COB-2021-1835 DEVELOPMENT OF A DEFECT DETECTION METHODOLOGY IN BEAMS USING MODAL ANALYSIS DATA IN OPTIMIZED ARTIFICIAL NEURAL NETWORKS.

Leonardo Costa Tavares
Wesley Padilha da Silva Homem
Universidade Federal do Pará, R. Augusto Corrêa, 01 - Guamá, Belém - PA
tavaresleonardo15@gmail.com
wesley.homem@itec.ufpa.br

Javier Salvador Ribeiro Ferreira
javiersrf@gmail.com

Camila Ferreira dos Santos
camila.ferreira.santos@itec.ufpa.br

Eduardo Garcia Mendes
eduardo.mendes@itec.ufpa.br

Leonardo Dantas Rodrigues
leodr@ufpa.br

Abstract. *Many researchers have dedicated themselves to the development of techniques for Structural Health Monitoring, which generates a large number of information that needs to be properly treated. This work deals with the development of a defect detection system in beams using two different types of trained and validated backpropagation artificial neural networks (ANN), cascade-forward and feedforward, with data from numerical modal analysis using the finite element method. To ensure a large volume of data to improve the effectiveness of ANNs, a python code was developed that communicates directly with Ansys APDL, automatically generating several models of beams with cracks with different positions and depths. The generated and input data are sent to ANNs also programmed in Python, being divided into groups for training, validation and testing. And then, with the data prepared, a grid search is used in the ANN in order to optimize its design by finding the number of neurons in the hidden layers that obtained the best result within a predetermined range. For these optimized ANNs, the one that obtained the best results showed a percentage error of 4.97% in relation to the test data, which can be considered quite satisfactory.*

Keywords: *Structural Health Monitoring, Artificial neural networks, Python, Damage Detection*

1. INTRODUCTION

The presence of damages affects the rigidity of the structure and, consequently, its mechanical response to certain demands (Maury *et al.*, 2019). To reduce the risk of failures in structures, ideally, defects are detected and treated as quickly as possible. For large complex structures, more traditional methodologies, such as visual inspection in search of defects, become impractical or ineffective. Thus, auxiliary tools that can give indications of the existence of defects and that can also estimate their location and severity can be very useful to better guide visual inspections and with instruments that can more accurately indicate the characteristics of the damage.

Many methodologies for monitoring structures, even in real time, have been developed over time. Many of these, to make it possible to analyze and treat large volumes of information provided by sensors, make use of artificial intelligence techniques, such as artificial neural networks (ANN), used, for example, by Abdeljaber *et al.* (2017) to detect structural damage analyzing real-time vibration. The quality of learning these ANNs to make them able to correctly interpret the data provided depends on their architecture, parameters and hyperparameters and also on the data to be used as references in their training.

Many works in the literature, such as Filho (2017), use modal characteristics of the structure in the detection and characterization of damages, as these change with the existence of defects in the component due to loss of stiffness and mass. Parameters such as natural frequencies and modal forms of damaged components are analyzed and compared with non-damaged components for this purpose. In this sense, the more information there is for comparisons, the more efficient the method will be.

In this work, natural frequencies obtained from experimental modal analyzes in beams with cracks were used as input data to train ANN in order for them to learn to detect and characterize defects from these parameters. Two network architectures were used for efficiency comparison purposes.

2. THEORETICAL BACKGROUND

2.1 Structural Health Monitoring

Due to the effect of weather and other weather conditions, such as earthquakes, corrosion or thermal effects, damage to structures occurs, which is a normal process and considered for designing engineering projects. To prevent damage from spreading and leading to the compromise of entire structures, several damage detection techniques have been created and continue to evolve.

According to Gadea (2002) and Fan *et al.* (2021), most damage detection methodologies are related to the dynamic behavior of the structure, as they provide a greater amount of information, but they are more expensive and require more sophisticated equipment than methodologies related to static behavior. Filho (2017) states that the insertion of damage to the structure generates a change in the rigidity of the body, due to a reduction in the cross section of the cracked elements, and modifies its responses to mechanical phenomena, therefore, a change in the natural frequency and verified accordingly with the position and depth of the damage, which is a widely used criterion for damage detection.

