Artificial Neural Networks Applied in Mechanical Structural Design

Article · April 2021

DOI: 10.5281/zenodo.4669797

3 authors:

João Alves Ribeiro
University of Porto
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Luís Filipe Gomes
University of Minho
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Sérgio M O Tavares
University of Porto
17 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Journal of Computation and Artificial Intelligence in Mechanics and Biomechanics ISSN 2184-8971
Volume 1, Issue 1, Page 14-21, 2021

Artificial Neural Networks Applied in Mechanical Structural Design

J.P.A. Ribeiro[1], L.F.F. Gomes[2], S.M.O. Tavares[1]

[1] *Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal, {jp.ar@hotmail.com; sergio.tavares@fe.up.pt}*
[2] *University of Minho, Campus de Gualtar, 4710-057, Braga, Portugal, luisgomes24.97@gmail.com*

Abstract
Artificial Intelligence has brought many new problem-solving approaches to society in the last few years. In Artificial Intelligence domain, the Machine Learning techniques are being very successful mainly because of their ability to learn. The main objective of this article is the research, development and application of Artificial Intelligence techniques for mechanical structural design. Selected method is based on the Artificial Neural Networks, that allows the prediction of a certain variable based on a given set of data. The applied artificial network was developed in Python with TensorFlow library and the database was prepared using the finite element software, Ansys, with parametric language for automatic data extraction. A classic solid mechanics case study, comprising a plate with a central hole subjected to uniaxial remote stress, is explored. It is intended to obtain the stress distribution for a plate with hole radius between 25 and 50 mm. With Artificial Neural Networks, a substantial reduction in the simulation time is observed, being, approximately, 79 times faster when compared to the solution time of the conventional finite element approach. The developed neural network has a relative average error of about 4.57%, which is considered satisfactory given that it is a first application of these networks in this domain. In conclusion, with this work it is possible to highlight the potential advantages of Artificial Neural Networks in applied to stress/strain calculation in solid mechanics: shorter response time, less computational resources and problem simplification, in detriment of a lower resolution/accuracy of structural behaviour. Optimization procedures and digital-twin concepts can take advantage of these benefits, enabling near real-time calculations.

***DOI**: 10.5281/zenodo.4669797*

Article Info

Keywords
Artificial Neural Networks
Finite Element Method
Plate with Central Hole
Structural Mechanics
Artificial Intelligence

Article History
Received: *09/07/2020*
Revised: *04/12/2020*
Accepted: *19/02/2021*

1. Introduction

Artificial Intelligence (AI) is becoming increasingly present in our daily lives. The benefits that it has brought to society in recent years are remarkable. Its use in daily complex life tasks, like autonomous driving, prove that the human being has more and more confidence in this kind of solutions. The success of AI models in the last few years is closely related to their ability to learn. The models with this property enable the development of Machine Learning (ML) approaches, allowing its exploitation in a wide range of engineering problems.

In a computer program, it is specified to the computer what it is expected to do when some input is presented. On the other hand, in ML systems, instead of programming the actions, they are learned using data of the specific problem that is intended to solve. With the advances in computer architectures and the good results shown by ML technologies, there is a growing interest in the scientific community on this topic. Efforts are being made to expand the AI areas of application and to improve its performance even more and to take advantage of its capabilities.

The purpose of this article is to study, develop and apply AI techniques, namely Artificial Neural Networks (ANNs), in the mechanical design. The specific case study will be the classical problem of a plate with central hole under uniaxial tension. In this way, it is intended, through ANNs, to predict the plate stress field and compare it with the stress field obtained from Ansys software based on the finite element method. The comparison terms are the model accuracy, the computational costs and the simulation time.

2. Artificial Neural Networks

Connectionist Systems, more commonly referred to as ANNs, are ML models inspired by the human neurological system. Perhaps the first ANN architectures were made to simulate the human brain, they have become powerful tools to solve a great variety of problems. One of the most popular tasks performed by these models is pattern recognition. For

Corresponding Author:
J.P.A. Ribeiro
jp.ar@hotmail.com

this purpose, it is given to the network input-output pairs, and then it will try to find a function that correctly approximates the real relations between them. In a simplified way, an ANN can be defined as presented in Eq.(1), assuming *f** as the real relationship between *x* and *y*. The function *f* is defined with the choice of the ANN architecture. The learning process will determine the values of $\boldsymbol{\theta}$ [1].

