Chapter 13

Application of Deep Learning Techniques for the Impedance-based SHM to the Oil & Gas Industry

Chapter details

Chapter DOI:

https://doi.org/10.4322/978-65-86503-83-8.c13

Chapter suggested citation / reference style:

Rezende, Stanley W. F., et al. (2022). "Application of Deep Learning Techniques for the Impedance-based SHM to the Oil & Gas Industry". *In* Jorge, Ariosto B., et al. (Eds.) *Fundamental Concepts and Models for the Direct Problem*, Vol. II, UnB, Brasilia, DF, Brazil, pp. 349–385. Book series in Discrete Models, Inverse Methods, & Uncertainty Modeling in Structural Integrity.

P.S.: DOI may be included at the end of citation, for completeness.

Book details

Book: Fundamental Concepts and Models for the Direct Problem

Edited by: Jorge, Ariosto B., Anflor, Carla T. M., Gomes, Guilherme F., & Carneiro, Sergio H. S.

Volume II of Book Series in:

Discrete Models, Inverse Methods, & Uncertainty Modeling in Structural Integrity

Published by: UnB City: Brasilia, DF, Brazil Year: 2022

DOI: https://doi.org/10.4322/978-65-86503-83-8

Application of Deep Learning Techniques for the Impedance-based SHM to the Oil & Gas Industry

Stanley W. F. Rezende[1*], José dos Reis V. Moura Jr[2],
José Waldemar da Silva[4], Diogo S. Rabelo[3], Quintiliano S. S. Nomelini[4],
Roberto M. Finzi Neto[1], Carlos A. Gallo[1], Julio E. Ramos[5]

[1] Post-Graduate Program – Mechanical Engineering, Federal University of Uberlandia, Brazil. e-mail: stanley_washington@ufu.br; finzi@ufu.br; gallo@ufu.br
[2] Post-Graduate Program – Modeling and Optimization, Fed. University of Catalao, Brazil. e-mail: zereis@ufcat.edu.br
[3] Post-Graduate Program – Industrial Engineering, Fed. University of Goias, School of Sciences and Technology, Aparecida de Goiania, Brazil. e-mail: diogo.rabelo@ufg.br
[4] Faculty of Mathematics - Federal University of Uberlandia, Brazil, e-mail: zewaldemar@ufu.br; quintiliano.nomelini@ufu.br
[5] Petrobras, Petroleo Brasileiro S.A., R&D Center (CENPES), Brazil. e-mail: julio.ramos@petrobras.com.br

*Corresponding author

Abstract

This chapter presents some basic concepts about some fundamental Deep Learning techniques currently used in the data processing. Next, the use of these techniques to aid decision-making in Electromechanical Impedance-based Structural Health Monitoring (ISHM) is presented. Initially, using a CNN to classify structural damage in specimens is evaluated, eliminating the need for temperature compensation. Then, an LSTM network prediction model of the evolution of an accelerated corrosive process (HCl acid) in specimens is presented. Finally, a model based on CNN is carried out as a case study of thickness loss in a real fuel storage tank plate.

Keywords: Deep Learning; Impedance-based Structural Health Monitoring; Oil and Gas Industry

1. Introduction

Human beings use several types of mechanical structures daily to carry out their tasks. These structures, when in operation, are susceptible to failure due to wear conditions, fatigue, or impacts of their mechanical components (Moura Jr and Steffen Jr [2006], Rezende, Barella and Moura Jr [2020]). Therefore, for such facilities to perform their activities correctly, the failures present in the mechanical system must be identified and repaired promptly, which makes integrity monitoring processes a critical aspect nowadays (Palomino *et al.* [2011], Eleftheroglou *et al*. [2018], Ghazvineh *et al.* [2021]).

In addition, it is known that incorrect identification of the presence of damage can have serious economic and safety consequences (Ghazvineh *et al.* [2021]). In this sense, the maintenance of systems considered critical becomes even more fundamental, and, consequently, different inspection and damage assessment techniques have been researched in recent years. These monitoring and inspection techniques now make up the so-called Structural Health Monitoring (SHM) (Zhao *et al*. [2011], Melville *et al.* [2018], Gulgec, Takáč and Pakzad [2019], Rastin, Ghodrati Amiri and Darvishan [2021]).

SHM formulations and methods, in general, aim to provide or improve issues such as safety, operability, minimization of maintenance and repair costs, logistical efficiency, and increase in useful structural life, among others (Moura Jr and Steffen Jr [2006]). For this, SHM methods commonly employ different types of software and hardware to characterize the systems under study, acquire and manage monitoring data, and evaluate, in the long term, these systems' environmental and operational conditions (Melville *et al.* [2018], Gulgec, Takáč and Pakzad [2019], Rastin, Ghodrati Amiri and Darvishan [2021]).

When a structure's degradation process is continuously monitored, its maintenance can be planned predictively rather than periodically, thus reducing associated costs and avoiding unnecessary system downtime (Eleftheroglou *et al*. [2018]). Therefore, this situation requires adopting a damage identification process with rapid prognostic capacity. For certain types of structures, it is necessary to evaluate the integrity and functioning conditions of the system in real-time (Zhao *et al.* [2011], Rastin, Ghodrati Amiri and Darvishan [2021]).

The electromechanical impedance-based SHM method (ISHM), in this context, presents itself as a promising technology to reduce structural risks and maintenance costs. This condition, in turn, is given by the ease that this method offers in its implementation and the possibility of integration with advanced data processing techniques (Moura Jr and Steffen Jr [2006], Leucas [2009], Palomino *et al.* [2011], Na and Baek [2018], Nomelini *et al.* [2021]).

In ISHM, piezoelectric transducers are usually integrated into the mechanical structures. Later, an electronic circuit becomes responsible for monitoring the variation of the electrical impedance of each chip. This procedure is carried out over a wide range of excitation frequencies (usually greater than 30kHz) (Moura Jr and Steffen Jr [2006], Leucas [2009], Na and Baek [2018], Rezende, Barella and Moura Jr [2020]).

As each piezoelectric transducer is mechanically coupled to the structure under investigation, variations in the electrical impedance values suggest a possible presence of structural damage (Nomelini *et al.* [2021]). However, to make a correct measurement of structural failure the method needs a large volume of data for evaluation, which causes a significant increase in the complexity of the abstraction process (Palomino *et al.* [2011], Silva *et al.* [2018], Rezende [2021]).

Due to this increase in computational complexity, recent SHM studies have sought to reconcile integrity monitoring methods with innovative and functional tools for pattern abstraction and classification (Verstraete *et al*. [2017], Ghazvineh *et al.* [2021]). The use of pattern abstraction techniques in the monitoring data sets aims to delimit a causal relationship between the different integrity states of the structures under study (Gulgec, Takáč and Pakzad [2017]). Thus, once the response behavior of the

system is assimilated, it is possible to detect promptly the existence of variations in this behavior, which indicate a possible presence of damage in such structures (Palomino *et al.* [2011], Melville *et al.* [2018]).

In this sense, it should be noted that interpreting the acquired data is more important than collecting the monitoring information. Therefore, pattern abstraction and classification techniques used in conjunction with SHM methods must have high detection power and precision, factors that must still be achieved together with rapid data processing (Gulgec, Takáč and Pakzad [2017], Freitas, Jafelice and Silva [2021], Rastin, Ghodrati Amiri and Darvishan [2021]).

Furthermore, data acquisitions performed during the processing of a given structure can easily contain thousands of pertinent information and, in this way, different types of tasks and operations can also be performed on them, such as damage level regression, classification of the kind of failures and anomaly detection in the monitored system (Barella [2021], Rezende *et al.* [2020]).

Therefore, to elucidate the practical execution of some of these tasks in the daily context of the SHM, in this chapter, three possible aspects of integration between the deep learning (DL) models and the monitoring method based on electromechanical impedance will be evaluated. For this, two neural architectures, namely CNN and LSTM, will be implemented in the following sections of this chapter. Different practical tests will be carried out to identify the presence and severity of failures in mechanical structures.

At first, a binary damage classification task in aluminum beams will be performed, analyzing the sensitivity of the combination of the CNN network and ISHM technique concerning environmental changes in the data acquisition stage. Then, with the LSTM architecture, a model will be developed to predict the level of corrosion in steel beams, which is caused by hydrochloric acid wear, also helping diagnose the structural life of this type of system.

The third and last experiment will be developed to identify (by the CNN network and ISHM method) the structural health conditions of a damaged tank plate (by grinding), thus determining the level of thickness reduction in the region of interest of the coupled PZT sensor.

Encouraging results were obtained by carrying out the previously mentioned experimental tests. Thus, it becomes possible to verify the employability of deep learning models (more specifically, the CNN and LSTM models) in support of the diagnosis of structural failures in mechanical systems.

That said, and to make the theoretical and practical foundation of this chapter clearer and more concise, it was subdivided into sections as follows: initially, in Section 1, the importance of SHM studies was discussed, in addition to pointing out the need to use of machine learning (ML) and DL models in support of the structural health monitoring methods.

In Section 2, the concepts of machine learning and deep learning are deepened, and then the theoretical formulations of the required techniques are presented separately in their subsections. Section 3, on the other hand, focuses on the applications and results obtained in different case studies. Finally, Section 4 highlights the positive aspects of

the methodology used and describes the conclusions that could be inferred throughout the chapter.

2. Machine Learning and Deep Learning Overview

Machine learning (ML) is a subarea of Artificial Intelligence (AI) focused on developing techniques and computational algorithms capable of extracting features in today's most varied problems (Provost and Kohavi [1998], Freitas, Jafelice and Silva [2021]).

In general, machine learning algorithms are based on creating a programmable inference model, in which, from sampled data, a predictive answer is obtained. In this way, such algorithms automatically find more valuable representations of the data, mapping unknown relations in the space of hypotheses (Verstraete *et al*. [2017], Rezende *et al.* [2020]).

Thus, the learning process in ML is performed inductively, where response patterns are automatically defined through experiences already observed by the mathematical model. Consequently, the main challenge is to formulate intermittent problems where a list of mathematical rules is not easily measurable (Freitas, Jafelice and Silva [2021]).

Some conceptions of AI, previously used to solve complex problems, were based on characteristics specified by human operators, who were also responsible for delimiting all types of knowledge used by the computer. However, a limitation of this type of approach is the difficulty in determining what kinds of features are needed for the abstraction process (Liang, Guixi and Hongyan [2015], Rezende [2021]).

On the other hand, ML techniques eliminate the need to formally define which characteristics the classifier should use through a hierarchical representation of concepts. Thus, knowledge is obtained through experience, enabling the computer to make predictions and classifications of complex ideas by abstracting more straightforward concepts.

In this sense, artificial neural networks (ANNs) have stood out as one of the main ML techniques, contributing at the same time to the development of several areas of knowledge (Menezes *et al.* [2009], Barros, Morais and Fernandes [2017], Gulgec, Takáč and Pakzad [2017], Mhatre *et al.* [2017], Sharma and Singh [2017], Cofre-Martel *et al.* [2019]). Its efficiency is mainly due to the way data processing is performed, in which the applied operations are particularly close to those performed by the human brain (Komijani *et al.* [2017], Melville *et al.* [2018]).

While other machine learning approaches use a series of logical blocks to execute the learning process, ANNs operate on a parallel network of nodes. These nodes are responsible for locally processing the abstracted information and, later, through a training algorithm, their response outputs are optimized to obtain the best resolution of the practical problem (Komijani *et al.* [2017]).

It should be noted that no neural architecture is equally applicable to all types of problems found in the literature. Thus, various ANNs have been developed over time, some of which are convolutional (CNN) and recurrent models (such as the LSTM network).

However, most neural architectures are composed of a series of processing units (called artificial neurons) distributed in groups (named layers) in the form of a chain. Thus, as each neural layer is a combined function of the predecessor layer, an ANN's assertiveness also depends on the number of layers used (Komijani *et al.* [2017], Rezende [2021]).

When an ANN is composed of three or more processing layers, it is characterized as a deep neural network, and its application comprises the so-called deep learning (DL) (Chen, Li and Sanchez [2015], Yu *et al.* [2015], Blanco *et al.* [2019]).

In the deep learning approach, the levels of representation of the data are used to follow a hierarchical flow; that is, the high-level characteristics are obtained exclusively through the composition of lower levels, which specialize in a specific type of information (Gulgec, Takáč and Pakzad [2017]).

For this, deep learning models usually require large volumes of data and a high amount of hidden layers for abstraction. As a result, a high computational cost is also required, with data processing often being performed on the computer's graphics unit (Yu *et al.* [2015], Gulgec, Takáč and Pakzad [2019]).