2.2 Modal analysis

According to Rao (2008), the modal analysis is the process of determining the dynamic characteristics of a structure based on its natural frequencies, vibration modes and damping factors, aiming to mathematically model its behavior.

2.2.1 Analytical method

A structure widely used in engineering and relatively simple to perform a modal analysis is a beam, as it can be modeled as a continuous system, so its behavior is governed by partial differential equations (Rao, 2008). For a beam under ideal conditions, we can analytically calculate its natural frequencies from Eq. (1)(Mendes, 2019),

$$f = \beta^2 (\frac{EI}{\rho A}), \tag{1}$$

Where $\mathbf{E}$ is the young's modulus, $\mathbf{I}$ the moment of inertia, ρ the density, $\mathbf{A}$ the cross-sectional area, β is a boundary condition parameter, changing according to the extremity: free-free, embedded, etc; and with the vibration mode to be analyzed.

2.2.2 Numerical method

One of the problems with Eq. (1) is that it doesn't work for more complex systems, making it difficult to analytically solve the problem. An alternative solution is the use of finite element methods, or numerical solution, in which the structure to be analyzed is discretized and represented by "nodes", together forming a three-dimensional element, which in sufficient quantity can bring accurate results when compared to a real situation (Fonseca, 2006).

According to Filho (2008), commercial finite element software has a library of elements that represent different geometry and physical behavior, and it is extremely important to know well the physical behavior that you want to evaluate, in order to choose the correct element. According to this author, each element has a specific stiffness matrix, and when we represent the analyzed structure from these elements, the stiffness of the structure is counted from the stiffness of the selected elements. After selecting the element, obtaining the stiffness matrix, and applying the boundary conditions, the software will solve Eq. (2),

$$[M]\{\ddot{x}\} + [K]\{x\} = [F], \tag{2}$$

Where $[M]$ is the mass matrix, $[K]$ the stiffness matrix, $[\ddot{x}]$ and $[x]$ are, respectively, the acceleration and displacements of the nodes. $[F]$ are the forces applied to the nodes that make up the structure.

It is possible to observe from Eq. (2) that the software assumes that the analyzed structure does not have significant damping, solving the problem without the use of the damping coefficient, being considered a conservative system (Mendes, 2019). Therefore, finite element software calculates the vibration modes and the value of natural frequencies by solving an eigenvalue and eigenvector problem.

2.3 Artificial neural networks

The ability to learn is considered essential for intelligent behavior. Activity such as memorizing, observing and exploring situations to learn facts, improving motor/cognitive skills through practices and organizing new knowledge into appropriate representations can be considered activities related to learning (Faceli *et al.*, 2011). Artificial intelligence techniques generalize this concept to computers, creating so-called machine learning.

In machine learning, computers are programmed to learn from past experience. For this, they employ a principle called induction, in which conclusions are drawn from a particular set of examples (Srivastava *et al.*, 2014). Thus, the idea of artificial neural network (ANN) can be consolidated, which tries to replicate the human brain processing these sets of examples through a complex structure interconnected by elements called neurons.

Artificial neural networks are composed of several processing units (Figure 1) called neurons, these neurons are connected by weighted channels and perform operations with the data that arrives on them through these channels.

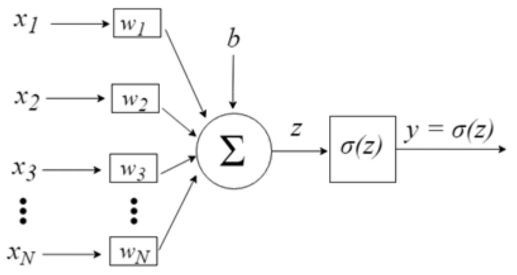

Figure 1. Computational model of a neuron. Source: Leite (2018)

In Figure 1, “b” is the term “bias”. Thus, when the components of the X input vector are null, the output must be just b (Khan, 2019).

The operation carried out in each unit occurs by the arrival of the data, in Figure 1 each data comes from a different input, these data are multiplied by a weight (represented in Figure 1 by the vector w), which indicates the influence on the output of the referring unit. that connection is then made the weighted sum of everything that arrived at the neuron and that sum is used as an argument in a chosen activation function. Thus, the output obtained can be analyzed according to the proposed problem. In the neural network literature, each layer except the input layer is considered to have all these operations, so this is often omitted from the Figures and only the neuron is represented.