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x},\boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{x}$ represents the input data, $\hat{\boldsymbol{y}}$ the predicted output data, f the approximate function of *f** and $\boldsymbol{\theta}$ the learning parameters.

The most popular ANN architecture is the feed-forward neural network, also called Multilayer Perceptron (MLP). In this architecture, the information flows in only one direction, from the input layer, passing through the hidden layers and ending in the output layer. Each computational node in an ANN is called a neuron. Fig.1 shows an example of the MLP architecture.

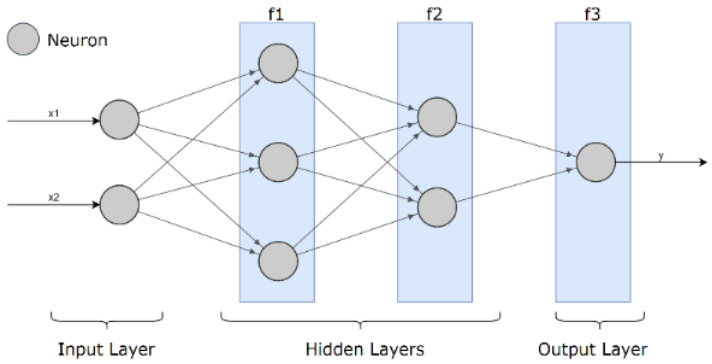


Fig.1 - Representative example of an ANN.

When an ANN has more than one hidden layer, it can be called Deep Neural Network (DNN), since the information is treated hierarchically, in layers with different depths. From Fig.1, one can rewrite the ANN definition, assuming that the *f* function in Eq.(2) is the composition of *n* functions, where *n* represents the number of computing layers in the network [1],[2].

$$\hat{\boldsymbol{y}} = f_3(f_2(f_1(\boldsymbol{x},\theta_1),\theta_2),\theta_3) \tag{2}$$

where f_i and θ_i represent the approximate function and the learning parameters of layer *i*, respectively.

The presented layers in Fig.1 are fully connected, which means that each neuron receives information from all the nodes in the previous layer, and its output is used from every neuron in the next layer.

(a) (b)

Fig.2 – Comparison between natural and artificial neurons: (a) the natural neuron receives information through the dendrites, processes it in the cell body and sends it to adjacent neurons through the axon; (b) The artificial neuron receives information from other neurons (x_i), processes it ($\sum$ and φ) and sends the result (y) to other nodes.

ISEP ACADEMIC PUBLISHING

The artificial neuron structure is an attempt to reproduce the human nerve cell as presented in Fig.2. The neuron receives information, processes it, and then forwards it to adjacent nodes. The process of transmitting information is called a synapse. In the artificial neuron, the information is processed in two stages. The first stage is a weighted sum of all the input data and the second is performed by an activation function, that decides if the neuron is activated or not, *i.e.* if the information will pass to adjacent neurons [3]. The activation function (φ) maps the weighted sum from the]-∞; +∞[interval to the desired domain. One of the most popular activation functions is the Rectified Linear Unit (ReLU), which has an output domain of [0, +∞[, activating the neuron only when the input is greater than 0 [4]. In the learning process, at the neuron level, the learning parameters are the weights from the input connections and the bias, defined as b. The bias is an additional parameter to improve the flexibility of neural networks.

$$y = ReLU(w1 \cdot x1 + w2 \cdot x2 + w3 \cdot x3 + b) \tag{3}$$

where x_i and w_i are the input data from the neuron i of the previous layer and its associated weight, b is the bias and y represents the processed information. The *ReLU* function is defined in Eq.(4).

$$ReLU(x) = \max(0, x) = \begin{cases} 0, & if\ x < 0 \\ x, & if\ x \geq 0 \end{cases} \tag{4}$$

The learning process is a loop, starting with all the weights set with random values, being adjusted in every iteration. Each iteration has two stages: the forward propagation, where the input data (x) is used to predict the output value ($\hat{y}$); and the weights update, using the back-propagation algorithm to calculate the error gradient for each layer, propagating it from the output to the other layers [5]. This process is determined by two main elements: the loss function and the optimizer. The loss function defines a measurement of the network success in its task during the learning iterations. It uses the results predicted ($\hat{y}$) by the network in each step and the real values (y) to calculate the error. One of the most used loss functions in regression models is the Mean Square Error (MSE) defined in equation Eq.(5).

$$MSE(\boldsymbol{y},\hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

where n is the number of data samples. The main goal is to minimize the loss function. Using the error gradient, the ANN must converge to its minimum. The optimizer defines the way that the weights in the network are updated based on the gradient calculated with the back-propagation algorithm. The optimizers implement some variant of stochastic gradient descent (SGD). Some of the most popular optimizers for regression problems are *RMSProp* and *Adam* [1],[2].