It should be noted that the topic of deep learning is an aspect of machine learning (Figure 1), which has gained significant importance in recent years, mainly due to its ability to solve problems that were not treatable until then; one of them being structural integrity monitoring (Verstraete *et al.* [2017]). Figure 1 presents the arrangement and relationships between the different tasks and areas involved in the context of ML and DL.

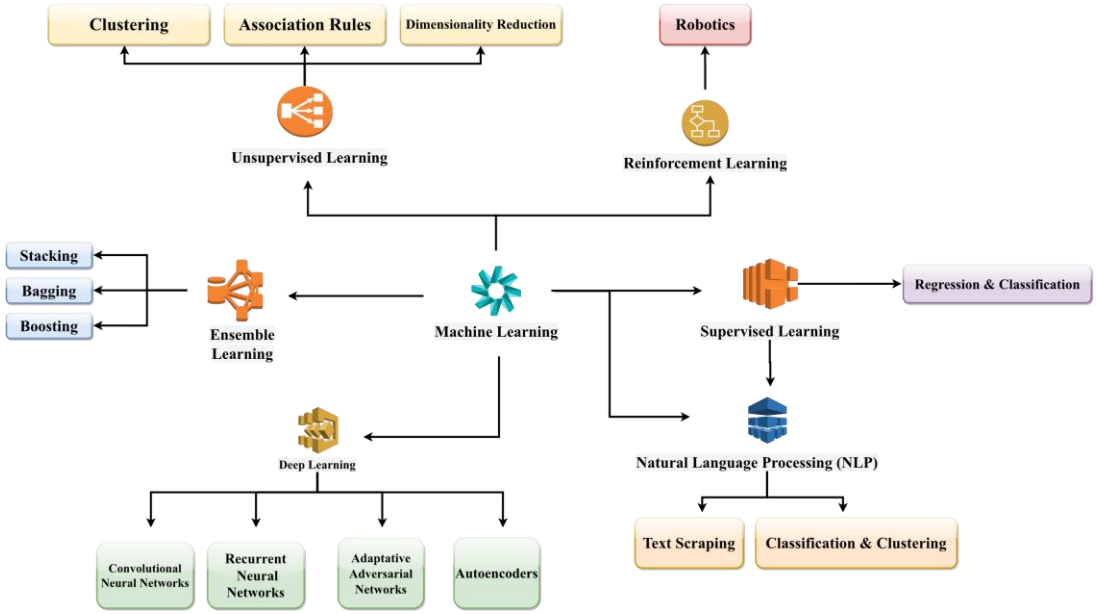


Figure 1: Association between the main tasks in machine and deep learning. Source: Adapted from Trekhleb and Xie [2022].

As shown in Figure 1, several machine learning methods and models are available in the literature that is usable for the most varied types of engineering problems, one of which is identifying the presence of damage in mechanical structures (Verstraete *et al.* [2017]).

Furthermore, DL models represent only a tiny portion of the entire set of models and techniques available. However, existing models have been widely used in several studies, such as natural language processing, computer vision, and automatic sample generation (Portsev and Makarenko [2018], Abdoli, Cardinal and Koerich [2019], Rezende [2021]).

From a general point of view, the objective of using a more significant number of layers in neural models is to increase the abstraction capacity of the data, which are evaluated from different perspectives, to condense better and associate the relationships between the inputs and outputs of the model of the system. Then, with the formulation of the neural model, it is then possible to infer, for example, the future behavior of the structures under study. This procedure is carried out from data not yet verified by the model (Rezende *et al.* [2020]).

2.1. Convolutional Neural Architecture

Convolutional Neural Networks (CNNs) are examples of deep learning models whose procedural formulation is inspired by some areas of the human visual cortex. These regions, called local receptive fields, are responsible for activating different neurons to perform the resource abstraction process and, through their overlap and specific selection, identify patterns of information present in the data (Chen, Li and Sanchez [2015], Albawi, Mohammed and Al-Zawi [2017], Gulgec, Takáč and Pakzad [2019]).

As a rule, CNNs architectures can be composed in different ways, associating different types of neural layers. In which at least one of them, the mathematical operation of convolution is applied and which, in turn, is responsible for the network abstraction capacity (Gulgec, Takáč and Pakzad [2017], Indolia *et al.* [2018], Agarwal *et al.* [2021]).

The convolution operation adopted by a convolutional layer is made from a sliding data window, commonly called a kernel, which is constituted by a mesh of synaptic weights and passes through all the inputs of the evaluated layer. This slippage and abstraction procedure is done to highlight the main local features of each dataset (Ghazvineh *et al.* [2021]).

For this, CNNs assume that in the topology of the experimental data, the values of closer indices are much more correlated than the values of distant indices. This ideal, in turn, can be considered the main factor that justifies the intense application of the convolution technique in images, audio, and other types of vector sets (Chen, Li and Sanchez [2015]).

Furthermore, it should be noted that each convolutional layer of a CNN may contain several abstraction kernels to improve the patterns of available information. The results achieved by each abstraction filter, on the other hand, generate a map of resulting features, which is used as input for the following constituent layers of the network (Chen, Li and Sanchez [2015]).

Figure 2 outlines a general representation of the CNN topology, considering a one-dimensional impedance signal (impedance amplitude only) as input to the model.

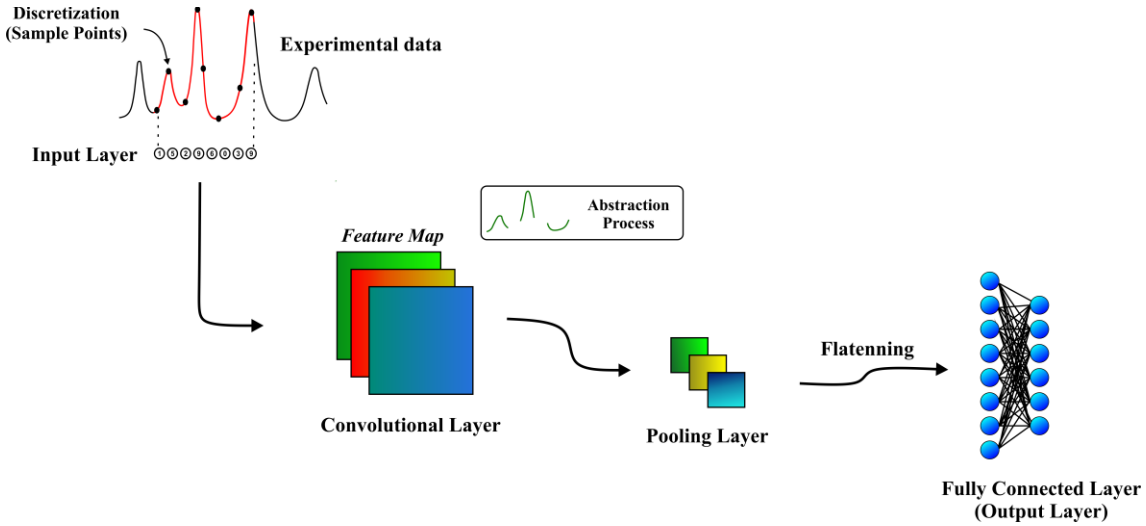


Figura 2: Topologia geral de uma rede convolucional unidimensional.
Source: Adapted from Rezende *et al.* [2020].

As can be seen from the image, we arrive at a feature map of the layer's input values due to applying the convolutional layer. This map, in turn, represents a template of the main characteristics observed in the input data about a given abstraction filter. However, due to the convolution process itself, some concepts intrinsic to this type of process are also present and significantly influence the result of the network; they are the value of stride, padding, and the sharing of free parameters (Chen, Li and Sanchez [2015]).

The stride defines the sliding relationship between the abstraction kernel and the input data vector. Therefore, this parameter is directly linked to the final sensitivity of the feature map. On the other hand, the padding ratio represents the number of zeros added at each end of the input vector to guarantee the mathematical balance of the process. Therefore, these two parameters, i.e., stride and padding, are significantly crucial for the proper functioning of the network.

At the same time, the sharing of free parameters occurs through the transmission of the values of the synaptic weights to other regions of the feature map. This transmission results from the linear displacement between the abstraction cores and the evaluated dataset. Such sharing produces a preliminary reduction in the parameters to be adjusted by the network, which also becomes of particular interest during the model training phase (Rezende [2021]).

Also, since CNNs are generally used to process high-dimensional data, it makes sense to use subsampling layers between their convolution layers. These pooling layers (Figure 2), in turn, are intended to synthesize the information previously abstracted by the convolutional layers, thus employing another reduction in the observed dimensionality without, however, the considerable loss of information necessary for the problem (Rezende *et al*. [2020]).

In the literature, several types of functions can be used in the subsampling step of the pooling layer. Among these functions, we can mention the value choice: maximum, average, and Euclidean norm, among others. However, they aim to represent the analysis region in a single numerical value, transmitted to the next layer through a reduced feature map (Chen, Li and Sanchez [2015], Ghazvineh *et al.* [2021]).

After the synthesis performed by the last pooling layer of the convolutional model, the output values of this layer are fully connected to a new feedforward model, which is responsible for classifying and defining the possible general outputs of the network (Agarwal *et al.* [2021]). A feedforward model is given with a set of neural layers, whose neurons in each layer are fully connected to neurons in its successor layers. Thus, the processing flow follows in a single direction, allowing the identification of patterns observed instantly through the current dataset.

Based on the configuration of practical problems, the output of a neural network must be designed according to the type of problem to be addressed, which can then be given through deterministic data (discrete values) or the degree of belonging to a group (continuous values). Therefore, the final dimensionality of the fully connected layers of a CNN must also be adjusted for this purpose.

It should also be noted that to make the connection between the last pooling layer and the first fully connected layer, a flattening sublayer is used between these two layers to maintain the mathematical and computational integrity of the problem. Such an event stems from the fact that the dimensionality achieved by the feature map resulting from the pooling layer is given as a two-dimensional matrix. However, the input of a feedforward model must be exclusively a one-dimensional vector. Consequently, an adjustment must be made to allow the algebraic operations performed by the network to be completed (Agarwal *et al.* [2021], Ghazvineh *et al.* [2021]).

Once the neural architecture to be used is formulated, its training process begins, which in this case takes place as a supervised learning process. Thus, throughout the synaptic weights' adjustment process, patterns of ideal responses are presented together with the input data groups of the network, and, through a whole process of backpropagation, the network parameters are reevaluated to reduce the errors returned (Gulgec, Takáč and Pakzad [2017]).

In this sense, it is noteworthy that, as CNN networks are Feedforward, they have only one property of immediate reasoning: information is acquired over time by modifying synaptic weights. As a result, all information processed by each layer is associated only with current input values rather than previously processed data by the network.

2.2. Long Short-Term Memory Architecture

In short, to characterize a dynamic system, the analysis of its behavior under current conditions is not enough since its previous state usually influences its current state. In this context, feedforward neural architectures do not always become applicable, as they only have a direct relationship between the current input of the network and its respective output (Bispo [2018]).

This condition of unidirectional input-output transmission performed by feedforward networks makes it impossible to process new information based on resources already processed by ANN. Therefore, such networks can be characterized only as immediate reasoning networks. Short-term memory resources are not present, and the learning process is achieved only by carrying out the entire process of training the synaptic weights (Rezende [2021]).

Thus, based on this principle, it is necessary to add to the ANNs some changes in their topology, which allow them to store a temporal relation of their previous states for their current evaluation. Among the various possible modifications, two stand out: delays in entering the network and the recurrence process.

Delays in the entry of an ANN consist of time-dependent samples, which are composited together with their previous states. Thus, with the application of the delay technique, only the set of inputs is changed, and the ANN architecture remains of the feedforward type, which facilitates its implementation process. However, this adaptation alternative is only applicable in medium-complexity systems. For high-complexity systems, internal structures of state changes are required, which is achieved by applying recurrence between the neurons of an ANN.

Therefore, the recurrence property in an ANN is given by the presence of cycles between its processing units; that is, the output of a neuron of the nth layer is used as input for a neuron of a lower-level layer or the layer itself, assigning feedback to the architecture that will serve as a short-term memory engine (Le *et al.* [2019]).

Figure 3 exemplifies the recurrence property in an artificial neural network, this effect being used both about different layers and the feedback of the same layer.

