Available online at www.eccomasproceedia.org
Eccomas Proceedia EUROGEN (2021) 118-131

ECCOMAS
Proceedia

EUROGEN 2021
14th ECCOMAS Thematic Conference on
Evolutionary and Deterministic Methods for Design, Optimization and Control
N. Gauger, K. Giannakoglou, M. Papadrakakis, J. Periaux (eds.)
Streamed from Athens, Greece, 28–30 June 2021

DESIGN OF REINFORCED CONCRETE ISOLATED FOOTINGS UNDER AXIAL LOADING WITH ARTIFICIAL NEURAL NETWORKS

German Solorzano[1] **and Vagelis Plevris**[2]

[1] Department of Civil Engineering and Energy Technology, OsloMet–Oslo Metropolitan University
Pilestredet 35, Oslo 0166, Norway
e-mail: germanso@oslomet.no

[2] Department of Civil and Architectural Engineering, Qatar University
P.O. Box: 2713, Doha, Qatar
e-mail: vplevris@qu.edu.qa

Abstract

In engineering practice, the design of structural elements is a repetitive task that has proven to be difficult to fully automate. This is mainly because of the complex relations of the design variables and the multiple strength and other requirements that must be fulfilled based on code provisions to ensure safety and endurance, usually under extreme loading conditions or harsh environments. An optimal design can be defined as a set of values for the design variables that correspond to the optimal performance of the structural element in terms of a given criterion, usually related to the minimization of cost, while also satisfying all constraints related to strength, serviceability, functionality and safety. Such a design problem can be formally written as a function that maps a structural element, under given loading conditions, into a unique optimal design. In recent years, Artificial Neural Networks (ANN) have been adopted as a powerful strategy to solve complicated regression and classification problems where the underlying mapping function is generally unknown and difficult to formulate analytically. The ANN learns patterns contained in large databases through an automated process called training and uses that information to make highly accurate predictions. In the present study, a methodology that uses ANNs for the optimal design of structural elements is developed and applied to the design of reinforced concrete (RC) isolated footings under axial loading. First, a Genetic Algorithm is employed for the generation of the training dataset for the ANN, which includes RC footing designs that are optimized in terms of the material cost. Then, the ANN is trained and finally asked to produce new optimal designs for new sets of input parameters. Parametric tests are performed to determine the required size of the dataset and the most suitable network architecture. The results show that the accuracy of the prediction is very good, especially when larger datasets are used. It is shown that training an ANN to design structural elements is a viable option that gives acceptable solutions quickly, requiring extremely low computational cost. Furthermore, it is highlighted that good results can be obtained using a simple ANN architecture and a relatively small training dataset.

Keywords: Machine Learning, Structural Design, Structural Optimization, Artificial Neural Networks, Reinforced Concrete, Isolated Footings.

Peer-review under responsibility of the organizing committee of EUROGEN 2021.
doi: 10.7712/140121.7955.18448

1 INTRODUCTION

In engineering design practice, the traditional trial and error approach is still widely adopted and used worldwide. This is because of the difficulties in applying an efficient and effective automated design process, due to the complex relations of the design variables and the multiple strength and other requirements that must be fulfilled based on code provisions to ensure safety and endurance, usually under extreme loading conditions or harsh environments.

In the traditional approach, the dimensions and other important properties of a structural element are proposed based on the empirical experience of the engineer. The proposed design is then subjected to a rigorous check, usually on the computer, to determine if it fulfills all the strength, safety and other requirements provided in the regional building code. If for any reason the design is not satisfactory, then the designer must propose new dimensions or other new properties and repeat the process. On the other hand, when the design turns to be satisfactory and an acceptable solution has been achieved, perhaps the designer will try to optimize it by trying smaller dimensions or other adjustments, to reduce the cost while also maintaining safety. In either case, most of the times the design process is done manually multiple times, which is inefficient, time consuming, and may not reach the "real" optimal solution. Nevertheless, the traditional methods are quickly changing towards more efficient and robust soft computing strategies [1].

Coello Coello et al. [2] presented a GA-based optimization model for the design of rectangular reinforced concrete beams subject to a specified set of constraints. Moayyeri et al. [3] investigated the cost-based optimum design of reinforced concrete retaining walls considering different methods of bearing capacity computation, using Particle Swarm Optimization (PSO). Solorzano and Plevris [4] employed a genetic algorithm (GA) with a dominance-based tournament selection technique for the design of reinforced concrete rectangular-shaped isolated footings in accordance with the American Concrete Institute ACI 318-19. Rojas et al. [5] studied the optimal design for rectangular isolated footings using the real soil pressure showing that the optimal design is more economical and more precise with respect to the traditional design, because standard design is done by trial and error.