2.3.1 Feedforward network

Different ANN architectures are presented in the literature, however the feedforward type network, illustrated in Figure 2, is the best known and used in several cases of artificial intelligence.

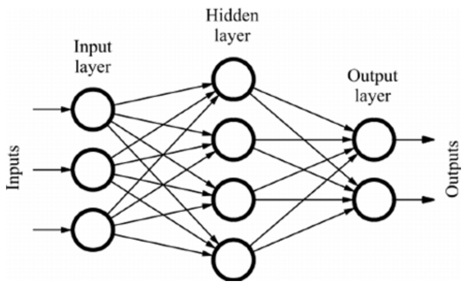

Figure 2. Sample of a one hidden layer feedforward neural network. Source: Quiza and Davim (2011)

The feedforward network is divided into layers and each neuron in one layer connects to all neurons in the next layer. This connection has its own numerical weight, so training a net basically consists of changing the values of these weights until the results generated in the last layer (output layer) are satisfactory. Therefore, there needs to be a significant sample of input and output data for training a network so that ANN can then be applied to unknown data.

2.3.2 Backpropagation Algorithm

As already defined, the training of an artificial neural network occurs by changing the weights of the connections between neurons, however, it is necessary to establish a criterion for this change. The backpropagation algorithm is one of the most used in ANN training, it consists of the complete propagation of input data throughout the neural network to obtain an output, this output is then compared with the expected results and a cost between them can be calculated. The gradient of this cost is then propagated back to the network, taking the opposite path, until reaching each weight. Thus, the back propagated gradient defines how much each weight will be penalized by the calculated cost, until this cost is minimized.

Thereby, a formulation can be defined for changing the weights of the neural network using the backpropagation algorithm, which is shown in Eq. (3),

$$W_{new} = W_{current} - \alpha(\frac{\partial C}{\partial W_{current}}), \tag{3}$$

Where W represents the weight and is the learning rate of the neural network, which is used to control the step size in the cost curve, so if it is too large it may be that the minimum cost is lost and if it is too small , training may take too long.

2.3.3 Cross validation

Cross validation is a statistical technique that determines, during training, the generalizability of the constructed network (Abu-Mostafa *et al.*, 2012). For this, the training data must be divided into three sets: the training set, used to train the network, the validation set, used to assess the generalizability of the network during training, and the test set, which will be used to evaluate the already trained network.

Cross validation works as follows: after training the network for a pre-defined number of epochs (training iterations), the training is stopped and the network receives the validation data to be tested. The process is repeated until the network performance stabilizes with the validation data at a value considered acceptable. The network, now trained, receives new values (test set) and its generalizability is tested.

2.3.4 Cascade-forward network

The cascade-forward network also uses the backpropagation algorithm and has a structure similar to feedforward, illustrated in Figure 2, however each neuron in a layer connects to all neurons in all subsequent layers, thus creating a greater number of items.

While feedforward networks can potentially learn virtually any input and output relationship, cascade-forward networks with more connections between layers can learn complex relationships more quickly. Thus, additional connections can improve the speed and accuracy with which the network learns the desired relationship (Mendes, 2019).

3. METHODOLOGY

3.1 Ansys APDL data generation

To feed the Networks, it is necessary to create a database with large amounts of samples, relating data used in the input and output of the ANN, which is difficult to perform experimentally, since this large number of samples demands a large amount of time. One way to get around this is to use data obtained through numerical modeling. In this work, the modeling was performed by the finite element method, using the Ansys software.

The python language has its own library dedicated to computational simulation in ansys APDL, which is the python library, created with the objective of facilitating software programming and opening doors for automation in a simpler way (Kaszynski, 2020). The pyansys library has the option to load the code and create a separate file in APDL format, which is a slower process, so we used the option to run the simulation using python's own compiler which, synchronized with the ansys APDL, performs the calculations without having to open the software, making the process faster and more practical, as the compiler can start a simulation immediately after the other is finished, something that ansys APDL does not allow due to its interface and programming limitations. The code has as "inputs" the position of the damage, being the x position in relation to the fixed extremity and the y position is the damage depth in relation to the upper face of the beam; and has as "outputs" the first 3 natural frequencies of the beam. The numerical modal analyzes were carried out on a beam fixed at one of the ends, in which a crack was inserted in different positions and with different depths. Each model was simulated with only one defect. After a convergence analysis, the final mesh of the models had 4900 elements of the SOLID186 type and 23235 nodes. Table 1 illustrates the properties of the beam used. Figure 3 shows one of the models, already with the final mesh, where the crack was inserted 45 cm from the fixed extremity (50% of the length) and with a depth of 50% of the height of the beam.