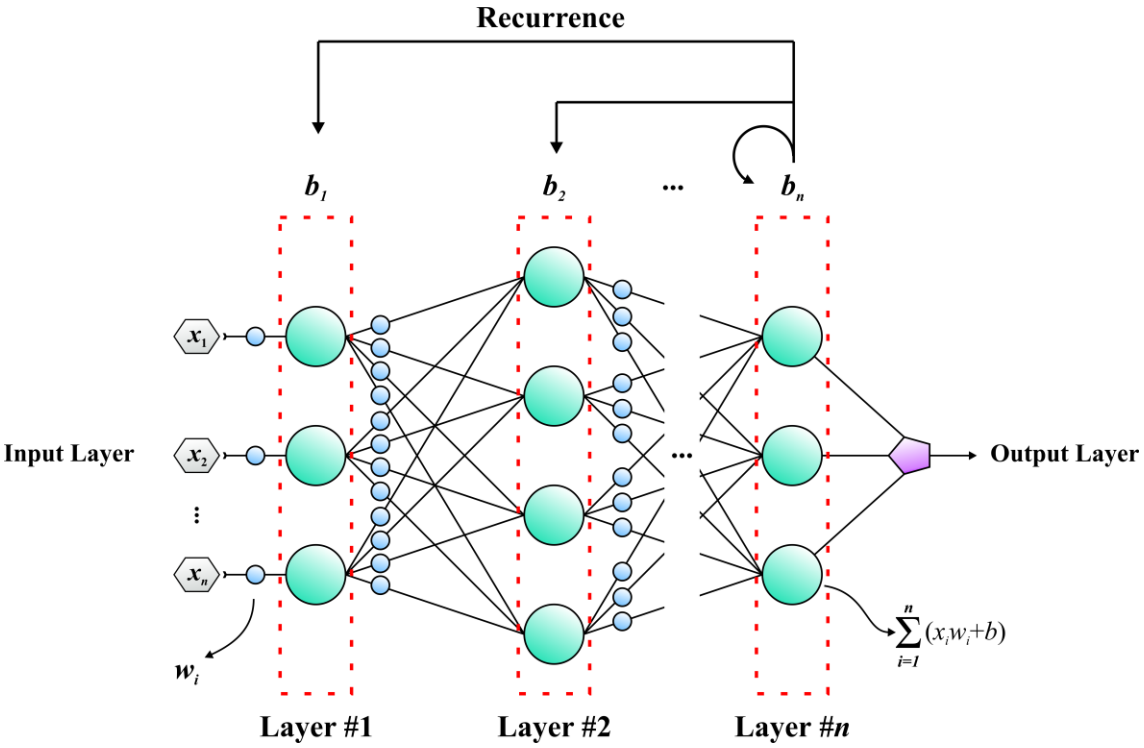


Figure 3: Exemplification of a recurrent neural network.

The practical effect of recurrence in ANNs is to streamline the neural models over time, causing the observed features to flow in both propagation directions and, at the same time, being able to abstract new information through parameters already measured. When a neural network is built with feedback neurons, it is called a Recurrent Neural Network (RNN) and is characterized as a feedback topology (Le *et al.* [2019], Rezende [2021]).

This short-term memory resource obtained by RNNs makes this type of neural architecture instrumental for many kinds of real applications. However, in practical terms, such models still have some disadvantages when evaluated on large datasets (with unlimited time windows), as their training process can become significantly complex and slow.

Thus, in the quest to improve the temporal processing limit reached by RNNs, about the amount of information kept in their short-term memory, two problems with the gradient descent calculation emerged in deep learning studies. The first, called Exploding Gradient Descendent, refers to an instability imposed on the synaptic weights of the network throughout its training process. This instability, caused by the accumulation of errors evaluated in large chains of layers, can cause some of the values of the synaptic weights to cancel out for the initial layers. In contrast, others become considerably large, and thus the prediction made by the model is affected.

The second problem, called Vanishing Gradient Descendent, is the inability of assimilation by the network for significant temporal dependencies; that is, the gradient values used in the error correction cancel out for the first-time parcels, and the training process ends up not working as expected (Hochreiter and Schmidhuber [1997], Zhao *et al.* [2016], Sherstinsky [2020]).

To effectively solve these two problems that occur with gradient descent when a large volume of sequential data is evaluated, computer scientists Sepp Hochreiter and Jürgen Schmidhuber introduced, in 1997, a new model of recurrent neural network called Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber [1997], Rezende [2021]).

The LSTM architecture is an example of a feedback network that can retain information for a more significant amount of network inputs, evaluated during the neural training process. The error flow calculated during the same processing is kept constant by including special units in its characteristic neurons. These units are popularly known as gates.

The gates of an LSTM allow the neural network to assertively adjust its synaptic weights while truncating the gradient when information is no longer needed. Such a procedure symbolizes a form of forgetfulness on LSTM, which avoids canceling certain parts of its training process.

Like all RNN architectures, LSTM networks can memorize some information about their previous states over time. However, it should be noted that the LSTM architecture manages to learn and control the time this information remains during the training process, working to create a much more efficient long-term memory mechanism than a normal RNN. This attribute is obtained through one of the gates of the LSTM neurons called forget gate (Le *et al.* [2019]).

In general, neurons (also known as memory cells or blocks) of an LSTM network are composed of 3 gates: the forget gate, the input gate, and the output gate. They all have an activation function to control the information that flows in each direction of the cell, thus enabling its schematization, according to Figure 4.

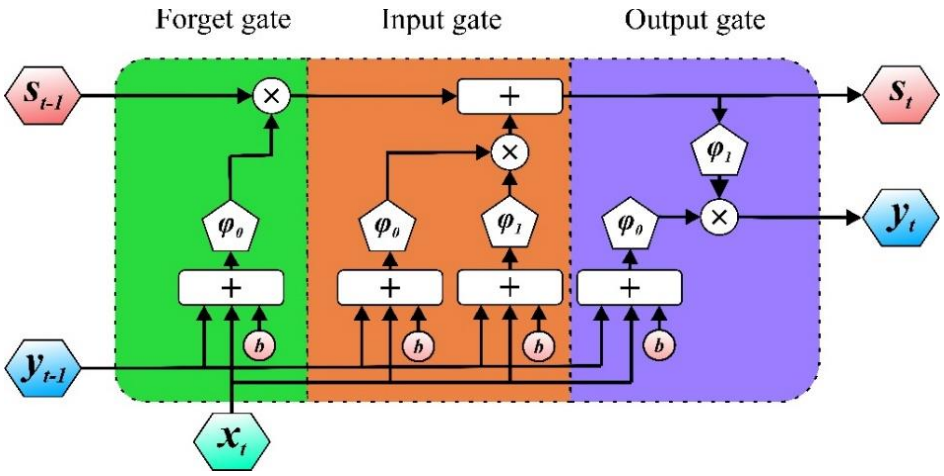


Figure 4: Representation of a memory cell of an LSTM network.
Source: Adapted from Rezende [2021].

As mentioned, the forget gate controls which information will be kept by the LSTM cell. Consequently, also responsible for eliminating unnecessary information from previous states, which contributes to the network learning process. For this purpose, three inputs are presented to the unit, which are: a current input vector (x_t), a previously hidden state vector (s_{t-1}), and an output vector of the previous layer cell (y_{t-1}). Each vector is multiplied by its respective weight matrix and added to the bias (which is responsible for compensating and dimensionalizing the input values in the activation function of this gate) (Yan [2016], Rezende [2021]).

After the usual propagation calculation inside the forget gate, the result is passed through a binary activation function (φ_0), where output 0 represents a forgetting process of the evaluated data, and output 1 means the correctness of the information for future use.

The input gate, on the other hand, is responsible for updating the LSTM cell at the current time, this procedure being based on the cell input values at time t and its previous hidden state s_{t-1} (measured by the forget gate). In this way, the flow of information within this unit is controlled according to equation 1,

$$s_t = \varphi_0(W_0 \cdot [y_{t-1}, x_t] + b) \cdot \left(s_{t-1} + \varphi_1(W_1 \cdot (y_{t-1}, x_t) + b)\right) \quad (1)$$

where φ_0 and φ_1 are generally nonlinear functions and W_i are weight matrices evaluated in each part of the cell structure. In the literature, the sigmoid and TanH functions are commonly used to represent the components φ_0 and φ_1, respectively.

The output (y_t) of the LSTM cell is obtained through the output gate, which uses a sigmoidal function to determine which of the state values will be remembered for the measurement of its final response. These values are then multiplied to a readjustment (exercised by applying a TanH function on the hidden state of cell s_t), to generate the output for the next cell (Yan [2016], Rezende [2021]).

A long-term memory neural network comprises several processing cells layered to form the LSTM architecture. This neural network model has been widely applied in recent years, mainly in treating natural language and time series analysis, thus justifying its evaluation in the present chapter.

3. Experimental Case Studies

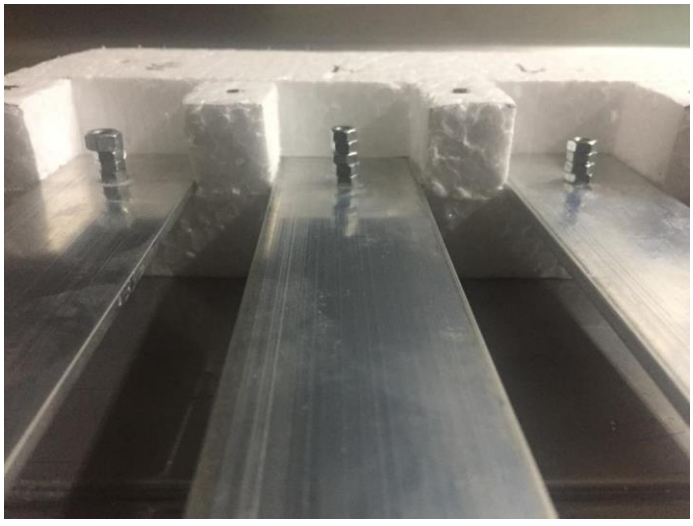
3.1. Application of CNN for Temperature Variation Problems

To facilitate understanding the implementation of a CNN in python language and already introducing it in the context of monitoring the structural integrity, the classification of the electromechanical impedance signatures of three geometrically identical aluminum beams (500x38x3.2*mm*).

Therefore, each structure was monitored under two different integrity conditions (with and without damage) and at three acquisition temperatures (0ºC, 10ºC, and 20ºC). Thus, the experimental conditioning adopted here aims to verify the ability to detect damage in structures subject to temperature variation using the CNN architecture.

The acquisition of impedance signatures was performed using a PZT transducer (20*mm* in diameter by 1*mm* in thickness) coupled 100*mm* away from one of the ends of each analyzed structure. The damage simulation was performed by adding mass to the three structural systems (three nuts ranging from 0.6-2.2*g* glued 380*mm* away from the PZT adhesive). Figures 5.a and 5.b show the three aluminum beams used and the imposed damage conditions.

a) Structural Systems (Aluminum Beams + PZTs Insert).



b) Mass addition damage conditions.

Figure 5: Structures used as the object of study.
Source: From Rezende *et al.* [2020].

They were submitted to a boundary condition (single support) to analyze the structures correctly. In this sense, polystyrene foams were used to minimize the influence of noise during the acquisition phase.

Environmental conditions can cause minor changes in the electromechanical impedance signatures during the acquisition process. Thus, in the literature, efforts have been made to minimize possible errors in structural prognosis (Afshari [2012], Rabelo *et al.* [2017a], Rabelo *et al.* [2017b] and Tsuruta *et al*. [2017]).

In this investigation, to delimit and control the three different temperature levels, a climatic chamber of the Platinous EPL-4H series was used, which is available at the LMest laboratory of the Mechanical Engineering course at the Federal University of Uberlandia. Figure 6 illustrates the climate chamber model used in this case study.

Figure 6: Platinous EPL-4H Climatic Chamber.
Source: From Barella [2021].

The Platinous EPL-4H series climate chambers are based on BTHC (Balanced Temperature and Humidity Control), employing a thermodynamic balance to control temperature parameters. The operating volume of the model used is approximately 900L (100x100x90cm) and can handle the temperature from a range of -35ºC to 180ºC.

For the construction, training, and validation of the CNN model, 20 electromechanical impedance signatures were collected for the baseline state (pristine condition of the structure) and 80 signatures for the damaged state, thus totaling 900 samples for evaluation (300 for each of the beams).

The frequency range used in acquiring impedance signatures was 68-77*kHz* with a step of 4.5*Hz* and 2000 sample points. It is also worth mentioning that the frequency band used was chosen through a trial-and-error process to subsidize a predominance of the random search optimization method.

The random search optimization method was applied to the set of electromechanical impedance signatures to delimit the best scanning range for identifying the presence of damage among the pre-chosen frequency portion. This procedure was based on the methodology imposed by Bento [2017], which uses the RMSD damage metric as a comparison of the best frequency range to be monitored.