Other than optimization, NN-based techniques have also been proposed for both the analysis and the design of structural elements [6]. Haque [7] investigated the suitability of an Artificial Neural Network for modeling a preliminary design of reinforced concrete beam-column. Moradi and Hariri-Ardebili [8] presented the results of a comprehensive study on different experimental models for steel plate and reinforced concrete shear walls, where they developed a predictive meta-model based on ANN, capable of forecasting the responses for any desired shear wall with good accuracy. Nazir et al. [9] studied the application of ANN as a tool for predicting bearing capacity of spread foundations in cohesionless soils. Afaq et al. [10] employed ANNs and general regression analysis for the prediction of the properties of FRP-confined concrete cylinders. Plevris and Asteris [11] presented a novel method with applying ANNs to approximate the failure surface for brittle materials such as masonry, under biaxial compressive stress.

2 METHODOLOGY

In the present study, the idea of employing ANNs to predict the outcome of the optimal design of structural elements is explored. Specifically, the methodology is applied to the structural design of rectangular-shaped reinforced concrete isolated footings. A large database of optimally-designed isolated footings is created and used to train an ANN. After the training is completed, the ANN is able to "predict" the design of any footing (withing the training domain)

fast, with relatively low computational effort. A scheme of the proposed implementation is given in Figure 1.

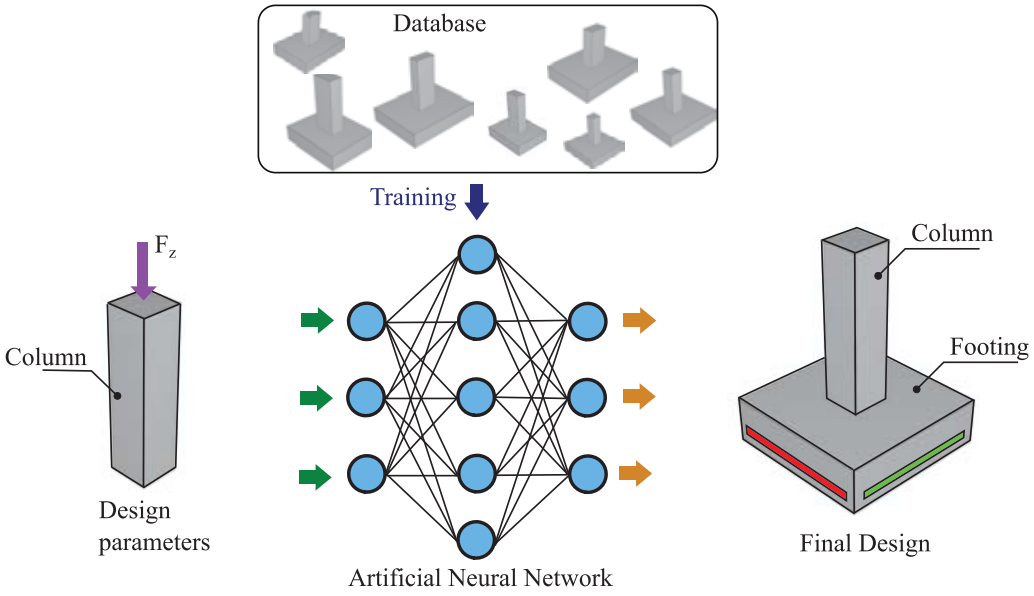


Figure 1: Implementation scheme.

The database required for training the network is generated using a Genetic Algorithm (GA). The design procedure and the application of the optimization algorithm to obtain optimal design results have been the subject of another study by the authors [4]. Now the same procedure is implemented for the generation of the dataset, based on design results obtained using the optimization procedure. After the ANN is trained, it is able to provide new design results, for different input parameters. The primary focus of the study is to investigate the accuracy of the ANN predictions, i.e. the quality of the final ANN-based design results and whether they are suitable for real-world practical applications. This is investigated together with the needed size of the training database to obtain these results, and the most suitable network architecture. The code has been written in Python where the Tensorflow and Keras frameworks are employed to implement the described methodology. The methodology has the potential to be used for the design of other types of elements such as walls, columns beams and connections.

2.1 Problem definition

A concrete isolated footing is a structural element that is part of the foundation of a building. These elements are usually built of reinforced concrete and their purpose is to support the columns of a structure and redistribute their weight into a larger area of the supporting soil, therefore, providing a stable base in which the construction can stand firmly. The design of the footing depends on a set of local parameters such as the magnitude of the load that the attached column is carrying, the maximum allowable pressure that the soil can withstand at that particular location and the resistance of the materials used for its construction. For simplicity, in this study we only consider rectangular shaped footings with concentric axial load.

The design problem is to find the proper dimensions of the concrete slab and the amount of steel reinforcement required to safely resist all the internal stresses developed while following the current accepted material strength theories adopted in the regional building design code. In

this study, a set of 5 input and 5 output variables are established to describe the problem, as seen in Figure 2 and Table 1.