Table 1. Beam properties. Source: The autor

Beam properties	
Length	0.9 m
Height	0.035 m
Width	0.025 m
Density	2770 kg/m³
Poisson	0.33
Young modulus	71 GPa
material	aluminum

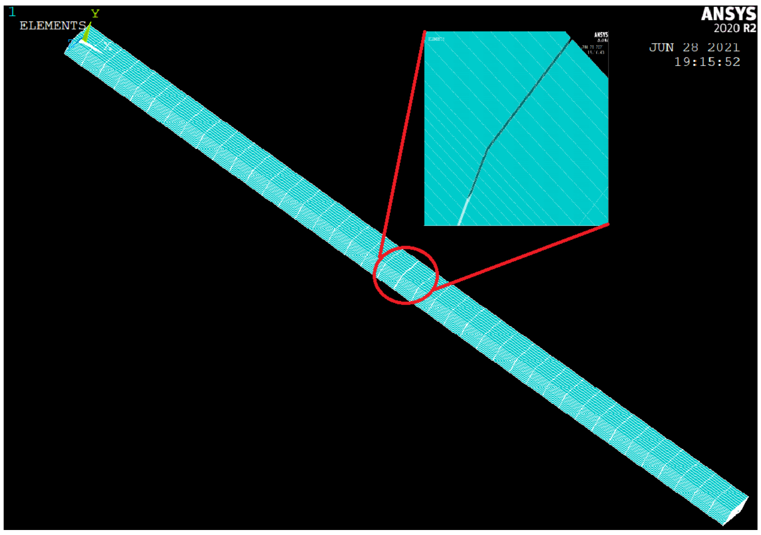

Figure 3. Generated mesh. Source: The autor

3.2 ANN creation and optimization

3.2.1 Data Treatment

Data pre-processing techniques are often used to improve data quality by eliminating or minimizing problems caused by incorrect, inconsistent, duplicate or missing values. This improvement can facilitate the construction of models that are more faithful to the real data distribution and, consequently, make it easier to interpret the patterns and extract a model (Mendes, 2019).

Therefore, a program was created in the python language that removes the inconsistencies regarding the natural frequencies of the defective beam. Inconsistencies can be identified when known relationships between certain attributes are violated (Faceli *et al.*, 2011). Thus, considering that the presence of cracks in structural elements tends to cause a decrease in their natural frequencies, data that contradict this information were eliminated, that is, natural frequencies of the damaged beam that, for some reason, proved to be higher than the natural frequencies of the studied beam without damage (Mendes, 2019).

After the incoherent data is removed, it goes through a normalization step. It is standard practice to normalize the input vector before applying it to the network. In this way, the program works with inputs in a normalized range, avoiding inconsistencies with activation functions.

Thereby, a program was created that normalizes the data set entries, so that they are in the same group from 0 to 1. The normalized data were obtained according to Eq. (4),

$$Relativefrequency_u = \frac{f_{du}}{f_{nu}}, \tag{4}$$

Where f_{du} represents the Umpteenth natural frequency of the damaged beam and f_{nu} is the Umpteenth natural frequency of the undamaged beam. Finally, after being normalized, the data is passed on to another program that randomly separates 70% for training, 15% for validation and 15% for testing.

3.2.2 Creation of the Feedforward ANN

With the data already treated, a feedforward neural network was created, using a code in Python language, to be trained with this data in a classification problem. The idea behind the problem was to separate the beam into eight sections and the network must be able to predict in which section the defect will be located by analyzing the first three natural frequencies of the beam, that is, the network has three neurons in the input layer.