3. Plate with Central Hole

The case study is a classic example of structural mechanics, namely a plate with a central hole subjected to uniaxial remote stresses [6],[7]. The plate variables are length plate L, width W, thickness t and the hole diameter, which is characterized by d, as seen in Fig.3. Other case studies can be found in [8].

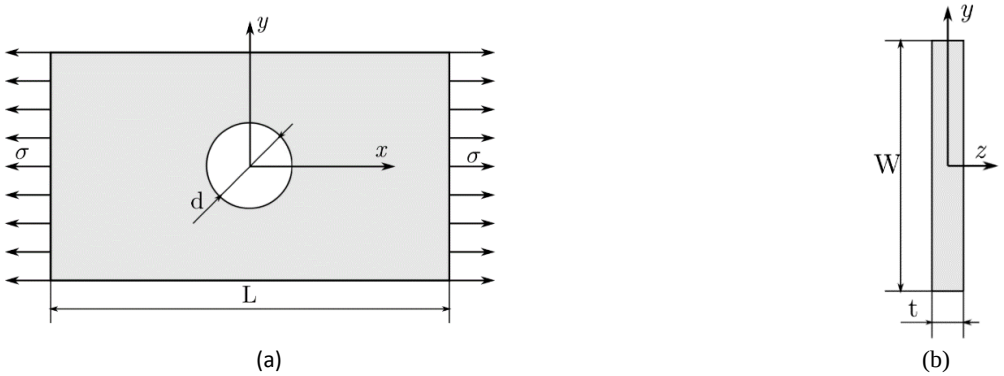


Fig.3 - Plate with central hole: (a) front view; (b) side view.

Thickness is much smaller than the other dimensions. Thus, a 2D analysis is performed, assuming a plane stress state case. Furthermore, the plate is symmetrical about the axes, so the problem can be simplified by analysing only a quarter of the plate, where *l* is half of the length, *w* half of the width and *r* the plate hole radius.

3.1. Problem Definition

Finite Element Software such as Ansys Mechanical calculates the stress/strain distributions efficiently, even in complex structures. However, this method requires large computational resources, and the time for generation of a regular finite element mesh and the time required for matrix calculation may be too long. Furthermore, whenever a boundary condition or material property changes, it is necessary to simulate again. All of this leads to a high computational cost, both in terms of necessary resources and in terms of time spent on mechanical design activities. With the advances in deep learning algorithms, it is possible to perform the same type of analysis of stress distribution using ANN. This new approach reduces the computational cost because the process is faster, introducing only an approximation error [9]. Thus, this research intends to determine the stress field using ANN. Table 1 shows the main properties of the problem, namely the geometric properties, the material properties and the request load value applied at the ends of the plate.

Table 1 - Problem properties: plate with central hole.

Variables	*l* (mm)	*w* (mm)	*r* (mm)	*E* (GPa)	*v*	σ (MPa)
	250	125	25/50	210	0.3	100

4. Methodology

Firstly, it is evaluated what it is intended to be predicted (output variable) and which variables affect the result (input variables). Then, it is created a database, where a set of values of the input and output variables are collected for different cases. These data are obtained with the Ansys Mechanical software. Finally, the entire structure of the ANN is built and applied to the case under study, where the Python language and the Keras library are used. As the material and geometric properties and the loading stress were considered constants, and only the value of *r* is variable. The variation of the stress field between each simulation is characterized by the radius variation. In addition, a stress field is formed by a group of normal stresses calculated on a set of points. Thus, the normal stress depends on the position of the plate point, which it is characterized by the respective Cartesian coordinates. Therefore, the ANN output variable is the normal stress and the input variables are the *x* and *y* coordinates and the hole radius value, as shown in Fig.4.