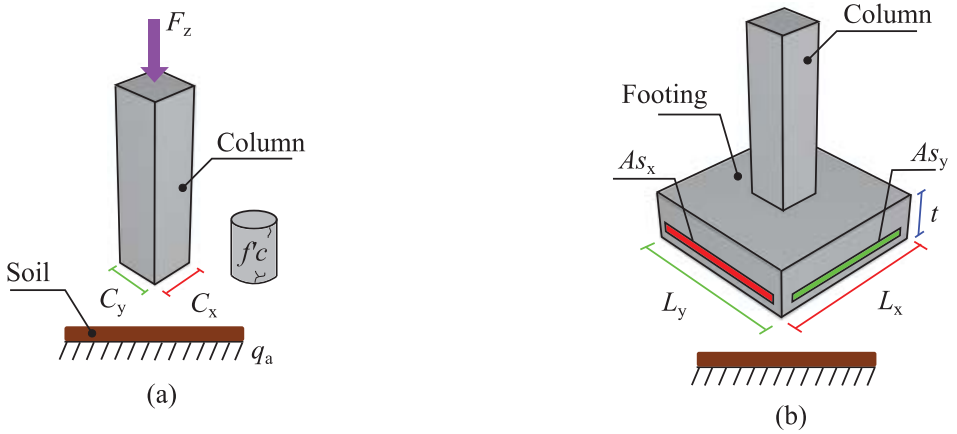


Figure 2: Schematic representation of (a) Inputs and (b) Outputs.

Type	Variable	Description
Input	C_x	Size of the column along the x direction
	C_y	Size of the column along the y direction
	f'_c	Compressive strength of the concrete
	q_a	Allowable pressure of the soil
	F_z	Axial load of the column
Output	L_x	Dimension of the footing along the x direction
	L_y	Dimension of the footing along the y direction
	t	Thickness of the footing
	As_x	Area of steel parallel to the x direction
	As_y	Area of steel parallel to the y direction

Table 1: Description of the inputs and outputs of the problem.

2.2 Creating the Training Database

The databases used to train the ANNs consist of several optimally-designed isolated footings that are created with an optimizer. The optimizer uses a Genetic Algorithm [4] to compute the optimal design of rectangular footings in accordance with the strength requirements and safety guidelines provided by the American Concrete Institute in the ACI 318-19 [12]. A single-objective function is defined where the goal is to minimize the total cost of the footing and an equality constraint enforces the safety and strength requirements provided in the ACI 318-19. A more detailed description of the optimization methodology can be found in a previous publication of the authors [4].

Every footing in the database is denoted as a datapoint or a sample consisting of 5 input values and 5 output values. To generate each sample, the 5 required input values are randomly generated inside a specific domain delimited by an upper and a lower bound. The lower and upper bounds of each design variable are described in Table 2, where P_{max} is the maximum compressive force that the column can withstand according to ACI 318-19. The adopted random sampling technique for the inputs is simple and works well for a small number of design

variables. The efficiency of the methodology can be further improved by using a better and more advanced sampling technique, such as Latin Hypercube Sampling [13] or others.

Variable	Lower Bound	Upper Bound
C_x	40 cm	120 cm
C_y	40 cm	120 cm
f'_c	25 MPa	40 MPa
q_a	200 kPa	1000 kPa
F_z	$0.10 \cdot P_{max}$	$0.95 \cdot P_{max}$

Table 2: Upper and lower bounds of the input variables.

After all the inputs are randomly generated within the design space, the design vector is processed by the optimizer to obtain the optimal design. The optimal design is defined as the 5 output values that produce the footing with the lowest cost while also complying with all the ACI-318 requirements for the given set of 5 input values. The described procedure is repeated until the desired number of samples is obtained. The dataset can be visualized as a table of 10 columns, where each row is a sample (an optimally designed footing). The first 5 columns correspond to the input values and the next 5 columns are the output values.

It is known that the accuracy of ANN predictions in general increases with the size of the training database. One of the aims of the present study is to determine the needed size of the database, i.e. the number of database points required to achieve reliable predictions from the ANN. For that reason, 6 datasets are generated. Each database contains a different number of samples for training, starting from 500, to 1000, 2000, 4000, 8000 and 16,000. Additionally, a dataset of 100 random samples is also generated that will be used for the validation process which will be explained in the following sections.

2.3 Neural Network Architecture

In the recent decades, Artificial Neural Networks have become widely adopted given their ability to solve complex regression and classification problems. The main idea is that an ANN can learn the patterns contained in large databases through an automated process called training [14]. The learned patterns resemble a digital form of "knowledge" which can be used to make highly accurate predictions without the need of a complex mathematical model.

An Artificial Neural Network is a collection of interconnected units referred as artificial neurons that resembles the basic structure of biological neurons that constitute animal and human brains. A biological Neural Network receives an input signal which is processed through the interconnected neurons to produce an output signal. In the same way, an ANN receives some numeric input values that are processed by the artificial neurons to produce a numeric response [15].