The splitting of the beam started at the fixed extremity and was up to 80% of the total length of the structure, with each section representing 10% of that length. A full analysis of the beam was not performed, as the end opposite the bezel is free and therefore the natural frequencies of a beam with damage close to this area would not vary much from the frequencies of the undamaged beam, so it is expected to be the zone in which the network would have the most difficulty in classifying.

It was stipulated that this neural network would have eight neurons in the output layer, each one representing an area, that is, if a neuron is activated, it means that the defect is present in the location it represents. As only one defect is being worked on, a single neuron will activate per output, those activated will be represented by the number 1 and those deactivated by the number 0, the neuron that will activate is the one with the closest output to 1.

A total of three hidden layers was also stipulated for this network, the choice of this number of layers was motivated by tests with neural networks using different amounts of hidden layers, and it was observed that this value was the one that presented the best results for the problem in question. And the number of neurons in these layers was defined as the hyperparameter to be optimized.

Finally, it remains to establish the activation function for each neuron and the cost function for the network results. The activation function used for this ANN was the Hyperbolic Tangent Sigmoid (TANSIG), which is shown by Eq. (5). The TANSIG function transforms the input into a value in the range of -1 to 1 as an output. It is one of the most used activation functions in neural networks and because of that it was selected.

$$F(X) = \frac{1}{1+e^{-x}}, \tag{5}$$

The cost function used for this network is the Binary Cross-Entropy (BCE), which, according to Khan (2019), is suitable for binary classification problems. The BCE is shown by Eq. (6):

$$C(Y,Y') = \frac{1}{m}\left[-Y\; o\; ln(Y') - (1-Y)\; o\; ln(1-Y')\right], \tag{6}$$

Where Y represents the expected output vector, Y' is the output vector estimated by the neural network, m is the total number of outputs in the data set, and "o" represents the elementary product between the vectors, also called Hadamard product.

3.2.3 Neural Network Optimization

The purpose of many machine learning tasks can be summarized as training a model that minimizes some predefined cost function, applied to a test set. Because of this, an optimization of an ANN can occur by searching for hyperparameters that produce an optimal model and minimize the cost on the test data, thus, the optimized network is able to present its best results.

As already mentioned, the optimization of this network was done by defining the best number of neurons in the hidden layers. For this, the Grid Search algorithm, suggested by Bergtsra and Bengio (2012), was used.

This algorithm basically consists of selecting the best hyperparameter, that is, the one with the best results, within a defined range. For this work, for example, a range of neuron numbers from 5 to 25 was stipulated and the optimal number was decided by the one with the highest percentage of correct answers, making the coded algorithm work like the flowchart in Figure 4:

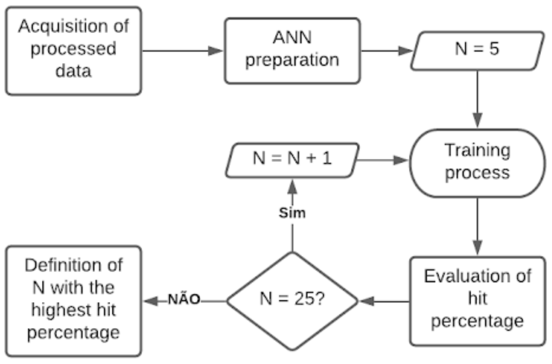
Acquisition of processed data
ANN preparation
N = 5
Training process
N = N + 1
Sim
Evaluation of hit percentage
N = 25?
NÃO
Definition of N with the highest hit percentage

Figure 4. Flowchart for the optimization algorithm. Source: The autor

Where N represents the number of neurons in hidden layers. Thus, the number of 20 neurons was selected as the optimal number for this neural network.

3.2.4 Training Parameters

Once the structural parameters of the network have been defined, it remains to establish some training parameters, which are listed in Table 2:

Table 2. ANN feedforward training parameters. Source: The autor

Training parameter	
learning rate (α)	0.5
Number of training seasons	500
Maximum number of validations	4000

3.2.5 Creation of a Cascade-forward ANN

After applying the feedforward network to the generated data, the results shown later were analyzed. The results obtained showed that this ANN architecture did not demonstrate such a good performance for the classification problem with lighter defects. Therefore, taking into account that monitoring the integrity of structures is not limited to severe defects, a new neural network architecture was proposed, which, according to Thatoi and Khuntia (2017) and Badde *et al.* (2013), presents good results for the analysis of defects in beams, the cascade-forward.