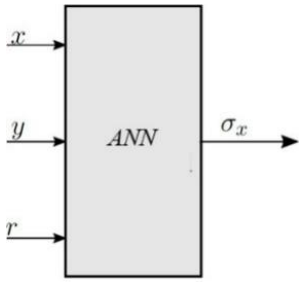


Fig.4 – ANN scheme.

4.1. Database

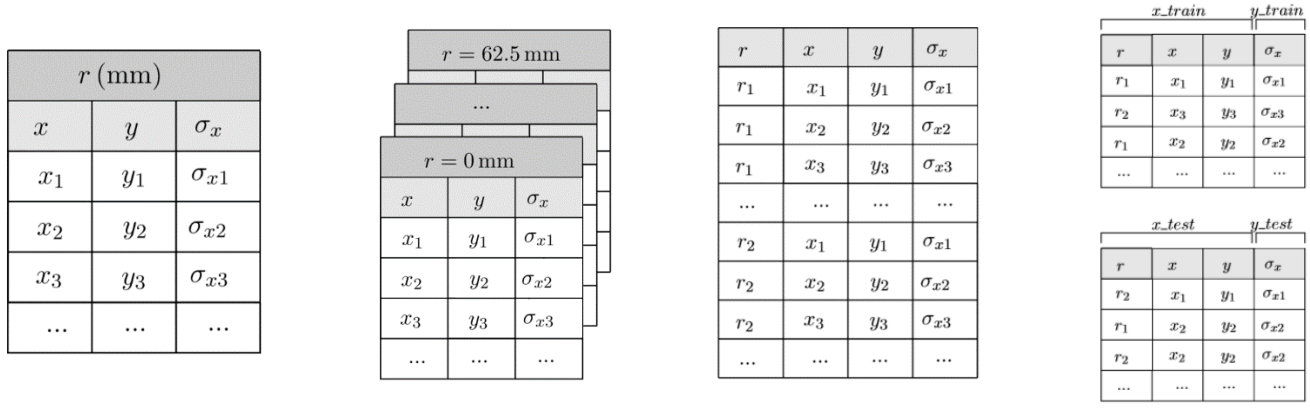
The database is formed by a set of CSV files (Comma-Separated Values), which present the *x* and *y* coordinates of the nodes that make up the finite element model mesh and the respective normal stress values, calculated in Ansys on those same nodes, for a given *r*, as shown in Fig.5 (a). Although quadratic quadrilaterals elements are used, only the values of calculated stresses in the corner nodes were registered in the database. The radius of the plate hole are between 0.1 to 62.5 mm, where the increment value between each simulation is 1.25 mm, except for the first one. Thus, 51 simulations are performed in Ansys, to obtain the data associated with each value of *r*, as shown in Fig.5 (b).

The database is built in two main steps. The first step is the automatic extraction of the stress values for a given value of *r*, using an APDL Command (Ansys Parametric Design Language). The second step, on the other hand, allows the set of simulations necessary for this analysis to be carried out automatically, and it is not necessary to simulate individuality whenever the value of *r* is changed. This procedure is performed through scripting.

4.2. Data Analysis

An initial data analysis was performed to better evaluate the problem. All the data from the different CSV files was gathered, Fig.5 (b), in a single file, Fig.5 (c). This procedure was performed with the support of the Python Pandas library.

With the structured data, box diagrams for each variable are represented and evaluated, Fig.6. This type of representation allows a better understanding of how the data is distributed.



(a) (b) (c) (d)

Fig.5 – Database: (a) for a given value of *r*; (b) for the 51 values of *r*; (c) all data; (d) data splitting.

4.3. Data Pre-processing

Data pre-processing is required to homogenize the data, increasing the effectiveness of the ANN. For the case under study, since that all input data are similar in size, it is used the original data. In addition, it is necessary to divide the data into two groups: training data and testing data. Data splitting is done through Python's Scikit-learn library. In Fig.5 (d) it is graphically represented this division. This division is performed through the *train_test_split* function that divides the data in a random way. For the case under study, 80% of the data corresponds to the training data and the remaining 20% to the testing data.