There are various types of ANN depending on the type of problem to be solved. For this study, the selected network type is a backpropagation neural network (BPNN). The architecture of BPNNs is relatively simple but still very efficient for regression problems [16]. The BPNN is composed of several layers and each layer has a specific number of neurons. The neurons of the first layer are connected to the ones of the second layer, which in turn are connected to the ones of the third layer, and so on. The very first layer and the last layer are known as input and output layers, respectively, while the other layers are called "hidden layers". The notation used to denote the layer composition is generally as follows:

$$N - H_1 - H_2 - \ldots - H_{L-1} - M,$$

where N denotes the number of neurons in the input layer (number of inputs), H_i stands for the number of neurons in the i-th (hidden) layer, M is the number of neurons in the output layer, and L is the total number of layers, including the output layer but excluding the input layer. For example, a 2-3-3-2 BPNN consists of an input layer with 2 neuron, 2 hidden layers with 3 neurons each, and an output layer with 2 neurons, as seen in Figure 3.

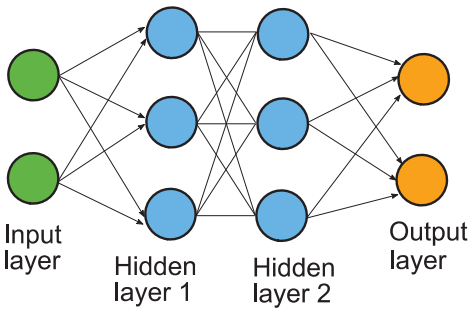


Figure 3: A 2-3-3-2 BPNN.

The mechanism in which the input information is fed to the neural network to obtain the corresponding response is commonly known as feedforward process [17]. It starts by setting some input values into the corresponding neurons at the input layer. Then, moving forward to the first hidden layer, a weighted summation of the neural values from the previous layer with a corresponding connectivity weight is performed. The result of this operation is processed by an activation function to obtain the neural values at the current layer. The procedure is repeated, layer by layer, until the values at the output neurons are obtained. The process is illustrated in Figure 4 to compute the output value at the output neuron labeled as "O" which is connected to 3 neurons from the previous (hidden) layer.

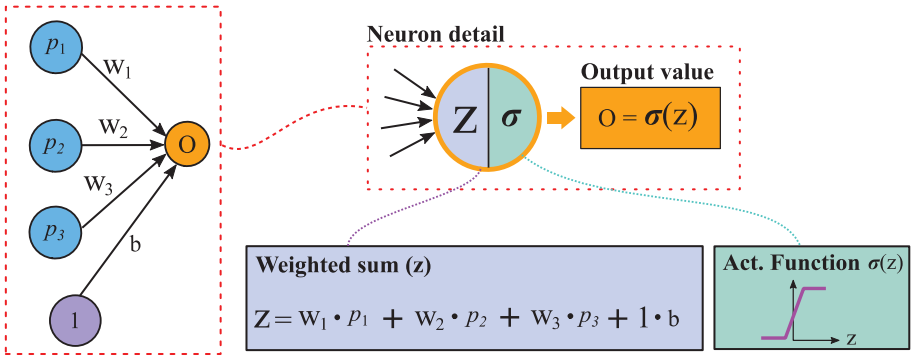


Figure 4: A simple ANN example. The weighted summation is illustrated for the output neuron O.

The operation can be written in matrix form as follows:

$$z = \mathbf{W} \cdot \mathbf{p} + b \tag{1}$$

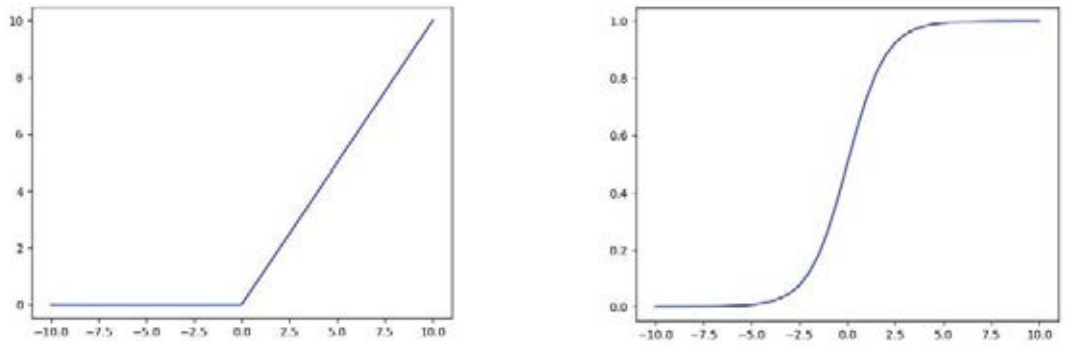
$$\mathbf{W} = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \tag{2}$$

$$\mathbf{p} = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}^T \tag{3}$$

$$O = \sigma(z) = \sigma(\mathbf{W} \cdot \mathbf{p} + b) \tag{4}$$

Where O is the output of the output neuron, $\mathbf{p}$ is a row vector with the values of the neurons at the previous layer; $\mathbf{W}$ is the weight vector that connects this output neuron with the previous layer; and b is a bias neuron that is added to the weighted summation to form the input for the activation function σ.

Activation functions are used to add non-linearity into the neural network and to facilitate the training process in which the gradient of the model parameters must be computed. Additionally, they help to restrict the neuron values to a certain limit providing stability to the network. Therefore, the activation functions are usually simple, continuous, and differentiable. Among the most commonly used activation functions are the Linear, ReLu and Sigmoid functions. Figure 5 presents the Relu and the Sigmoid activation functions.