For this network, a regression problem was presented, that is, the neural network tries to predict the exact values of both the location of the damage along the length of the beam and its depth. Thus, for regression problems, talking about hit percentage does not make much sense, so a percentage error is calculated that determines how close the predicted output is to the real one. Therefore, in addition to the three neurons in the input layer referring to the natural frequencies of the beam, this network has only two outputs, one for the location of the defect and the other for its depth.

Also, only one hidden layer was used, which went through the same optimization described above, (however, the interval for grid search was 5 to 15 neurons), now using as decision parameter the quantity that presented the smallest percentage error, resulting on an optimal number of 10 neurons for that layer.

The chosen cost function for this regression problem is the average squared error, shown in Eq. (7):

$$C(Y,Y') = \frac{1}{2m}(Y - Y')^2, \tag{7}$$

Finally, for the function of activating neurons, the use of TANSIG, in the hidden layer, and the linear function (Purelin), in the output layer, was established. The other training parameters are described in Table 3:

Table 3. ANN cascade-forward training parameters. Source: The autor

Training parameter	
learning rate (α)	0.5
Number of training seasons	500
Maximum number of validations	4000

4. RESULTS

4.1 ANN feedforward evaluation

500 numerical models were generated for the data sets, and their inconsistencies were removed so that in total 384 valid models were left, equivalent to 76.8% of the original quantity. By analyzing the discarded data, it was seen that most of them are located very close to the free end and the few that are not in this location have very low severity (it is important to make it clear that only some less severe data were removed). However, it is noteworthy that the differences between the natural frequencies of these samples and those of the beam without defects are low, so for more severe defects and in positions with more relevant bending effects, the model generated consistent results. Thus, 15% of the data already filtered were destined to the test set, another 15% to the validation set, and the rest was used for network training.

With the data already processed and prepared, the ANN feedforward was trained and validated and, thus, it was possible to evaluate its results. According to the program, the network obtained the highest percentage of correct answers, for the validation data, of approximately 57.8%. Using the weights that obtained this result, the network was tested with the test data and obtained a percentage hit of 54.39%. Therefore, it is correct to state that the network did not present good accuracy to identify the location of these damages for the proposed problem.

With the analysis of previous results, it was seen that the network was not able to adequately learn the proposed problem. Because of this, in order to ease the problem and, consequently, improve the performance of the ANN, the data that presented damage with a depth of less than 40% were removed. Based on this modification, the neural network was trained again and below, represented by Figure 5, is the graph of the percentage of correct answers as a function of the number of iterations referring to the validation data.

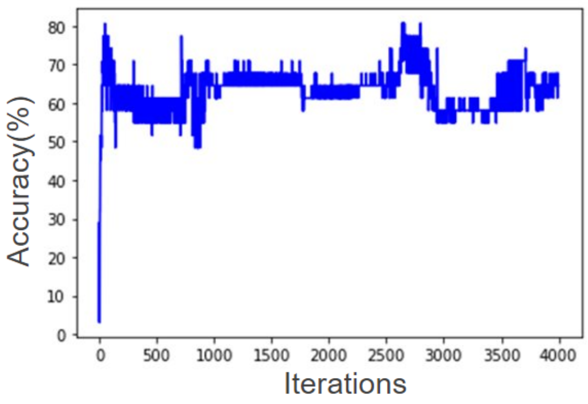

Figure 5. Percentage accuracy for severe defects. Source: The autor

From this graph, it is noticed that the neural network presented a hit percentage of a little higher than 80% regarding the validation data. Using the weights that obtained this result, the network was tested with the test data and obtained a percentage hit of 64.52%, already showing an improvement in relation to the previous analysis.

4.2 ANN Cascade-Forward Evaluation

As mentioned in item 3.2.5, the cascade-forward network was proposed in order to avoid the limitation of deeper damage. For her, a regression problem was given and its function is to predict the location of the defect and its depth. Thus, the data generated and properly treated, without limitation of severity, were used for the training of this ANN.