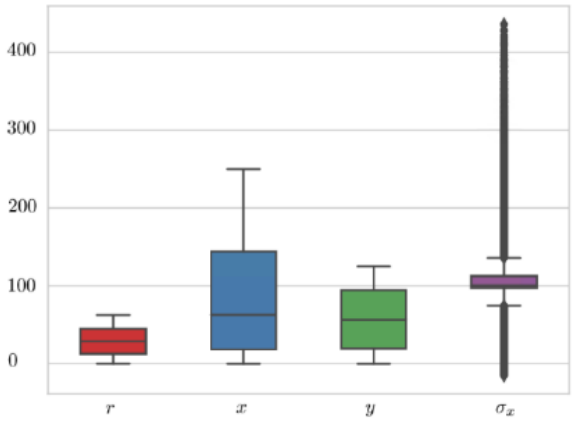


Fig.6 - Box diagrams: *r*, *x*, *y* and σ_x.

4.4. Model Construction

The construction of the ANN model is performed using the Sequential model. The model is formed by six fully connected layers and an output layer that allows obtaining single continuous values. In all layers, except for the last one, the activation function is *ReLU*. In the first layer 128 neurons are used and presented three input variables in vector form. Then, two more layers are used with 128 neurons, one with 64, one with 32 and another with 16. Finally, there is a layer with only 1 neuron that corresponds to the output variable. Fig.7 shows the graphical representation of the ANN, obtained through the ANN Visualizer library.

After built the model, the training configuration is defined. For this, the compile method is used. The optimizer chosen is *RMSProp*. The evaluation metrics are the Mean Absolute Error (MAE), the Mean Square Error (MSE) and the Mean Absolute Percentage Error (MAPE). The chosen loss function is MSE.

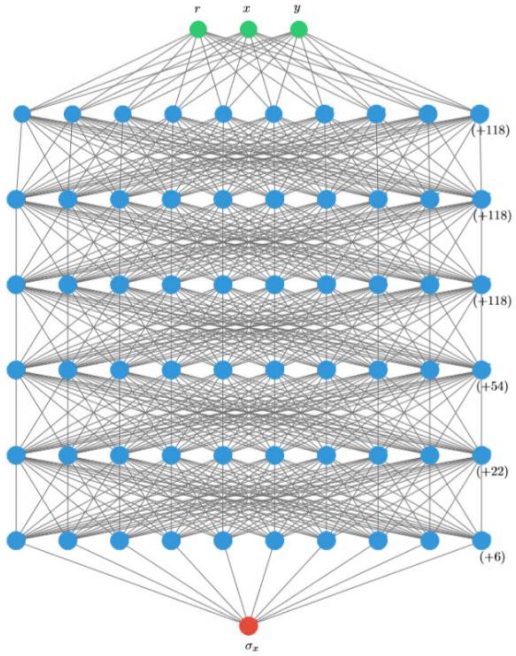


Fig.7 - Representation of the ANN to predict σ_x.

4.5. Train the Model and Evaluation

For model training, the *fit* method is adopted, which allows training the network for a given number of epochs. Note that an epoch corresponds to the time when all training data is processed only once. For the case under study, data stored in *x_train* matrix are used as input data, and data stored in *y_train* as output data. Also, training data is divided using the *validation_split* function, which is used as validation data. For the case under study, 20% of the training data are defined for validation purposes. In addition, it is required to evaluate the number of seasons for model training. In this case, a maximum value of 500 epochs is defined. *Batch_size* is the size of the training sample subset, which will be used during the training process of the network. For the case under study, it is defined a *batch_size* of 20. Finally, callbacks are used to improve the performance of the neural network. Two types of *callbacks* are used: *EarlyStopping* and *TensorBoard*. *EarlyStopping* stops training when a specific monitored metric stops to improve for a specified number of epochs, previously defined with the *patience* argument. As it is intended to decrease the loss function, the metric chosen to be monitored is *val_loss*. Furthermore, 20 epochs are attributed to the *patience* argument. On the other hand, *TensorBoard* is a visualization tool that allows representing and saving the metrics defined according to the epochs. Thus, it is possible to achieve a straightforward visualization of the ANN best configuration.