(a) (b)

Figure 5: ReLu (a) and Sigmoid (b) activation functions, plotted from -10 to 10.

In this study, several different architectures of BPNNs are tested, as shown in Table 3. As the number of layers and neurons increases, so does the computational demand of the training phase. The selected architectures are chosen in an attempt to discover how the number of neurons and layers affects the ANN accuracy and the total training time.

BPNN	No of hidden layers	Architecture
NN1	1	5-24-5
NN2	2	5-12-12-5
NN3	1	5-48-5
NN4	2	5-24-24-5

Table 3: ANN architectures used for training and testing.

2.4 Training Procedure

The training involves a highly iterative process in which the weights of the network (and the biases) are progressively adapted to minimize the error and increase the accuracy of the ANN predictions. The accuracy is measured by comparing the ANN predicted output with the real known output (commonly referred as ground truth). In this study, the mean squared error is used as accuracy metric and is expressed as follows:

$$C = \frac{1}{n}\sum_{i=1}^{n}\left(T_i - p_i\right)^2 \tag{5}$$

where T_i is the known output; p_i is the predicted value; and n is the number of data points. The above equation corresponds to the case with only one output neuron, i.e. a single output value for the network. For multiple neurons in the output layer (i.e. multiple output values), a double summation is needed together with the calculation of the average one time in the end.

Each time that the full dataset values have been fed to the network is called an "epoch". At every epoch the data is randomly subdivided into smaller subsets known as batches [18]. Every batch is then fed to the neural network and the error at the output neurons is obtained with the specified error function. Now, the idea is to determine the gradient or the rate of change of the error function with respect to the model parameters (i.e. the weights). Such operation is not trivial given the complex architecture of the ANN. However, by applying a chain rule the process is greatly simplified using partial derivatives, as follows:

$$\frac{\partial C}{\partial w_L} = \frac{\partial z_L}{\partial w_L}\frac{\partial p_L}{\partial z_L}\frac{\partial C}{\partial p_L} \tag{6}$$

The above equation is defined at one neuron in the output layer L. C is the cost function; w_L denotes the incoming weights from the previous layer; z_L is the weighted summation that becomes the input of the activation function and p_L is the output value at the neuron. The three partial derivatives of Eq. (6) are quite straightforward to compute. For the first term, z_L is a weighted summation, so the derivative with respect to the weights w_L is the output value $p_{L\text{-}1}$ from the previous layer. In the second term, the derivative of p_L with respect to z_L is the derivate of the activation function. And in the third term, the derivative of the Cost function C with respect to the output value is simply $2(T_i - p_i)$ for the mean square error case. Now, using the backpropagation scheme, the computed error is propagated backwards to the previous layers and by applying a similar chain rule, the gradient of all the weights can be obtained. Finally, all the weights are updated by a factor obtained by multiplying their gradients with a scalar value known as the "learning rate". This process of optimizing the model parameters is known as stochastic gradient descent.

The stopping criterion for the training is usually reaching a fixed number of training epochs. Alternatively, a second criterion could be set if no meaningful improvement (reduction of the error) is observed after a predefined number of iterations. Nevertheless, a more appropriate approach is to compare the progress in real-time with a so-called validation set. A validation set is a smaller database that is not used at all during the training phase. At the end of every epoch or iteration, the ANN accuracy is measured for both the training and the validation datasets, independently. If both errors are reducing with every iteration, this generally indicates that the ANN is still "learning" and improving. If on the other hand, the error from the training set is being reduced while the error from the validation set starts to increase, it is an indicator of overtraining or overfitting where the training error is decreased but the network loses its generalization capabilities. An overtrained network with show a very small error in the training set but will fail to predict proper output values for input data that are not part of the training set.

In the present study, the stopping criterion is fixed at a number of 100 epochs while a random subset of 100 samples is used as a validation set to make sure that there is no overtraining.