Thus, with the application of the random search optimization method, the frequency range chosen became 71*kHz* to 75*kHz*, with a total of 888 sample points. The averages of the impedance signatures can be seen in Figure 7.

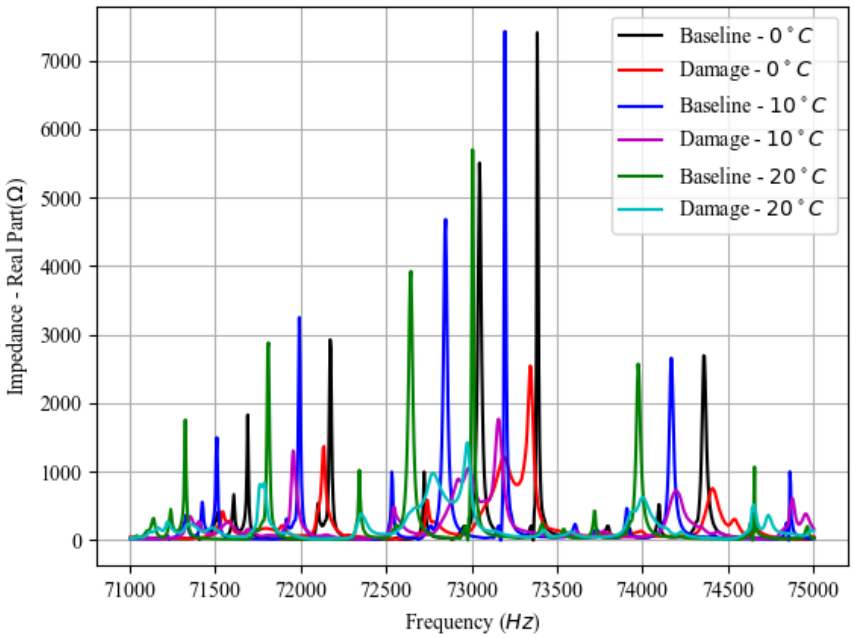


Figure 7: Means of the impedance signatures of each evaluated group.
Source: From Rezende *et al.* [2020].

To define the best CNN architecture for the problem in question, the type and dimensions of the input data used in the network must be identified. Electromechanical impedance signatures are two-dimensional vectors, where we have the electromechanical impedance signals in the frequency domain. Such a phenomenon can be modeled, in this way, by a 2D-CNN using its two parameters. However, if we consider a standardized frequency range for all samples evaluated, another one-dimensional approach can still be performed, using only the impedance values measured in the transducer.

Because the samples considered are measured in the same frequency range (71-75*kHz*), one-dimensional analysis of electromechanical impedance signatures will be employed using a 4-layer 1D-CNN architecture. Thus, the topology used will comprise a convolutional layer, a pooling layer (MaxPooling type), and two fully connected layers, as shown in Figure 8.

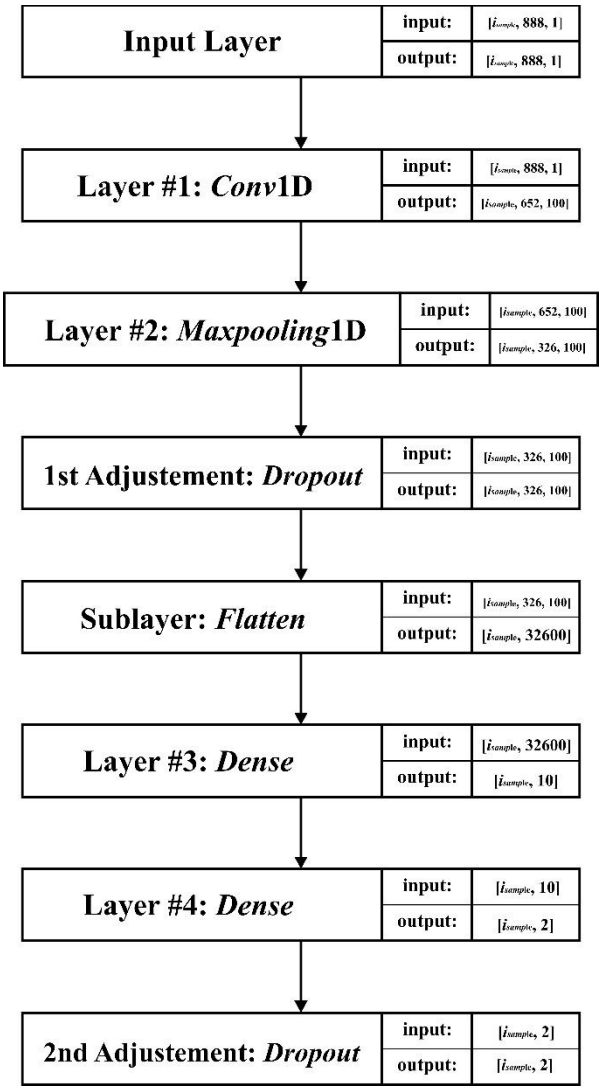


Figure 8: Convolutional neural architecture implemented in the experiment. Source: From Rezende *et al.* [2020].

It is worth noting in Figure 8 that the steps called "1st Adjustment", "Sublayer" and "2nd Adjustment" were not included in the count of the layers used since these, by their purpose, apply only one preset to the set of synaptic weights. Furthermore, the dimension i_{sample} represents the batch size of samples evaluated during iteration i of the training phase (although each of these samples is processed separately to adjust the synaptic weights of the network) (Rezende *et al.* [2020]).

To initialize the construction of the convolutional neural network model in python, one must initially import the TensorFlow library into the system's memory and stabilize and stack the first layer (input) of the CNN model. This process can be performed by executing the commands below:

```
>> import tensorflow as tf
>> input = tf. keras.layers.Input(shape=(888,1))
```

All sampling points of the electromechanical impedance signatures were used in this contribution. In this way, the developed inference system adds more degrees of freedom, which provides greater reliability to the classification method.

To introduce the full extent of the impedance signatures in the CNN model, an input layer with 888 neurons was implemented. Thus, each sampling point has its respective input neuron, enabling the convolution between the electromechanical impedance signatures and the abstraction filters.

As a next step in constructing the input layer, the other convolutional and pooling layers must be alternately stacked to the CNN model to elaborate the network architecture to be developed.

Altogether, there are three types of convolutional layers previously implemented in the Keras package: *Conv1D*, *Conv2D*, and *Conv3D*. However, all kinds of convolutional layers perform the same mathematical functions and processes, differing only in the number of dimensions in which the data is convoluted (Rezende [2021]).

The only one-dimensional analysis will be applied to the electromechanical impedance signatures. Therefore, for the implementation of the network, a 1D convolutional layer with 100 abstraction filters was used, where each filter had a dimension of 237 randomly chosen synaptic weights.

The implementation of the convolutional layer developed in this section followed the TensorFlow formulation in the python IDE:

```
>> net_cnn = tf.keras.layers.Conv1D(filters=100, kernel_size=237,
                                    strides=1, padding='valid',
                                    use_bias=True,
                                    activation='relu')(input)
```

where the parameters given to the function are, respectively, the number of filters used for convolution, the length of the abstraction kernel, the linear displacement of the kernel, the type of padding, the use of bias in the convolutional layer, and the activation function used.

The displacement between the abstraction kernels and the input data was assigned as step 1 to evaluate all possible sample points for the convolution process. However, the padding action (valid) was not attributed since the extreme data of the signatures used were already null.

After having performed all the convolutions of the input signals by the convolutional layer, an activation function of the *ReLu* type was applied to its feature maps. The purpose of using this function is to normalize the results of each feature map, thus allowing only the propagation of positive values of the output data of the predecessor layer.

It is worth mentioning that all parameters used in modeling the convolutional layer were chosen to analyze a better portion of the signals used, which are specifically dependent on each case under investigation.

Still, due to the application of the parameters mentioned above, it can be seen in Figure 8 a reduction of 26% in the amount of data to be evaluated by the pooling layer.

This paradigm contributes to its computational cost and helps the classification method's abstraction and data separability.

On the other hand, coupling a section layer to a CNN model depends mainly on the type of function imposed by this same layer, thus varying between the maximum or medium value of each subregion of the characteristics map.

In this work, the pooling layer adopted in the CNN model used only order two filters to abstract the impedance characteristics, keeping only the highest value reached for each of the analyzed areas of the feature map. In this way, only 50% of the input data from the pooling layer will be propagated for consecutive layers.

The python function used for coupling the pooling layer in the CNN model is given as follows:

```
>> net_cnn = tf.keras.layers.MaxPool1D(pool_size=2)(net_cnn)
```

where the only parameter provided to the function is the inspection window dimensionality, which corresponds to the size of the feature map subregion to be evaluated.

Two other adjustment sublayers were implemented to couple the fully interconnected layers to the CNN model, one for dropout and the other for flattening. Such an application aims to computationally adjust the response data of the pooling layer so that they are used as input to the feedforward model.

Due to the significant dimensionality of the output data from the pooling layer, two fully connected layers were added to the classification model. The first dense layer was implemented with ten neurons to allow a reduction in the adjustable parameters of the last layer, which in turn is responsible for the network response output.

The CNN architecture implemented in this contribution aims to catalog the impedance signatures in two states, with and without damage. In this way, the response layer of the model has only two neurons, whose outputs represent the probability of belonging, respectively, to the classes with damage (activation of the second neuron) and without damage (activation of the first neuron).

For the output layer to respond to the probability of each sample belonging to a specific group, the *Softmax* logistic function was used in the fully connected layers. This function performs input data distribution, classifying them according to their similarity with the pre-delimited target classes (Chollet [2017]). Thus, the configuration of the parameters used to implement the fully connected layers is given as follows:

```
>> net_cnn = tf.keras.layers.Dropout(0.2)(net_cnn)
>> net_cnn = tf.keras.layers.Flatten()(net_cnn)
>> net_cnn = tf.keras.layers.Dense(units=10, activation='softmax') (net_cnn)
>> net_cnn = tf.keras.layers.Dense(units=2, activation='softmax') (net_cnn)
```

After the neural topology is developed, the storage and construction of the CNN model with the optimization parameters and adjustment of the synaptic weights must be carried out. Such parameters define how the training algorithm will access and regulate the accessible attributes of the implemented model and delimit the loss function used to identify the error absorbed by the network.

To build a CNN model using the TensorFlow library, type the following commands in the python IDE:

```
>> model = tf.keras.models.Model(input, net_cnn)
>> optimizer = tf.keras.optimizers.RMSprop(learning_rate=0.001, rho=0.8)
>> loss_function = tf.keras.losses.BinaryCrossentropy()
>> model.compile(optimizer= optimizer, loss=loss_function, metrics=['accuracy'])
```

The "Model" function performs the sequencing and construction of the neural layers that make up the network. This process is based on pre-defined tensors. The "Compile" function, on the other hand, configures the neural architecture training and learning processes, delimiting the loss function and the optimization method according to the parameters provided to it.

In this experiment, the *Binary Crossentropy* loss function and the RMSprop (*Probabilistic Root Mean Square*) optimization method are used since they exhibit good efficiency in classification problems. Still, in this section, it is worth noting that the learning rate (η) and the gradient decay factor (ρ) were defined by carrying out previous experiments since they directly influence the performance of the neural training process (Rezende [2021]).

After executing the entire process of building and configuring the CNN model, the network training and learning stage starts. The model is adjusted according to the training samples and target values. To train a CNN in python, we used the fit function, as shown below:

```
>> history = CNN.fit(training_samples, target_training, batch_size=2, epochs=40)
```

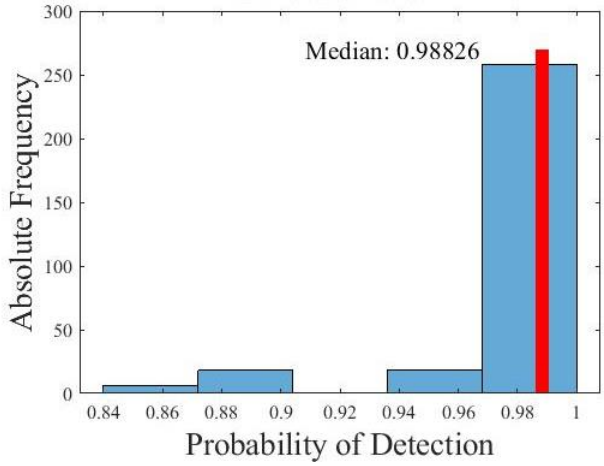
The total epochs for training and the number of samples needed to update the synaptic weights (batch size) are also given to this method. The fit history of the synaptic weights is then stored in a data structure for further analysis, such as verifying the learning curve and measuring the accuracy achieved.