The first analysis can be made on the graph in Figure 6, which represents the average percentage error, on the validation data, as a function of the iteration number:

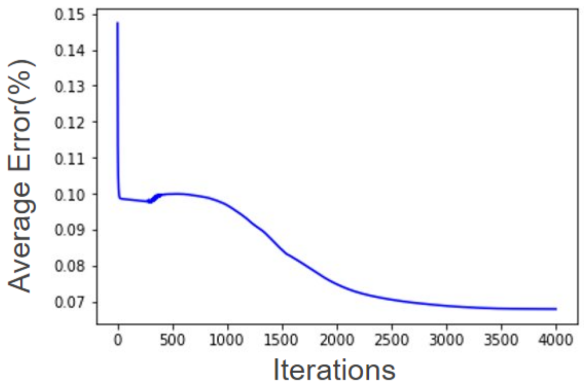

Figure 6. Average error as a function of the number of iterations. Source: The autor

By analyzing Figure 6, it is possible to see that this ANN presented a much less noisy and variable learning when compared to the feedforward network. It also presented, for the validation data, a minimum average error of approximately 6.40%. As was done previously, the weights referring to this result were used with the test set, to finally measure the generalizability of this ANN, resulting in an average error of 4.97%, even lower than that obtained for validation data, this result is an error combined between locations and depths.

With the low percentage error obtained, two column graphs were also plotted for ten random locations and depths of the test data, in order to obtain a clearer visualization of the results obtained, which are shown in Figure 7:

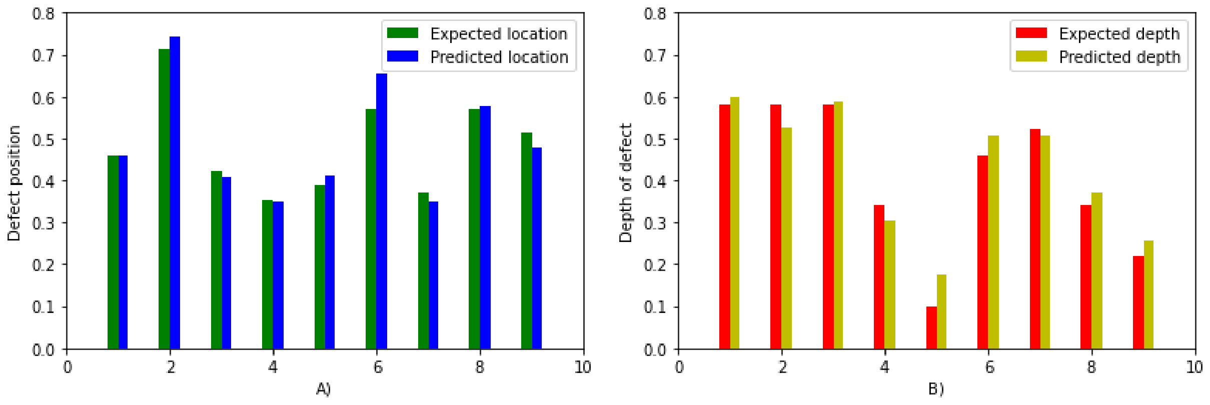

Figure 7. Bar charts for ANN Cascade-forward. Source: The autor

In Figure 7 A) represents the locations of the defects and B) represents their depths. The error only for the position of the defect was 5.77% and for the depth was 4.17%, showing that the network had a better performance in predicting the severity of the damage.

By analyzing the results and Figure 7, it was evident that the cascade-forward network performed better than feedforward, being able to estimate with high precision both the location of the defect and its severity.

5. CONCLUSION

In this work, two ANN backpropagation architectures were used to assist in the detection and characterization of cracks in beams from information of natural frequencies from numerical modal analyzes performed with the FEM.

Automating the generation of models from a python command code integrated with Ansys APDL proved to be very useful, since a large amount of data is desirable for a better learning of ANN. 500 data sets were generated for this work with this methodology.

ANN feedforward, which are widely used in various types of applications, showed limited effectiveness, with accuracy levels of 54.39%. For more severe defects, with depths greater than 40% of the thickness, this percentage of correctness increased to 64.52%, which cannot yet be considered a satisfactory result.

The cascade-forward network, on the other hand, presented a very satisfactory efficiency. With an overall average error of 4.97%, where, separately, for the locations the error was 5.77% and for the depths it was 4.17%.