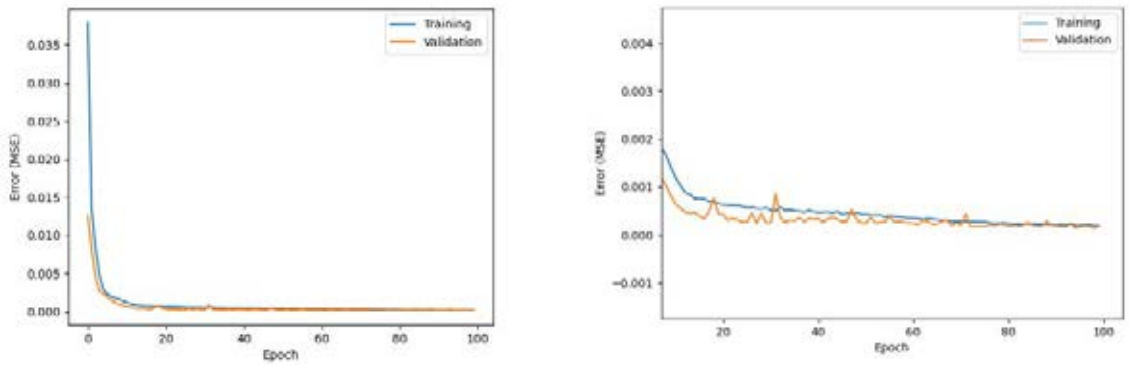
3 RESULTS AND DISCUSSION

3.1 Parameters Summary

A total of 6 databases are randomly generated as inputs containing 500, 1000, 2000, 4000, 8000 and 16,000 data points. Additionally, a set of 100 samples are generated for validation and testing purposes in each case. An optimizer based on Genetic Algorithms is used to generate all the output samples. For details on the optimization procedure, the interested reader is referred to [4] where the methodology is described in detail. Each database is used to train 4 ANNs (NN1, NN2, NN3 and NN4) with different architectures, as shown in Table 3. The ReLu activation function is used in all layers except the output layer where a linear function is used instead. All the ANNs have been trained for 100 epochs as a maximum. The validation database of 100 samples is used to control and avoid overtraining. The batch size is set to 10 and the weight adjustment is conducted using stochastic gradient descent with the backpropagation scheme. All the data inputs and outputs are normalized beforehand. The Tensorflow and Keras frameworks are employed to implement the described methodology in Python. A regular PC with an i7-6700HQ CPU and 16 GB RAM was used to run all the simulations and training operations.

3.2 Training loss

The reduction of the error through the training process (often call training loss) is illustrated in Figure 6. At every epoch the mean squared error of the full training and validation datasets are computed to produce the plot. Only the 5-24-24-5 BPNN (with 2000 data points) is shown in this case as all the other trained ANNs produced similar plots. It can be noticed that no overtraining is happening as the error of both curves shows the same behavior as it goes down with epochs. The training phase takes about 60 seconds to complete using 100 epochs and a batch size of 10.



(a) (b)

Figure 6: (a) Training and validation loss for the BPNN 5-24-24-5 trained with the dataset of 2000 points, (b) The graphic in the right side is just a zoom-in of the graphic on the left.

3.3 ANN performance comparison

After all the ANNs are trained, their performance is compared. For such a comparison, the testing set that contains the 100 randomly generated samples is used. The total mean average error is computed and compared for each of the 4 ANNs that were trained with each of the 6 datasets. Therefore, a total of 24 ANNs are compared in Figure 7. Each curve corresponds to a specific ANN architecture and each point in the curves represents the obtained error (y-axis)

for the corresponding size of the training database (x-axis). Every ANN is run 10 times and the result is averaged to obtain a more reliable measurement given the stochastic nature of the training process.

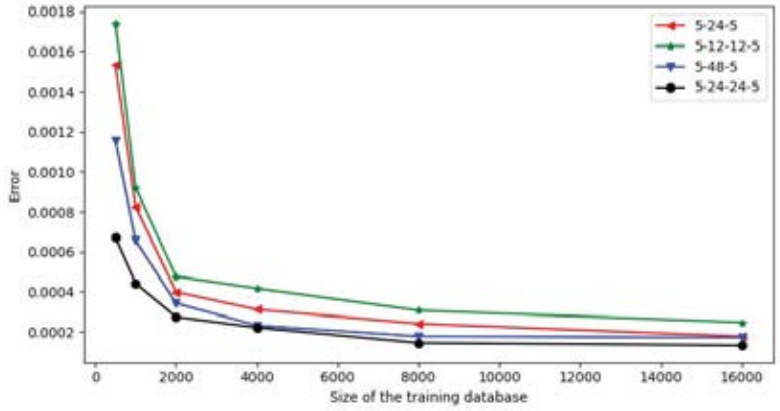


Figure 7: Performance comparison of all the NNs using the testing set.

Regarding the time needed for the training phase, a second comparison is made in Figure 8. Again, each curve corresponds to a specific ANN architecture and each point in the curves represents the elapsed time in seconds (y-axis) for the corresponding size of the training database (x-axis).

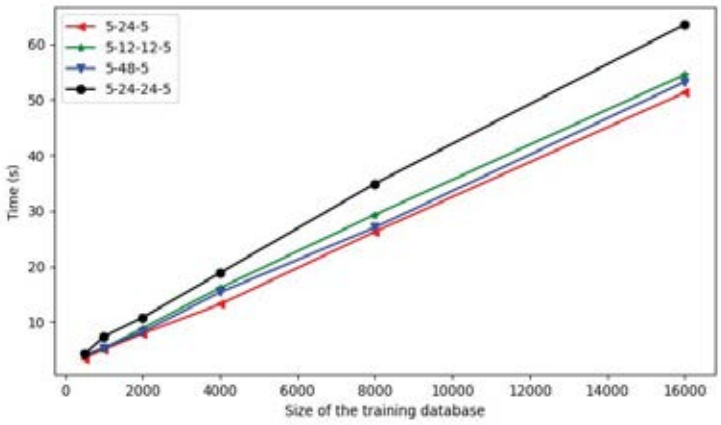


Figure 8: Comparison of the training time needed for all the NNs for 50 epochs.