It is noteworthy that a CNN model was built and trained for each beam structure individually to enable the prediction of structural damage. Thus, from the 300 impedance signatures for each model, 30 random samples were removed for testing, and the remaining 270 samples were considered to train and build the previously described model.

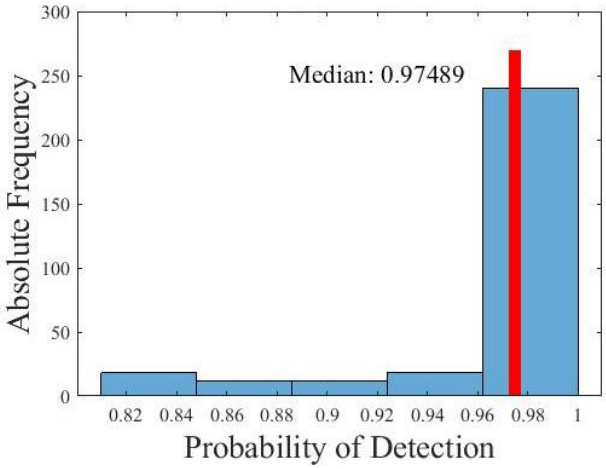
As mentioned, during the training process, the *Binary Crossentropy* loss function was used to verify the response accuracy of the networks. All models achieved a loss function of about 0.221, representing a slight difference between the result and the target groups related to the complexity of the problem.

Subsequently, the RMSprop optimization algorithm calculated the adjusted weights of each network. At the end of this stage, the accuracies were approximately 85.74%, 89.40%, and 95% for beam structures #1, #2, and #3, respectively.

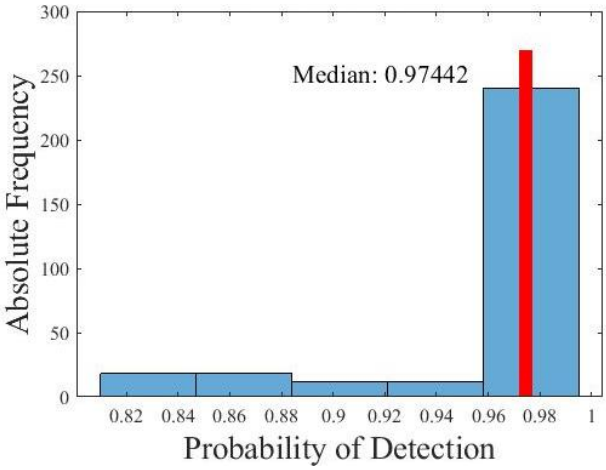
After adjusting the weights, the trained networks were evaluated using the test samples as input to the models. Thus, each model had ten runs for all 30 samples in the batch. As a result, all three models obtained different degrees of probability of damage detection. Figures 9.a, 9.b and 9.c show a histogram of all tests performed.



a) Test results for structure #1.



b) Test results for structure #2.



c) Test results for structure #3.

Figure 9: Histogram with the test results imposed on each model.
Source: From Rezende *et al.* [2020].

According to Figure 9.a, the probability of detection of damage by the first CNN model (structure #1) varies from 84% to 100%. This graph includes all assessed

integrity conditions (baseline and damage) in the test. Since the distribution of results does not follow a normal distribution, its median (which is a better parameter for comparing groups) is evaluated, reaching 98.83%.

Figure 9.b illustrates the histogram for the model of structure #2, and similar results were obtained for the second model, with a probability of detection of damage from 81% to 100%, with a median of 97.49%. Finally, the last histogram presents the probability of damage detection for the third model (structure #3), ranging from 81% to 100%, with a median of 97.44%.

In this initial evaluation, all the developed CNNs models reached a probability of damage detection greater than 97%, meaning a good damage classification capacity for the three structures used. Thus, it is worth mentioning that, although the model used in this section is simple (composed of only one convolutional layer and pooling), the results achieved are favorable to the application of this deep learning technique in the electromechanical impedance SHM method.

Also, since environment temperature is very relevant for the impedance-based SHM technique, changing the amplitudes of its electromechanical impedance signatures, this specific result demonstrates the ability of models to separate damage and primitive signatures, regardless of temperature, being very relevant to the proposed methodology.

3.2. Application of LSTM for Corrosion Problems

In contrast, to execute the implementation process of an LSTM network in Python language, two geometrically identical steel beams (300x50x3.2*mm*) were considered under controlled corrosive action. This experimental condition was formulated to verify alterations in the impedance signatures resulting from the corrosion process of the structures, analyzing the progression of severity in them.

Thus, the experimental design adopted here aims to verify the ability to predict the magnitude of structural failures in steel beams subjected to hydrochloric acid (HCL) corrosion using the LSTM architecture.

The acquisition of impedance signatures was performed using a PZT transducer (30*mm* in diameter by 2*mm* in thickness) coupled 50*mm* away from one of the ends of each analyzed structure.

To properly evaluate the two steel beams under similar corrosion conditions, all measurements took place with the specimens in the bi-supported form, with contoured regions (cradles) being delimited for the subsequent application of acid in each of the structures separately. Figure 10 shows the beams used in the experimental procedure adopted in this section.

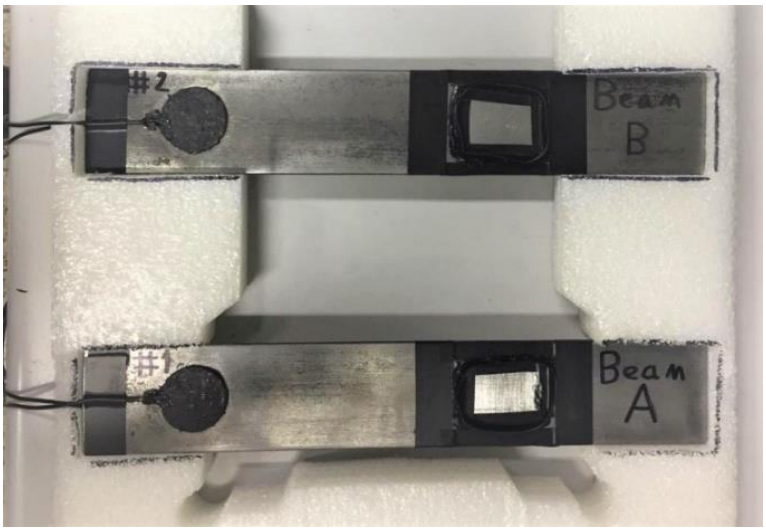


Figure 10: Steel beams used in the HCL corrosion experiment.
Source: From Rezende [2021].

The region (cradle) of HCL application in each structure was defined with a dimension of 32*mm* wide by 44*mm* long, positioned 150*mm* away from its respective PZT adhesive. The concentration of acid used is 36.5%. Its application in each structure was carried out by dripping, which allowed a better optimization of the acquisition process concerning other forms of application present in the literature, such as corrosive mist application.

The order of collection of impedance signatures and application of HCL acid in the monitored structures was defined according to Table 1, in which 30 measurements of impedance signatures were collected for each evaluated severity set, thus totaling 240 samples (120 referring to each of the beams) for the construction of the neural model.

Table 1: Sequencing of collections and integrity groups evaluated in the implementation of the experiment.

Measurement	Application of HCL Acid	Integrity Status Assessed
#1	–	Baseline
	Yes	
#2	–	Damage #1 (24*h* after the 1st Application)
	Yes	
#3	–	Damage #2 (24*h* after the 2nd Application)
#4	–	Damage #3 (48*h* after the 2nd Application)

As can be seen, four levels of integrity of the evaluated structures were considered: an untouched condition of the steel beams (baseline) and three other conditions after partial corrosion of the material by applying HCL acid.

Then, the impedance signatures were monitored in a frequency range of 30-100kHz, with a step of 10.02Hz and 6980 sample points, each calculated with 512 means.

All experiments with the specimens were carried out in an environment enclosed by grids but subject to the external environment, providing variations in temperature and humidity in the monitored beams. Such a formulation was carried out to approximate the experimental tests to the real operating conditions of this type of structure. Thus, it should be noted that the changes in the impedance signatures evaluated may depend both on the characteristics of the progressive corrosion process and the environmental conditions (temperature and noise) of the acquisition phase.

Thus, to minimize the effects of temperature variation on the impedance signatures, a temperature compensation algorithm known as the Effective Frequency Displacement Method by Correlation Analysis was applied, making minor adjustments (horizontal and vertical) in the impedance signatures to maximize the correlation coefficient between the evaluated signal groups (Rabelo *et al.* [2017b]).

After compensating for the effects of temperature variation on the impedance signatures, the RMSD damage metric was implemented to construct the LSTM neural model further. This metric index was used as input to the neural model due to its ability to identify the progression of damage severity better and allow the mutual application of delay and recurrence techniques.

Figures 11.a and 11.b show the impedance signatures and metric index groups before and after temperature compensation.

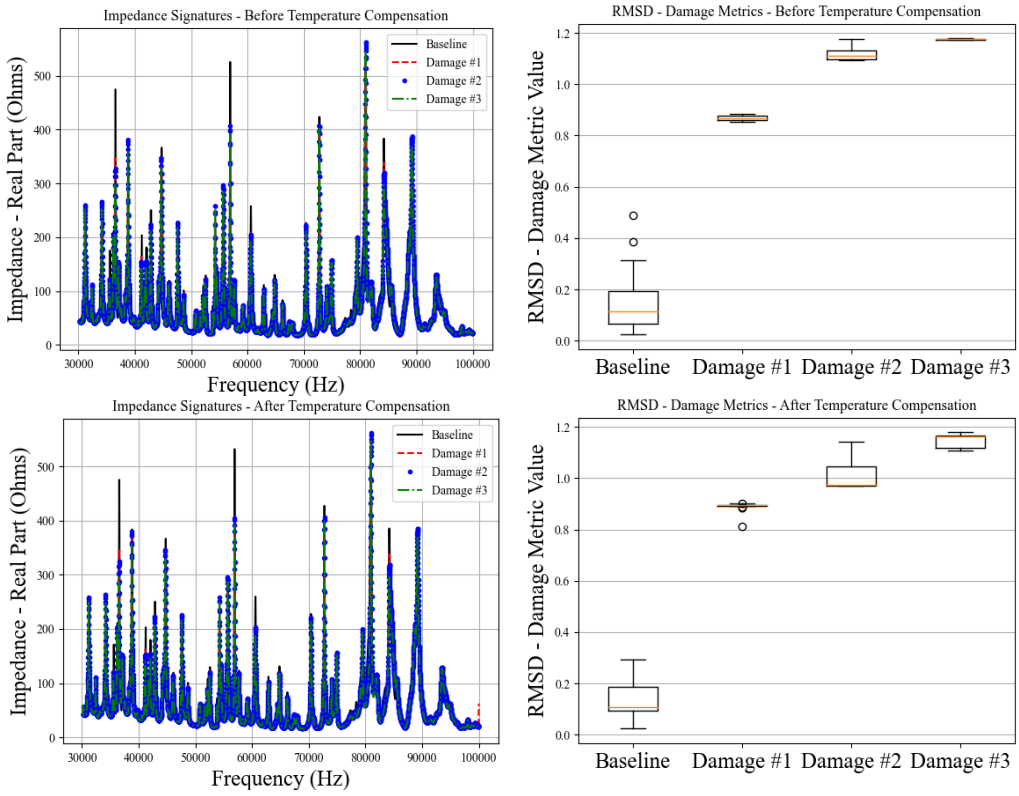
In the present case study, a two-layer LSTM model (Figure 12) will be used to predict the severity of corrosion in the monitored structures, using only the values of the RMSD damage metric as input to the neural model.

The use of this formulation stems from the fact that, in corrosion problems, the level of structural severity obtained by calculating the damage metrics tends to increase over time, allowing the use of such values in the modeling of the phenomenon. Thus, given the above, the neural architecture used will be composed of only one LSTM layer and a fully connected layer expressed according to Figure 12.

