account for offset and better approximate target functions.

6. **Connections:** Connections or edges between neurons carry weighted information. Data flows through these connections from input to output, facilitating information processing and feature extraction.

7. **Feedforward Propagation:** During feedforward propagation, data travels from the input layer through the hidden layers to the output layer. Neurons compute weighted sums and activation values, producing predictions.

8. **Loss or Cost Function:** The loss function quantifies the difference between predicted and actual values. The network aims to minimize this loss during training to enhance accuracy.

9. **Backpropagation:** Backpropagation is the core learning algorithm in ANNs. It computes gradients of the loss with respect to weights, facilitating weight updates that minimize the loss over iterations.

10. **Optimization Algorithms:** Optimization algorithms, like gradient descent, guide the adjustment of weights to minimize the loss function efficiently. Variations such as Adam and RMSprop optimize learning rates and enhance convergence.

11. **Epochs and Batch:** Training occurs over epochs, with each epoch representing one complete pass through the training data. Batch training divides data into smaller subsets for weight updates, enhancing efficiency.

12. **Hyperparameters:** Hyperparameters are settings that determine how the network learns. Learning rate, number of hidden layers, neurons per layer, and batch size are examples of hyperparameters that impact training and performance.

1D Convolutional Neural Networks (CNNs)

:

The 1D Convolutional Neural Networks (CNNs) constitute the building blocks enabling these networks to process sequential data efficiently. These elements collectively shape the network's behavior, enhancing its ability to learn and extract meaningful features from the input data:

1. **Convolutional Layers:** The core of 1D CNNs, these layers utilize filters to perform convolutions over the input data. These filters slide across the data, detecting local patterns and extracting relevant features.

2. **Activation Functions:** Applied after convolutions, activation functions introduce non-linearity, enabling the network to model complex relationships. Popular choices include ReLU (Rectified Linear Unit) and sigmoid functions.

3. **Pooling Layers:** These layers reduce the spatial dimensions of the data, enabling the network to capture essential features while reducing computational complexity. Max pooling and average pooling are common techniques.

4. **Fully Connected Layers:** These layers connect all neurons from the previous layer to those in the current layer. They integrate extracted features for final predictions or classifications.

5. **Padding:** Padding adds additional elements to the input sequence, maintaining its length after convolutions and preserving important edge information.

6. **Strides:** Strides determine how the filter moves across the input sequence during convolutions. Different strides influence the degree of overlap and the reduction in spatial dimensions.

7. **Dropout:** Dropout is a regularization technique that randomly deactivates neurons during training, preventing overfitting and enhancing generalization.

References

[1] Samira Gholizadeh. A review of non-destructive testing methods of composite materials. *Procedia structural integrity*, 1:50–57, 2016.

[2] Sanjay Kumar and DG Mahto. Recent trends in industrial and other engineering applications of non destructive testing: a review. *International Journal of Scientific & Engineering Research*, 4(9), 2013.

[3] Fausto Pedro García Márquez, Andrew Mark Tobias, Jesús María Pinar Pérez, and Mayorkinos Papaelias. Condition monitoring of wind turbines: Techniques and methods. *Renewable energy*, 46:169–178, 2012.

[4] DM McCann and MC Forde. Review of ndt methods in the assessment of concrete and masonry structures. *Ndt & E International*, 34(2):71–84, 2001.

[5] Bing Wang, Shuncong Zhong, Tung-Lik Lee, Kevin S Fancey, and Jiawei Mi. Non-destructive testing and evaluation of composite materials/structures: A state-of-the-art review. *Advances in mechanical engineering*, 12(4), 2020.

[6] Sebastiano Foti, Carlo G Lai, Glenn J Rix, and Claudio Strobbia. *Surface wave methods for near-surface site characterization*. CRC press, 2014.

[7] Peter Hatherly. Overview on the application of geophysics in coal mining. *International Journal of Coal Geology*, 114:74–84, 2013.

[8] RA Smith. Composite defects and their detection. *Materials science and engineering*, 3(1):103–143, 2009.

[9] James G Fujimoto, Costas Pitris, Stephen A Boppart, and Mark E Brezinski. Optical coherence tomography: an emerging technology for biomedical imaging and optical biopsy. *Neoplasia*, 2(1-2):9–25, 2000.