Although the regression problem proposed to this network is different from the ANN feedforward classification problem, if intervals were stipulated for defect zones above the locations in Figure 7, these intervals being limited to 5% in length above and below the actual outputs From this graph, the network would hit 8 of the 9 classification areas created for this data, showing superiority even for classification problems.

New analyzes also taking as a parameter modal curvatures are in progress and should be considered and compared with the current analysis in future works.

6. REFERENCES

Abdeljaber, O., Avci, O., Kiranyaz, S., Gabbouj, M. and Inman, D.J., 2017. "Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks". *Journal of Sound and Vibration*, Vol. 388, pp. 154–170.

Abu-Mostafa, Y.S., Magdon-Ismail, M. and Lin, H.T., 2012. *Learning from data*, Vol. 4. AMLBook New York, NY, USA:.

Badde, S.D., Gupta, A.K. and Patki, V.K., 2013. "Cascade and feed forward back propagation artificial neural network models for prediction of compressive strength of ready mix concrete". *IOSR Journal of Mechanical and Civil Engineering*, Vol. 3, No. 1, pp. 1–6.

Bergtsra, J. and Bengio, Y., 2012. "Random search for hyper-parameter optimization". *Journal of Machine Learning Research*, Vol. 13, pp. 281–305.

Faceli, K., Lorena, A.C., Gama, J. and Carvalho, A.C.L.P.F., 2011. *Inteligência Artificial: Uma abordagem de aprendizado de máquina*. Editora LTC, Rio de Janeiro.

Fan, G., Li, J. and Hao, H., 2021. "Dynamic response reconstruction for structural health monitoring using densely connected convolutional networks". *Structural Health Monitoring*, Vol. 20, No. 4, pp. 1373–1391.

Filho, A.A., 2008. *Elementos Finitos: A Base da Tecnologia CAE / Análise Dinâmica*. Editora Érica, São Paulo.

Filho, M.V.M.O., 2017. *Identificação experimental de trincas em vigas para monitoramento de integridade estrutural*. Ph.D. thesis, Setor de Tecnologia, Universidade Federal do Paraná, Curitiba.

Fonseca, A.V., 2006. *Melhoria do processamento de celulose industrial utilizando Método de Elementos Finitos – MEF*. Master's thesis, Tecnologia dos Materiais e Processos de Fabricação, Universidade de Taubaté, Taubaté.

Gadea, A.S.M., 2002. *Identificação de danos estruturais a partir das funções de resposta em frequência (FRF)*. Master's thesis, COOPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Kaszynski, A., 2020. "Pyansys project". GitHub repository, https://github.com/pyansys. Accessed 20 February 2021.

Khan, R., 2019. "Nothing but numpy: Understanding & creating binary classification neural networks with computational graphs from scratch". Towards Data Science, https://towardsdatascience.com/nothing-but-numpy-understanding-creating-binary-classification-neural-networks-with-e746423c8d5c. Accessed 6 May 2021.

Leite, M.T., 2018. "Redes neurais, perceptron multicamadas e o algoritmo backpropagation". Medium, https://medium.com/ensina-ai/redes-neurais-perceptron-multicamadas-e-o-algoritmo-backpropagation-eaf89778f5b8. Accessed 19 May 2021.

Maury, M., Sadarang, J. and Panigrahi, I., 2019. *Detection of crack in structure using dynamic analysis and artificial neural network*. Engineering Solid Mechanics, Canadá.

Mendes, M.S.F., 2019. *Identificação de danos em viga utilizando redes Neurais Artificiais*. Master's thesis, Trabalho de conclusão de curso (Graduação), Universidade Federal do Pará, Belém.

Quiza, R. and Davim, J.P., 2011. "Computational methods and optimization". In *Machining of hard materials*, Springer, pp. 177–208.

Rao, S., 2008. *mechanical vibrations*. Editora Pearson Prentice Hall, Sâo Paulo.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. "Dropout: a simple way to prevent neural networks from overfitting". *The journal of machine learning research*, Vol. 15, No. 1, pp. 1929–1958.

Thatoi, D. and Khuntia, G.P., 2017. "Fault diagnosis of cracked beam structure using advanced neural network techniques". *International Journal of Engineering Research & Technology (IJERT)*, Vol. 6, pp. 932–943.

7. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.