By analyzing Figure 7, the results seem to be consistent with what was expected from the tests. For all the 4 different ANNs, the error obtained for the testing set decreases as the number of datapoints used in the training increases. Additionally, the 5-24-24-5 BPNN seems to better predict the results by achieving the lowest error among all the tested ANNs. However, the 5-24-24-5 is also the most computationally expensive ANN of them all. By inspecting Figure 8, it can be noted that for the same number of total neurons, using two hidden layers achieves slightly better predictions but is more computationally expensive. Therefore, it seems that it must be a decision from the data scientist to determine the appropriate architecture based on the problem at hand. For this study, the 5-24-24-5 network is recommended given that the increase in training time is relatively low compared to the other ANNs.

The most significant improvement is observed when the training dataset is increased from 500 datapoints to 2000. In this region, all the error curves show an exponential decrease of the error. From 2000 to 8000, the change appears to be linear with a 1/9 slope and from 8000 to 16000, the change shows minimal improvement. Therefore, it appears that a database of 2000 datapoints or higher would be the best option to achieve reliable predictions. However, it is difficult to accurately interpret the meaning of these error quantities. In other words, it is hard to determine if the predictions from an ANN are practically useful just by reading the obtained errors from these curves. Therefore, an exercise is conducted in which the design of a footing will be predicted by the 5-24-24-5 BPNN trained with all the 6 datasets (500, 1000, 2000, 4000, 8000 and 16000). In this way, we look at the numbers from an engineering point of view instead of just looking at the error values. The datapoint to predict is the #99 which is taken randomly from the testing set. The corresponding inputs and outputs are shown in Table 4 and Figure 9.

Input Values		**Output Values (from GA optimizer)**	
C_x	67.64 cm	L_x	361.49 cm
C_y	91.37 cm	L_y	385.84 cm
f'_c	27.26 MPa	t	82.74 cm
q_a	571.48 kPa	As_x	336.03 cm^2
F_z	7484.33 kN	As_y	356.01 cm^2

Table 4: Inputs and outputs of the testing sample #99.

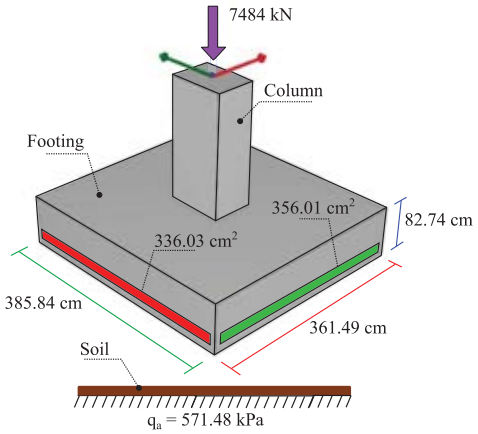


Figure 9: Graphical representation of sample #99 from the testing set.

The ANN-predicted values are shown in Table 5. The first column displays the ground truth computed with the optimizer (also shown in Table 4). Each of the other 6 double-columns corresponds to the results obtained with the 5-24-24-5 BPNN trained with the dataset displayed in the header of the column (i.e. 500, 1000, 2000 etc.). The sub-column **P.** shows the predicted value while the sub-column **R.E.** shows the relative error with respect to the ground truth. The last row of the table is an average of all the relative errors.

Ground	500		1000		2000		4000		8000		16,000	
Truth	P.	R.E.	P.	R.E.	P.	R.E.	P.	R.E.	P.	R.E.	P.	R.E.
L_x=361.49	362.00	0.14%	362.80	0.36%	362.69	0.33%	361.97	0.13%	360.96	0.15%	360.93	0.15%
L_y=385.84	384.47	0.36%	383.29	0.66%	387.57	0.45%	388.66	0.73%	382.88	0.77%	382.82	0.78%
t=82.74	83.95	1.47%	83.75	1.22%	82.26	0.58%	83.27	0.64%	83.09	0.42%	83.47	0.88%
As_x=336.03	380.97	13.38%	367.70	9.43%	314.65	6.36%	345.05	2.68%	346.09	2.99%	337.08	0.31%
As_y=356.01	401.68	12.83%	395.23	11.02%	339.69	4.58%	350.66	1.50%	368.67	3.56%	359.80	1.07%
Average		**5.63%**		**4.54%**		**2.46%**		**1.14%**		**1.58%**		**0.64%**

Table 5: Predictions of the testing sample #99 using the 5-24-24-5 BPNN trained with the 6 different databases;

By inspecting the predictions in Table 5, it can be noticed that the average error is gradually reducing from a 5.63% (with 500 data points) to 0.64% (when using 16,000 data points). However, from an engineering point of view, both solutions are not too far from each other. Comparing the two cases, the first two output values related with the geometrical dimensions of the footing differ by no more than 1.7 cm, while the difference in the third, the thickness, is only 0.5 cm. The other two output values, related to the steel reinforcement area, are different by roughly 40 cm^2. Considering that the area of steel is distributed along the whole length of the element, such a difference may not be critical and the solution using the small 500-points database could be easily considered for preliminary, pre-design purposes.