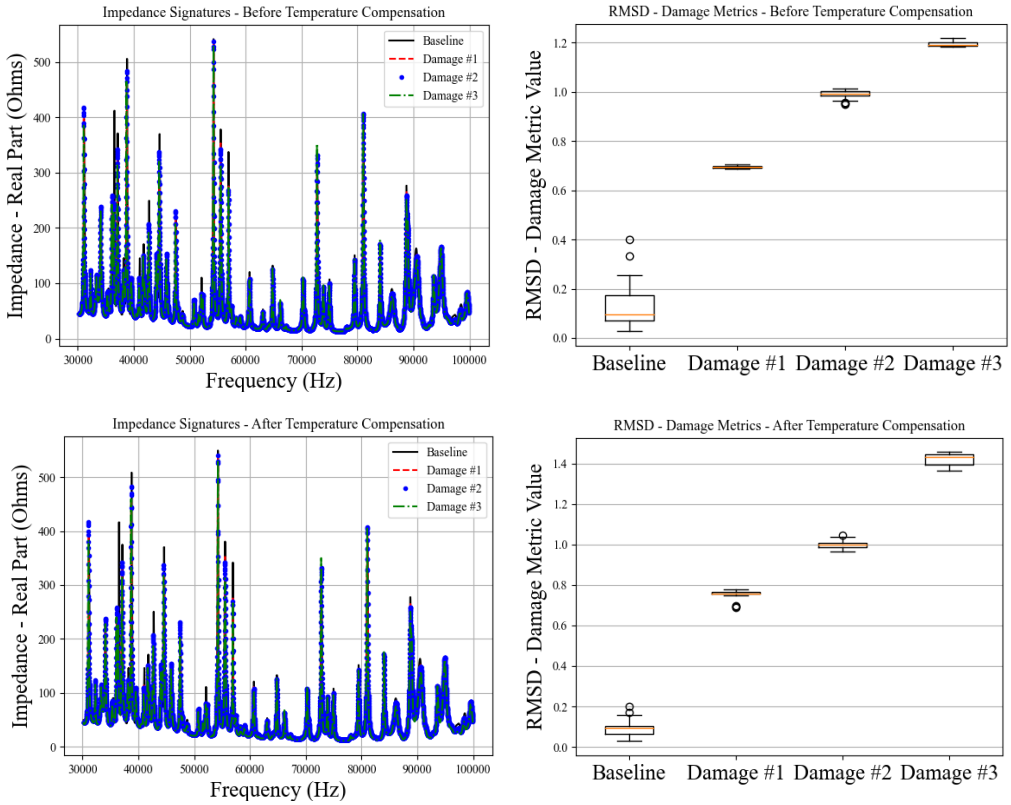
Millstein [2018] highlighted that the modeling of a neural network using the Keras package is performed through the sequencing of tensor layers, in which prototype functions are stacked to build the neural architecture. Thus, the modeling used in this case study followed the same formulation presented in the previous section for the input and fully connected layers, varying only their respective implementation parameters.

As for the recurrence process, an LSTM layer with 120 memory cores was used, in which its activation functions φ_0 (cell activation) and φ_1 (recurrence function) were defined according to the following command:

```
>> net_cnn = tf.keras.layers.LSTM(units=120,
                                  activation='tanh',
                                  recurrent_activation='sigmoid',
                                  use_bias=True)
```



a) Structure #1



b) Structure #2

Figure 11: Averages each severity level's impedance signatures and their respective RMSD metric values.

Source: From Rezende [2021].

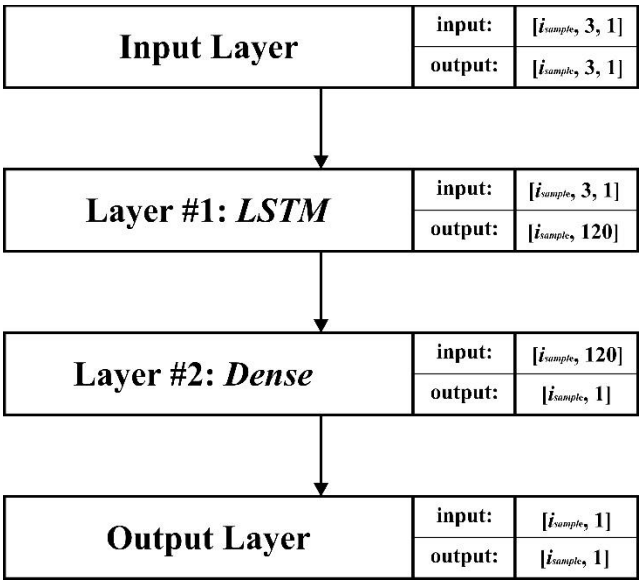


Figure 12: LSTM neural architecture implemented in the experiment.
Source: From Rezende [2021].

As a result of the present case study being evaluated as a regression problem, the RMSE loss function and the optimization method known as Adam was used to configure the neural model, as they present a better efficiency when compared to other methods and procedures available (Gulli and Pal [2017]).

It is also worth noting that an LSTM model was built and trained for each beam structure separately to validate the methodology imposed in this contribution. Thus, of the 120 metric values of each model, 48 sequential samples were removed for testing, and the remaining 72 samples were considered to train and build the previously described neural model.

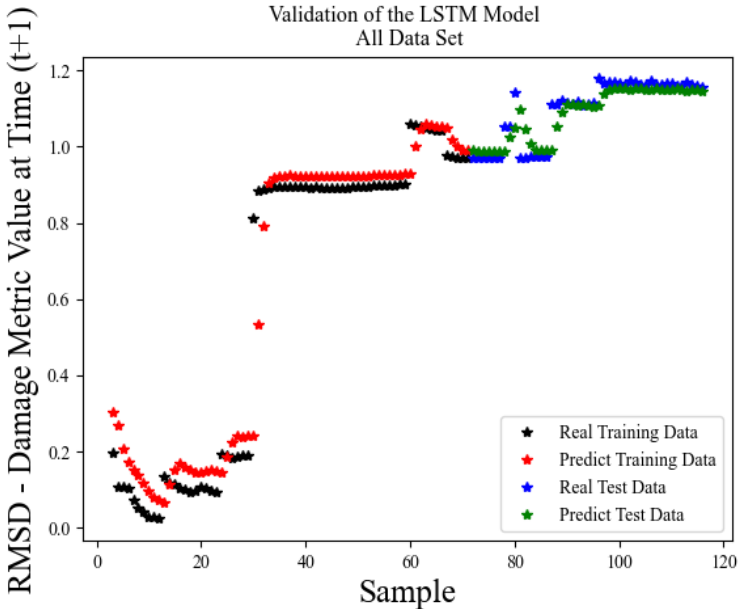
Furthermore, it is worth mentioning that both the delay and recurrence techniques (provided by the formulation of the LSTM model) were applied jointly to the data group evaluated. However, to introduce the delay technique in the present experimental process, it was used for each forecast point and its respective previous units of the series, which caused a reduction in the number of points to be incorporated in the construction of the model. Thus, considering a delay level $j=3$, the final number of points to be used in the model are 45 values for testing and 69 samples for training the network.

As mentioned, during the training process, the RMSE function was used to verify the response accuracy of the networks. All models achieved a loss function of about 0.0109, representing a slight difference between the expected result and the target values related to the complexity of the problem.

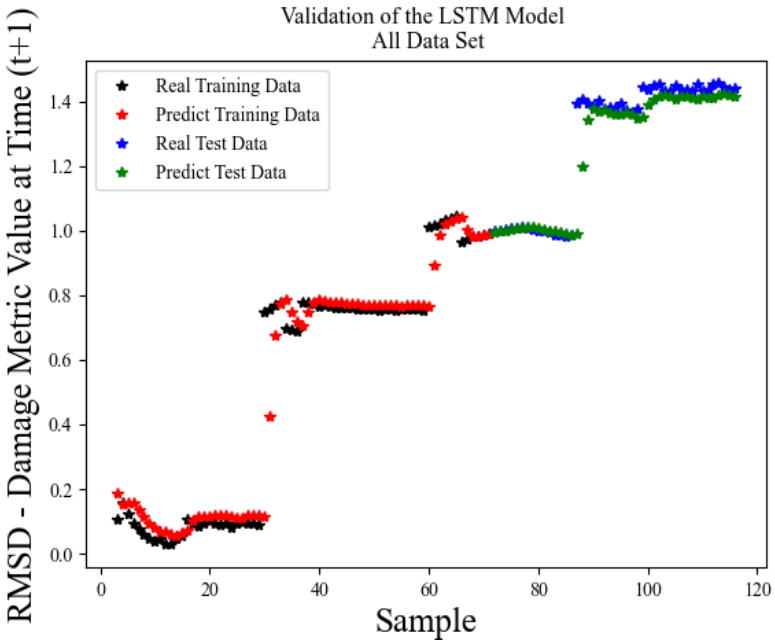
After the error calculation, the optimization algorithm used (*Adam*) adjusted the synaptic weights of each network. At the end of this stage, the accuracy was approximately 90.42% and 90.22% for beam structures #1 and #2.

The trained networks were tested for the efficiency of the test samples after the local parameters of each model had been adjusted. As a result, the accuracy of each model in predicting the structural severity of each surveyed beam varied.

The comparison between the expected results and the actual values of each beam is presented in Figures 13.a and 13.b, and it can be observed that the modeling used in this contribution was able to achieve a high level of precision in terms of structural severity prediction in beams subjected to the corrosive action of hydrochloric acid. For model test data *nº1*, this accuracy was 96.1%, and for model test data *nº2*, it was 92.68%.



a) **Structure #1**



b) Structure #2

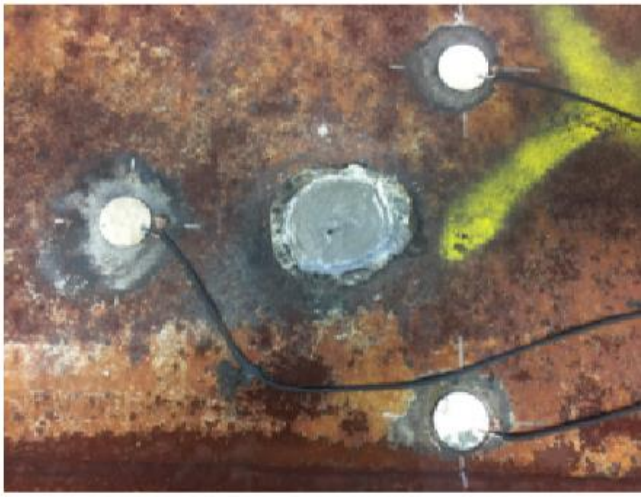
Figure 13: Relationship between the data provided by the LSTM neural models and the pre-delimited target values.
Source: From Rezende [2021].

Consequently, despite the simple LSTM architecture used in this section (containing only one recurring layer), the results were also beneficial for using the technique in conjunction with the SHM method via electromechanical impedance.

2.3. CNN Application for Monitoring Machined Tank Plates

An iron plate-like structure was used as a host system to carry out the third case study of this contribution. At the same time, successive thickness reductions by grinding part of the material were considered the method of artificial damage. Figure 14.a shows the support/structure set used in the present case study, and Figure 14.b shows the application regions of the piezoelectric sensors and the insertion site of the imposed fault.

a) Experimental bench



b) Top view of the structure used

Figure 14: Experimental configuration adopted in the thickness loss study.

As seen in the previous image, three piezoelectric sensors (30mm in diameter) were inserted in the region of interest, equidistant from the area to be ground, restricted by an external diameter of 60mm. However, to simplify the present case study, only PZT-1 impedance signatures will be considered for the next steps of the damage classification process.

Thus, once the instrumental configuration to be adopted throughout the experiment was delimited, a total of 4 conditions of the structural integrity of the plate

under study were evaluated, that is, the initial condition of the plate (integral state) and another three monitoring configurations resulting from material thickness reduction by the plate metal grinding.

In this sense, for each state of structural integrity considered, 30 repetitions of impedance signatures were measured to allow the later use of the CNN architecture in damage classification. In addition, the plate thickness levels in the region of interest were also measured (in a total of 15 repetitions) using ultrasonic tests. Table 2 presents the thickness relationships acquired in each of the repetitions of the ultrasound tests for each state of structural integrity monitored on the plate.

Table 2. Monitored plate thickness measurements in each considered health state.

	Baseline	**Damage #1**	**Damage #2**	**Damage #3**
Measurement #1	5.39	3.65	1.35	1.64
Measurement #2	5.53	3.61	1.97	1.44
Measurement #3	5.53	4.08	2.26	1.38
Measurement #4	5.61	4.30	2.53	2.11
Measurement #5	5.39	4.25	2.06	1.83
Measurement #6	5.65	3.91	2.19	1.71
Measurement #7	5.48	4.04	2.03	1.77
Measurement #8	5.59	3.82	2.07	1.57
Measurement #9	5.35	3.82	2.69	2.19
Measurement #10	5.44	3.65	3.44	2.19
Measurement #11	5.48	3.61	2.50	1.67
Measurement #12	5.50	3.62	2.54	2.07
Measurement #13	5.39	3.52	2.17	1.83
Measurement #14	5.54	3.61	1.57	1.64
Measurement #15	5.57	3.44	1.58	1.64
Average value	5.50	3.80	2.20	1.78
% of thickness	100	69.1	40.0	32.4
Class	1	2	3	4

As can be seen, different levels of material thickness were considered for each state of integrity. These changes, in turn, reflect on electromechanical impedance signatures. Thus, through statistical and mathematical methods, it is possible to infer reliable information about the usability conditions of the structure, making it possible to estimate its useful life and reduce costs and ensure better levels of safety.