[10] Ivan Graham, Ulrich Langer, Jens Melenk, and Mourad Sini. *Direct and inverse problems in wave propagation and applications*, volume 14. Walter de Gruyter, 2013.

[11] Martin Schickert, Martin Krause, and Wolfgang Müller. Ultrasonic imaging of concrete elements using reconstruction by synthetic aperture focusing technique. *Journal of materials in civil engineering*, 15(3):235–246, 2003.

[12] Martin Genzel, Jan Macdonald, and Maximilian März. Solving inverse problems with deep neural networks–robustness included? *IEEE transactions on pattern analysis and machine intelligence*, 45(1):1119–1134, 2022.

[13] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

[14] Mathias Fink, William A Kuperman, Jean-Paul Montagner, and Arnaud Tourin. *Imaging of complex media with acoustic and seismic waves*. Springer Science & Business Media, 2003.

[15] Peter M Shearer. *Introduction to seismology*. Cambridge university press, 2019.

[16] Brian Kennett. *Seismic wave propagation in stratified media*. ANU Press, 2009.

[17] Robert E Sheriff and Lloyd P Geldart. *Exploration seismology*. Cambridge university press, 1995.

[18] Jin Wang and Ta-Liang Teng. Artificial neural network-based seismic detector. *Bulletin of the Seismological Society of America*, 85(1):308–319, 1995.

[19] Christiane Maierhofer, Hans-Wolf Reinhardt, and Gerd Dobmann. *Non-destructive evaluation of reinforced concrete structures: Non-destructive testing methods*. Elsevier, 2010.

[20] E Martinho and A Dionísio. Main geophysical techniques used for non-destructive evaluation in cultural built heritage: a review. *Journal of Geophysics and Engineering*, 11(5):053001, 2014.

[21] E Safak and K Hudnut. Real-time structural monitoring and damage detection by acceleration and gps sensors. In *8th US national conference on earthquake engineering, San Francisco, California*, volume 10. Citeseer, 2006.

[22] Andrei-Alexandru Tulbure, Adrian-Alexandru Tulbure, and Eva-Henrietta Dulf. A review on modern defect detection models using dcnns–deep convolutional neural networks. *Journal of Advanced Research*, 35:33–48, 2022.

[23] Hui Lin, Bin Li, Xinggang Wang, Yufeng Shu, and Shuanglong Niu. Automated defect inspection of led chip using deep convolutional neural network. *Journal of Intelligent Manufacturing*, 30:2525–2534, 2019.

[24] Masoud Jalayer, Carlotta Orsenigo, and Carlo Vercellis. Fault detection and diagnosis for rotating machinery: A model based on convolutional lstm, fast fourier and continuous wavelet transforms. *Computers in Industry*, 125:103378, 2021.

[25] Xun Cheng and Jianbo Yu. Retinanet with difference channel attention and adaptively spatial feature fusion for steel surface defect detection. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2020.

[26] Oleg S Amosov, Svetlana G Amosova, and Ilya O Iochkov. Defects detection and recognition in aviation riveted joints by using ultrasonic echo signals of non-destructive testing. *IFAC-PapersOnLine*, 54(1):484–489, 2021.

[27] Shaun W Lawson and Graham A Parker. Intelligent segmentation of industrial radiographic images using neural networks. In *Machine Vision Applications, Architectures, and Systems Integration III*, volume 2347, pages 245–255. SPIE, 1994.

[28] Haoyu Xu, Zhenqi Han, Songlin Feng, Han Zhou, and Yuchun Fang. Foreign object debris material recognition based on convolutional neural networks. *EURASIP Journal on Image and Video Processing*, 2018:1–10, 2018.

[29] Chinedu I Ossai. Corrosion defect modelling of aged pipelines with a feed-forward multi-layer neural network for leak and burst failure estimation. *Engineering Failure Analysis*, 110:104397, 2020.

[30] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

[31] 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.

[32] Google Brain. Tensorflow. https://www.tensorflow.org/. Retrieved Aug 12, 2023.

[33] Jean-Paul Ampuero. SEMLAB 2.0. https://www.mathworks.com/matlabcentral/fileexchange/6154-semlab. MATLAB Central File Exchange. Retrieved May 22, 2023.

[34] Anthony T Patera. A spectral element method for fluid dynamics: laminar flow in a channel expansion. *Journal of computational Physics*, 54(3):468–488, 1984.