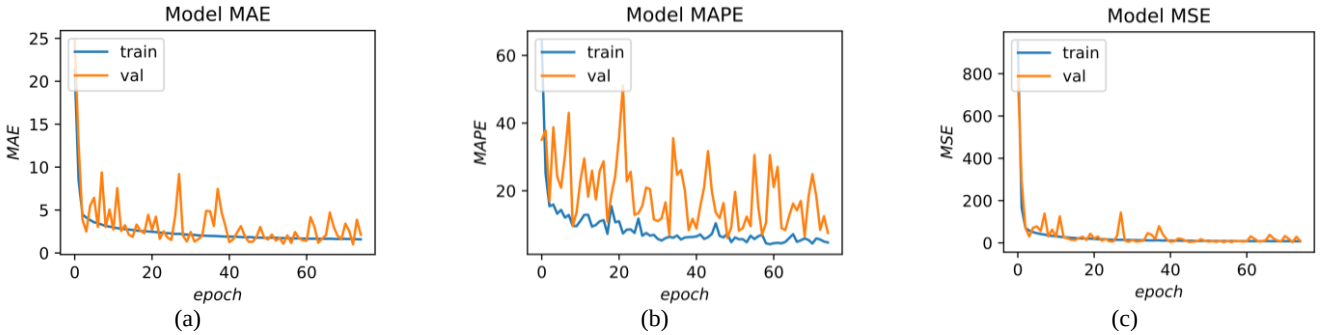
Fig.8 shows the evolution of MAE, MAPE and MSE along the epochs. As noted, the ANN only required 75 epochs to train. Both, for training and validation, the value of the three metrics decreases as the number of epochs increases. Since that the loss function is the same as the MSE metric, it also decreases as the amount of epochs increases.

After the ANN is trained, test data is evaluated, using the *evaluate* method. With this method and for the case under study, the values of the metrics are: *loss = 10.88; mae = 2.18; mse = 10.88; mape = 6.34*. As noticed, the metrics show good results, which means that the network is forecasting correctly.

5. Results

The *predict method* is used for normal stress calculation (output variable), considering the input variables, previously presented. For this case study, it is intended to obtain the stress distribution, σ_x, for a plate with a central hole of radius *r*. Thus, through the ANN it is possible to predict the stress values necessary to build the stress map. This forecast is made for a plate with radius hole between 25 and 50 mm. To obtain the stress field, it is necessary to estimate the stress

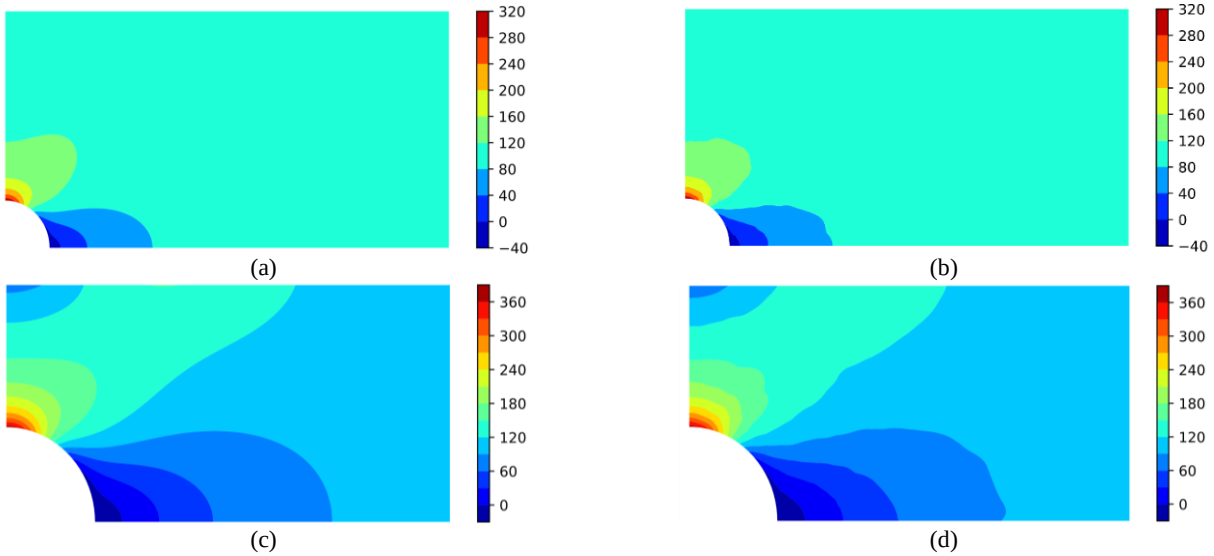
value in a set of points. 873 and 829 points are used for the radius 25 and 50 mm, respectively. The *x* and *y* coordinates are obtained through the CSV file extracted from Ansys Finite Element software.