A 5-layer 1D-CNN architecture was then implemented in the present case study to verify the CNN topology's sensitivity in identifying and classifying damage due to the loss of machined material. Therefore, the neural model used here was based on a deep neural network composition, being given by an input layer, a convolutional layer, a pooling layer (of the Max Pooling type), and two other fully connected layers (which are to the 3rd hidden layer and the network output layer). Figure 15 presents a schematic of the neural topology used in the tests in this section.

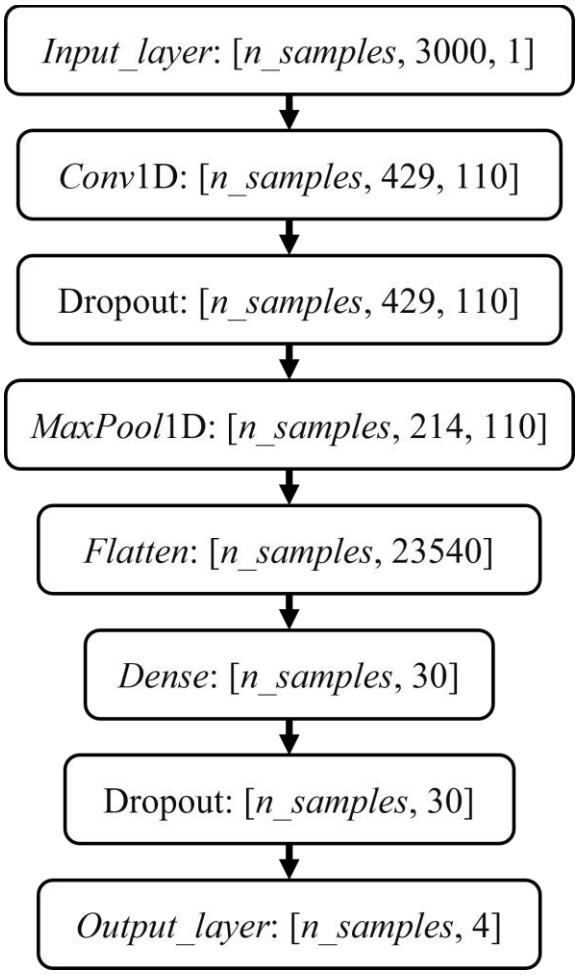


Figure 15: Convolutional neural architecture used in this case study.

It is worth mentioning that, as in the previous problem, the dropout and flattening steps were also not considered in the counting of these neural layers, as they only serve as data manipulation and processing functions during the training phase. Furthermore, it should be noted that all sampling points of the impedance signatures were again considered as input to the CNN architecture, which, in addition to introducing more degrees of freedom to the problem under study (providing more realistic modeling of the observed problem) it also avoids the need to define in advance the unique characteristics of the structure so that its integrity conditions can be identified, making the model responsible for this purpose.

The input layer of the network, in this sense, was formulated with 3000 synaptic neurons, each one representing a specific frequency point of the impedance signature. Then, a *Conv1D* layer was developed with 110 abstraction filters (kernels) and later coupled to the model. Each abstraction filter in this last layer was composed of 200 synaptic weights randomly chosen, and its construction process was like the architecture of the first case study.

For the convolution process, the displacement between the abstraction kernels and the model input data was assigned as step 7 to abstract the main parts of the impedance signal separately. This abstraction reduced 85.7% of the input data,

converting them into a representative vector of 429 sample points for each abstraction filter.

As for the pooling layer of this section, size two kernels were also used, thus keeping only the highest value of each subregion of the data observed in each feature map. Therefore, only 50% of the layer's input data is propagated to consecutive layers, introducing a second reduction of the observed dimensionality (converting each signal into a representative vector of 214 sample points).

After the pooling layer, two other fully connected layers were inserted into the neural network architecture so that the first dense layer was responsible for filtering the data (converting the 23540 input values of the layer into only 30 significant points), and the second layer was responsible for processing them and later defining the network output.

Bearing in mind that the present case study aims to identify and classify the different states of the structural integrity of a machined iron plate system (with loss of material by grinding). In this way, the output of the considered model was attributed through a fully connected layer (of 4 synaptic neurons). Each layer neuron represented the probability that the sample in question belongs to a specific output class (Table 2).

Therefore, to ensure that the output layer responds to the probability that each impedance signature belongs to a group, the softmax logistic activation function was also used in the present case study, with the optimizer model and the loss function equally given the experiment in Section 3.1 of this chapter.

Then, to verify the generalization capacity of the developed neural model (concerning the evaluation and prediction of new impedance data), the set of signals monitored during the instrumentation phase was separated into two subsets of signals, being one for training the network (containing 125 randomly chosen signatures) and the other (including the remaining 55 signatures) for testing and validating the network.

Therefore, considering that the choice of each signature belonging to each set was made at random, the formulation of these sets was characterized by being unbalanced about the evaluated classes. Hence, the number of samples considered in each category is not proportional during the training stage. Thus, although the situation described implies a greater complexity in the processing of the model (since it makes it more sensitive to specific characteristics of the impedance signatures), it brings it closer to the real conditions of its operation since, in concrete monitoring situations, the symmetrical acquisition of signals for the different observed states is not always feasible.

In addition, during the training phase, a cross-entropy loss function was applied to the network responses to verify their accuracy in the target values. At the same time, Adam's algorithm (Adaptive Moment Estimation) adjusted the synaptic weights of the network for each iteration. The characteristic curve of the loss function values (whose final value is 0.000965) obtained in this step and the precision values (whose absolute value is 1) are also measured in Figure 16.

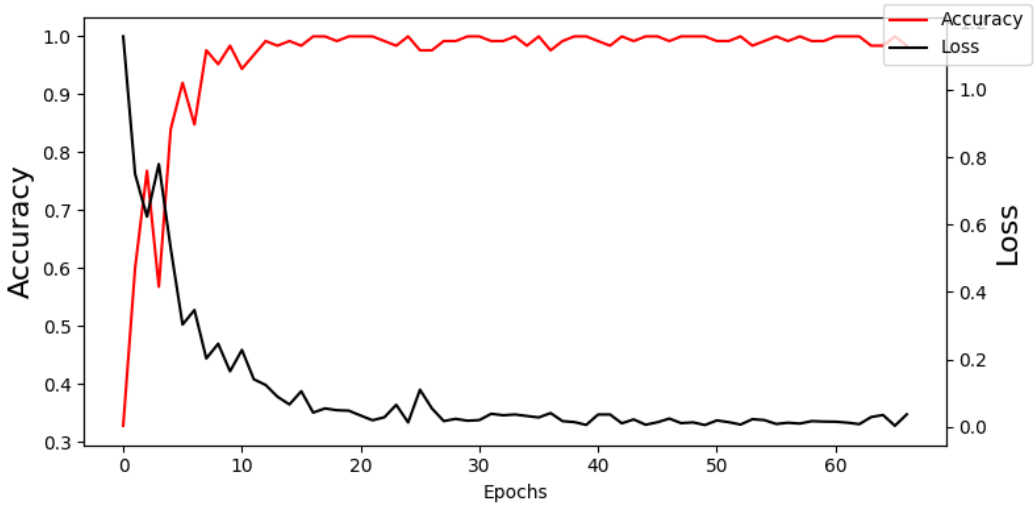


Figure 16: Loss and accuracy function achieved during the 1D-CNN neural network training process.

In Figure 16, when the model reaches the 30th epoch, the adjustment of the synaptic parameters of the network is almost negligible since the magnitude of the observed error is significantly tiny (≈0). Thus, even if in the training configurations of the convolutional neural network used, the number of predefined epochs was 1000, the *EarlyStopping* procedure was applied already in the 67th epoch, thus interrupting the training process and, subsequently, avoiding a possible condition of overfitting of the model.

As the last approach to be adopted in the present case study, the neural model developed here was then evaluated for its effectiveness through the confusion matrix, which compares the values predicted by the model and the target values already known from the structure integrity states. In this sense, Figure 17 compares the values predicted by the model and the pre-established target values for the same, both about the training and test sets.

From the above, it can be observed that the model developed is totally sensitive to the integrity classes imposed in the present case study, reaching 100% of accuracy for all the foreseen conditions, both concerning the training and the test set.

Thus, based on what has been exposed in this section, one can verify the efficiency and sensitivity of convolutional neural models in monitoring the structural integrity in iron plate-like structures. The damage condition is still given as reducing the thickness level of material used.

Furthermore, jointly evaluating the results of Sections 3.1 and 3.3, it can be inferred that the use of the 1D-CNN architecture in support of the SHM method based on electromechanical impedance becomes of great applicability, allowing with a certain level of precision, the delimitation of the states of the structural integrity of the evaluated systems, independently of the environmental and operational conditions applied to the experiment during the stage of acquisition of the monitoring signals.

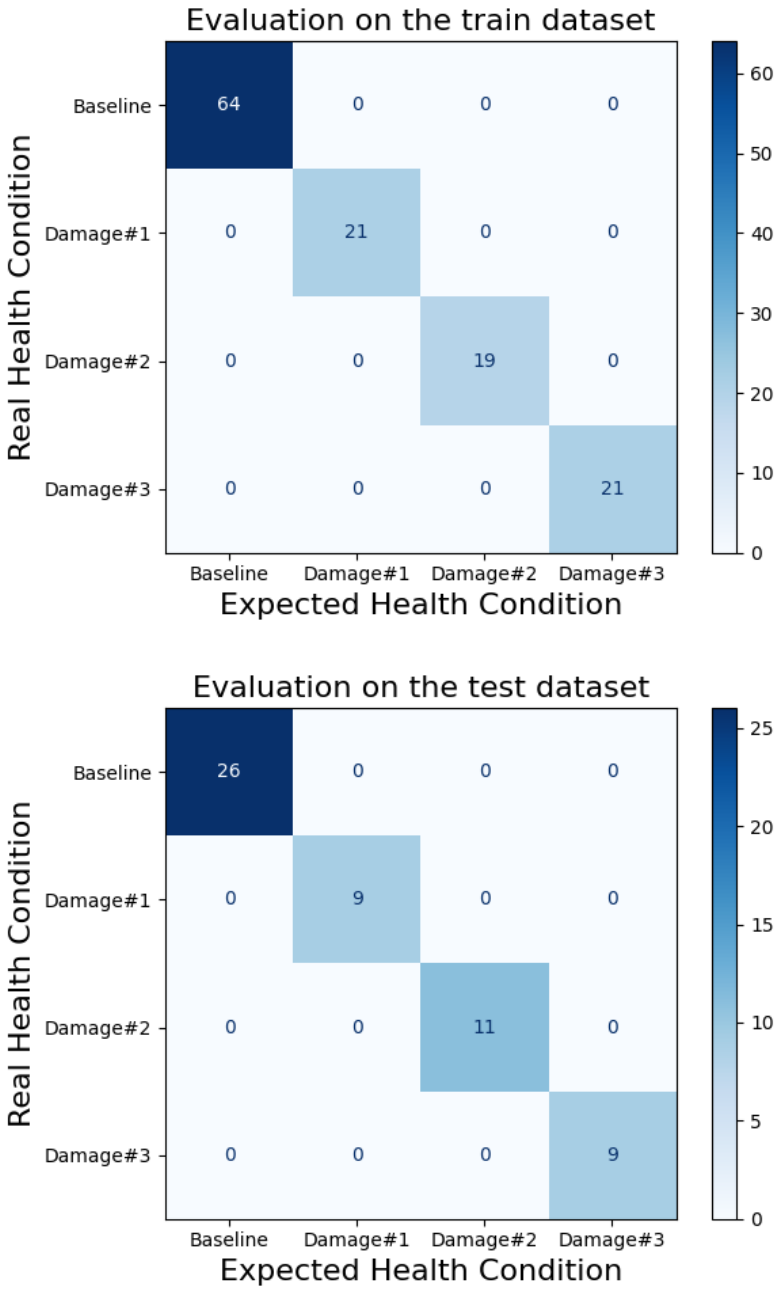


Figure 17: Confusion Matrix of the values predicted by the model in the thickness reduction experiment.

4. Final Remarks

As noted in the previous sections, deep learning models have great flexibility and the ability to represent complex physical phenomena. A peculiarity was observed in the use of CNN, specifically in classifying types of damage in structures due to the lack of a usual pre-processing of the data for temperature compensation. This is expected since Machine Learning techniques present demands of previous feature engineering tasks, unlike Deep Learning techniques that use mechanisms to explore relationships between features, eliminating data pre-processing steps. Still, in the exploratory set using the LSTM networks, it was possible to use these complex relationships to predict a complex

phenomenon such as corrosion in metals. In the final case study, using a plate extracted from a real fuel storage tank, it was possible to present the monitoring of the evolution of the loss of thickness, one of the main problems of corrosive wear explored by the oil and gas industry in this structural type.

Acknowledgments

All authors thank Petrobras—Petroleo Brasileiro S.A. (CEP:21941-915) RJ, Brazil, for their support in this project.

References

S. Albawi, T. A. Mohammed and S. Al-Zawi. Understanding of a convolutional neural network. *International Conf. on Engineering and Technology* (ICET) (IEEE) pp. 1–16. 2017.

S. Abdoli, P. Cardinal and A. L. Koerich. End-to-end environmental sound classification using a 1d convolutional neural network. *Expert Syst. Appl.* 136 252–63. 2019.

M. Afshari. Vibration-and Impedance-based Structural Health Monitoring Applications and Thermal Effects. *Ph.D. thesis — Virginia Tech*, 2012.

A. Agarwal, A. Agrawal, V. Meruane and K. S. Sangwan. Development of a Machine Learning-Based Model for Damage Detection, Localization and Quantification to Extend Structure Life. *Procedia CIRP*, v. 98, p. 199-204, 2021.

B. P. Barella. Machine learning modeling applied to structural health monitoring systems. (In Portuguese: Modelagem com aprendizado de máquina aplicada aos sistemas de monitoramento de integridade estrutural). *156 f. Master's thesis, Modeling and Optimization, Federal University of Goias, Brazil. 2021*.

W. K. P. Barros, D. S. Morais and M. A. C. Fernandes. Skin cancer detection with artificial neural networks (In Portuguese: Detecção de câncer de pele com redes neurais artificiais). In: *XIII Congresso Brasileiro de Ciências da Computação.* [S.l.: s.n.], 2017.

J. P. M. Bento *et al*. Frequency range optimization by the use of ACO and BCO methods in the impedance-based SHM. In: *24th ABCM International Congress of Mechanical Engineering*. 10.26678/ABCM.COBEM2017.COB17-1363: COBEM, 2017.

T. D. Bispo. LSTM Architecture for Cross-Lingual Hate Speech Classification English-PtBR. (In Portuguese: Arquitetura LSTM para Classificação de Discursos de Ódio Cross-Lingual Inglês-PtBR). *Master's thesis, Computer Science/Smart Computing, Federal University of Sergipe*, *Brazil*. 2018.

A. H. Blanco, B. H. Flores, B. Tomás and B. N. Colorado. A systematic review of deep learning approaches to educational data mining. *Complexity*, *Hindawi*, v. 2019, 2019.

Z. Chen, C. Li and R. V. Sanchez. Gearbox fault identification and classification with convolutional neural networks. *Shock and Vibration*, v. 2015, 2015.

F. Chollet. Deep Learning with Python. [S.l.]: New York, NY, USA: *Manning Publications Co*., 2017.

S. Cofre-Martel, P. Kobrich, E. Lopez Droguett and V. Meruane. Deep convolutional neural network-based structural damage localization and quantification using transmissibility data. *Shock and Vibration*, Hindawi, v. 2019, 2019.

N. Eleftheroglou, D. Zarouchas, T. Loutas, R. Alderliesten and R. Benedictus. Structural health monitoring data fusion for in-situ life prognosis of composite structures. *Reliability Engineering & System Safety*, v. 178, p. 40-54, 2018.

F. A. Freitas, R. M. Jafelice and J. W. Silva. A new data normalization approach applied to the electromechanical impedance method using adaptive neuro-fuzzy inference system. *J Braz. Soc. Mech. Sci. Eng*. 43, 475 (2021). https://doi.org/10.1007/s40430-021-03186-z

S. Ghazvineh, G. Nouri, S. H. H. Lavassani, V. Gharehbaghi and A. Nguyen. Application of 2-D Convolutional Neural Networks for Damage Detection in Steel Frame Structures. *arXiv preprint arXiv:2110.15895*, 2021.

N. S. Gulgec, M. Takáč and S. N. Pakzad. Structural damage detection using convolutional neural networks. In: *Model validation and uncertainty quantification, Volume 3. Springer, Cham*, p. 331-337. 2017.

N. S. Gulgec, M. Takáč and S. N. Pakzad. Convolutional neural network approach for robust structural damage detection and localization. *Journal of Computing in Civil Engineering, American Society of Civil Engineers*, v. 33, n. 3, p. 04019005, 2019.

A. Gulli and S. Pal. Deep learning with Keras. [S.l.]: *Packt Publishing Ltda*, 2017.

S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, *MIT Press*, v. 9, n. 8, p. 1735–1780, 1997.

S. Indolia, A. K. Goswami, S. P. Mishra and P. Asopa. Conceptual understanding of convolutional neural network-a deep learning approach. *Procedia Computer Science.* 132 679–88. 2018.

H. Komijani, S. Rezaeihassanabadi, M. R. Parsaei and S. Maleki. Radial basis function neural network for electrochemical impedance prediction at presence of corrosion inhibitor. *Periodica Polytechnica Chemical Engineering*, v. 61, n. 2, p. 128-132, 2017.

X. H. Le, H. V. Ho, G. Lee and S. Jung. Application of long short-term memory (LSTM) neural network for flood forecasting. *Water, Multidisciplinary Digital Publishing Institute*, v. 11, n. 7, p. 1387, 2019.

L. F. Leucas. Use of electromechanical impedance and Lamb waves techniques to identify damage in structures with rivets. (In Portuguese: Utilização das técnicas de impedância eletromecânica e ondas Lamb para identificação de dano em estruturas com rebites). 80 *f. Master's thesis, Mechanical Engineering. Federal University of Uberlândia, Brazil,* 2009.

W. Liang, L. Guixi and D. Hongyan. Dynamic and combined gestures recognition based on multi-feature fusion in a complex environment. *The Journal of China Universities of Posts and Telecommunications, Elsevier*, v. 22, n. 2, p. 81–88, 2015.

J. Melville, K. S. Alguri, C. Deemer and J. B. Harley. Structural damage detection using deep learning of ultrasonic guided waves. In: *AIP Conference Proceedings. AIP Publishing LLC*, 2018. p. 230004.

F. C. D. Menezes, K. P. S. O. Rodriguez Esquerre, R. D. A. Kalid, A. Kiperstok, M. C. D. O. Matos and R. Moreira. Artificial neural networks applied to the clotting process. (In Portuguese: Redes neurais artificiais aplicadas ao processo de coagulação). *Engenharia Sanitaria e Ambiental, SciELO Brasil*, v. 14, n. 4, p. 449–454, 2009.

S. M. Mhatre *et al*. A review paper on artificial neural network: A prediction technique. *International Journal of Scientific and Engineering Research* (IJSER), v. 8, p. 2229–5518, 2017.

F. Millstein. Convolutional neural networks in python: Beginner's guide to convolutional neural networks in python. *CreateSpace Independent Publishing Platform, North Charleston,* SC, USA, 2018.

J. R. V. Moura Jr and V. Steffen Jr. Impedance-based Health Monitoring for Aeronautic Structures using Statistical Meta-modeling. *Journal of Intelligent Material Systems And Structures*, v. 17, p. 1023-1036, 2006.

W. S. Na and J. Baek. A review of the piezoelectric electromechanical impedance-based structural health monitoring technique for engineering structures. *Sensors*, v. 18, n. 5, p. 1307, 2018.

Q. S. S. Nomelini, J. W. Silva, C. A. Gallo, R. M. Finzi Neto, J. R. V. Moura Jr and J. E. Ramos. Statistical Process Control (SPC) of Damage Metrics in the Impedance-based Structural Health Monitoring. *Revista Brasileira de Biometria*, v. 39, p. 7-24, 2021.

L. V. Palomino, J. R. V. Moura Jr, K. M. Tsuruta, D. A. Rade and V. Steffen Jr. Impedance-based health monitoring and mechanical testing of structures. *Smart Structures and Systems*, v. 7, p. 15-25, 2011.

R. J. Portsev and A. V. Makarenko. Convolutional neural networks for noise signal recognition *IEEE XXVIII International Workshop on Machine Learning for Signal Processing* (MLSP). 2018

F. Provost and R. Kohavi. Glossary of terms. *Journal of Machine Learning*, v. 30, n. 2-3, p. 271–274, 1998.

D. S. Rabelo, V. Steffen Jr, R. M. Finzi Neto and H. B. Lacerda. Impedance-based structural health monitoring and statistical method for threshold-level determination applied to 2024-t3 aluminum panels under varying temperature. *Structural Health Monitoring, SAGE Publications Sage UK: London, England*, v. 16, n. 4, p. 365–381, 2017a.

D. S. Rabelo, K. M. Tsuruta, D. D. Oliveira, A. A. Cavalini Jr, R. M. Finzi Neto and V. Steffen Jr. Fault detection of a rotating shaft by using the electromechanical impedance method and a temperature compensation approach. *Journal of Nondestructive Evaluation, Springer*, v. 36, n. 2, p. 25, 2017b.

Z. Rastin, G. Ghodrati Amiri and E. Darvishan. Unsupervised structural damage detection technique based on a deep convolutional autoencoder. *Shock and Vibration*, v. 2021, 2021.

S. W. F. Rezende, J. R. V. Moura Jr, R. M. Finzi Neto, C. A. Gallo and V. Steffen Jr. Convolutional neural network and impedance-based SHM applied to damage detection. *Engineering Research Express*, v. 2, n. 3, p. 035031, 2020.

S. W. F. Rezende, B. P. Barella, J. R. V. Moura Jr. Damage Identification of Vehicle Brake Disks by the use of Impedance-Based SHM and Unsupervised Machine Learning Method. *International Journal of Advanced Engineering Research and Science*, v. 7, p. 324-330, 2020.

S. W. F. Rezende. Deep Learning Techniques Applied to Electromechanical Impedance-based Structural Health Monitoring Method. (In Portuguese: Técnicas de aprendizado profundo aplicadas ao monitoramento de integridade estrutural por impedância eletromecânica). 2021. 143 *f. Master's thesis, Modeling and Optimization, Federal University of Goias, Brazil*. 2021.

P. Sharma and A. Singh. Era of deep neural networks: A review. In: *IEEE. 8th International Conference on Computing, Communication and Networking Technologies* (ICCCNT). [S.l.], 2017. p. 1–5.

A. Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena, Elsevier*, v. 404, p. 132306, 2020.

R. B. Silva, F. I. Ferreira, F. G. Baptista, P. R. Aguiar, R. S. Ruzzi, H. B. Hubner and E. C. Bianchi. Electromechanical impedance (EMI) technique as alternative to monitor workpiece surface damages after the grinding operation. *The International Journal of Advanced Manufacturing Technology*, v. 98, n. 9, p. 2429-2438, 2018.

O. Trekhleb and L. Xie. Homemade Machine Learning - Interactive Machine Learning Experiments, GitHub. Available at: https://github.com/trekhleb/homemade-machine-learning#machine-learning-map. Accessed on: 03/06/2022, at 14:00.

K. M. Tsuruta, D. S. Rabelo, C. G. Guimarães, A. A. Cavalini Jr, R. M. Finzi Neto and V. Steffen Jr. Electromechanical impedance-based fault detection in a rotating machine by using an operating condition compensation approach. In: *International Society for Optics and Photonics. A Tribute Conference Honoring Daniel Inman*. [S.l.], 2017.

D. Verstraete, A. Ferrada, E. L. Droguett, V. Meruane and M. Modarres. Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings. *Shock and Vibration*, v. 2017, 2017.

S. Yan. Understanding LSTM and its Diagrams. ML Review: Machine Learning Research, Projects and Educational Materials. 2016. Available at: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714.

W. Yu, F. Zhuang, Q. He and Z. Shi. Learning deep representations via extreme learning machines. *Neurocomputing*, *Elsevier*, v. 149, p. 308–315, 2015.

X. Zhao, R. L. Royer, S. E. Owens and J. L. Rose. Ultrasonic Lamb wave tomography in structural health monitoring. *Smart Materials and Structures*, v. 20, n. 10, p. 105002, 2011.

R. Zhao, J. Wang, R. Yan and K. Mao. Machine health monitoring with LSTM networks. In: *IEEE. 10th International Conference on Sensing Technology* (ICST). [S.l.], 2016. p. 1–6.