(a) (b) (c)

Fig.8 – Representation of metrics in function of the epochs (a) MAE; (b) MAPE; (c) MSE.

Fig.9 shows the stress fields predicted by Ansys software and by ANN, for the two cases under study: 25 and 50 mm radius. Note that the visual effect of plate deformation is neglected. Comparing the forecast obtained with the Ansys solution, the Mean Relative Error (MRE) is obtained for both situations under study, Table 2.



(a) (b) (c) (d)

Fig.9 – Graphical comparison of the normal stress fields in the *x* direction of the plate with central hole, in MPa: (a) Ansys, *r*=25 mm; (b) ANN, *r*=25 mm; (c) Ansys, *r*=50 mm; (d) ANN, *r*=50 mm.

Moreover, the computational consumption of the two methods is analysed, namely Ansys and ANN to obtain the necessary data for the characterization of the stress field. For this, a comparison of the time spent in each of the methods to calculate the stresses is established. Note that both situations were performed with the *Intel(R) i7-4710HQ CPU @ 2.50GHz* processor and with *16 GB* of installed memory (RAM). These values are recorded in Table 2, along with the time it takes the ANN to train.

Table 2 - Results overview of the approximation and computational cost.

r (mm)	MRE (%)	Ansys (s)	ANN (s)	Training (min)
25	2.54	3	0.05	3.24
50	6.59	3	0.03	

6. Discussion and Conclusion

As can be seen in Fig.9, the stress maps are very similar, both in terms of the stress values and their distribution over the plate. The major differences lie in the limits between different contour regions of the stress maps, where there is a greater irregularity in the map predicted by the ANN. Also, from Table 2 results it is substantiated that the error is acceptable for both situations, with a MRE of 2.54% for the 25 mm radius and 6.59% for the 50 mm radius.

At the temporal level, compared to Ansys, the ANN is 59 times faster at determining the stress field for *r* equal to 25 mm and almost 100 times faster for *r* equal to 50 mm. For the case under analysis, it takes about 3.24 minutes to train the network. Despite being a high value, when compared to the simulation time, it only must be done once. After the network is trained, it is possible to predict any type of input data very quickly, when compared to a traditional finite element calculation.

Thus, it is concluded that this first ANN implementation is able to correctly and effectively predict the normal stress for a given radius value and for any point on the plate, keeping the remaining properties constant. In this way, computational costs can be significantly reduced, since the ANN can calculate, on average, the stress field 79 times faster than a classical finite element software. The forecast error is considered acceptable, presenting an MRE of 4.57% based on the two cases, where the greatest differences reside within the region's boundaries. The main disadvantages of ANN are the database building process to train the network, and the required time for training. Regarding the database, to improve the process, a script based on APDL command language is adopted. The needed time to train the network could represent a significant computational cost, but it is only necessary to perform the training once.

In conclusion, this work made it possible to highlight the potentialities of ANN in applied to stress/strain calculation in solid mechanics: shorter response time, less computational resources, and problem simplification, in detriment of a lower resolution/accuracy of structural behaviour. Considering these advantages, it is believed that these methods will have a significant potential impact in several applications, particularly in mechanical design. This research work is a first attempt to this topic, and it is recognized that there are many other techniques that could allow the results improvement in a more effective way.

References

[1] Goodfellow I, Bengio Y, Courviele A. Deep Learning (pp. 168-227). The MIT Press; 2016.
[2] Chollet F. Deep Learning with Python (pp. 46-52, 58-60). Manning Publications Co; 2018.
[3] Basheer IA, Hajmeer M. Artificial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods. 2000; 43(1):3-31.
[4] Nair V, Hinton G. Rectified linear units improve restricted Boltzmann machines. In ICML'2010; 2010.
[5] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986; 323(6088), 533-536.
[6] Pilkey WD, Pilkey DF. Peterson's stress Concentration Factors. John Wiley & Sons Inc; 2008.
[7] Timoshenko S, Goodier JN. Theory of Elasticity. McGraw-Hill; 1951.
[8] Ribeiro J, Tavares S, Parente M. Stress–strain evaluation of structural parts using artificial neural networks. Proceedings of the Institution of Mechanical Engineers, Part L: Journal of Materials: Design and Applications; 2021.
[9] Oishi A., Yagawa G. Computational mechanics enhanced by deep learning. Computer Methods in Applied Mechanics and Engineering; 2017.