4 CONCLUSIONS

A methodology that uses a BPNN to perform the design of reinforced concrete isolated footings has been described and implemented. The training datasets have been generated using a Genetic Algorithm that optimizes the footing design in terms of the material cost while considering the strength requirements and dispositions of the ACI 318-19. It was found that using an architecture of 5-24-24-5 with a training dataset of 500 random samples achieves predictions with a relative average error of 6%, which from an engineering point of view, could be considered as "good enough" to be used, at last for pre-design purposes. Additionally, as expected, by increasing the size of the training dataset the error decreases significantly and the predictions are far more accurate, reaching a relative error of 1.14% when trained with 4000 random samples. Such results are almost as good as the ground truth and could be considered as the final design. Therefore, the methodology proves to be a simple and powerful strategy for the automated design of concrete footings or other structural elements, provided that the training datasets are easy to obtain or generate.

4.1 Future work

Based on the findings of this study, we can identify many interesting directions for further research, to further extend or improve the methodology. For example, in the present study only concentrically axial loaded isolated footings have been considered for simplicity, but the method could also be applied to solve more complex scenarios such as eccentrically axial loaded columns and the inclusion of moments acting directly on the column.

One more interesting application would be to adapt the methodology to predict the design of other structural elements such as walls, columns and connections. By doing so, one could obtain a reliable method that predicts the design of a whole building in a few seconds right after the analysis which would improve the pre-design and design process of a building significantly. Although it is possible that for different structural elements and more complex loading scenarios, the proposed ANN architecture with the given parameters and size of the training set

presented in this study may not work as good, this study still provides a good starting point for such implementations.

REFERENCES

[1] Plevris, V. and G. Tsiatas, *Computational Structural Engineering: Past Achievements and Future Challenges.* Frontiers in Built Environment, 2018. **4**(21): p. 1-5 DOI: 10.3389/fbuil.2018.00021.

[2] Coello Coello, C.A., A.D. Christiansen, and F.S. Hernández, *A simple genetic algorithm for the design of reinforced concrete beams.* Engineering with Computers, 1997. **13**(4): p. 185-196 DOI: 10.1007/BF01200046.

[3] Moayyeri, N., S. Gharehbaghi, and V. Plevris, *Cost-Based Optimum Design of Reinforced Concrete Retaining Walls Considering Different Methods of Bearing Capacity Computation.* Mathematics, 2019. **7**(12): p. 1-21.

[4] Solorzano, G. and V. Plevris, *Optimum Design of RC Footings with Genetic Algorithms According to ACI 318-19.* Buildings, 2020. **10**(6): p. 1-17 DOI: 10.3390/buildings10060110.

[5] Luévanos-Rojas, A., S. López-Chavarría, and M. Medina-Elizondo, *Optimal design for rectangular isolated footings using the real soil pressure.* Ingeniería e Investigación, 2017. **37**(2): p. 25–33 DOI: 10.15446/ing.investig.v37n2.61447.

[6] Hajela, P. and L. Berke, *Neural networks in structural analysis and design: An overview.* Computing Systems in Engineering, 1992. **3**(1): p. 525-538 DOI: 10.1016/0956-0521(92)90138-9.

[7] Haque, M., *An Artificial Neural Network Model For Preliminary Design Of Reinforced Concrete Beam Column*, in *2002 ASEE Annual Conference*. 2002, ASEE Conferences: Montreal, Canada. p. 7.164.1 - 7.164.8. DOI: 10.18260/1-2--10929.

[8] Moradi, M.J. and M.A. Hariri-Ardebili, *Developing a Library of Shear Walls Database and the Neural Network Based Predictive Meta-Model.* Applied Sciences, 2019. **9**(12): p. 2562.

[9] Nazir, R., et al., *An Artificial Neural Network Approach for Prediction of Bearing Capacity of Spread Foundations in Sand.* Jurnal Teknologi, 2015. **72**(3): p. 9-14 DOI: 10.11113/jt.v72.4004.

[10] Ahmad, A., V. Plevris, and Q.-u.-Z. Khan, *Prediction of Properties of FRP-Confined Concrete Cylinders Based on Artificial Neural Networks.* Crystals, 2020. **10**(9): p. 1-22 DOI: 10.3390/cryst10090811.

[11] Plevris, V. and P.G. Asteris, *Modeling of Masonry Failure Surface under Biaxial Compressive Stress Using Neural Networks.* Construction and Building Materials, 2014. **55**: p. 447-461 DOI: 10.1016/j.conbuildmat.2014.01.041.

[12] ACI Committee 318, *Building Code Requirements for Structural Concrete (ACI 318-05) and Commentary*. 2005, American Concrete Institute.

[13] Florian, A., *An efficient sampling scheme: Updated Latin Hypercube Sampling.* Probabilistic Engineering Mechanics, 1992. **7**(2): p. 123-130.

[14] Ghaboussi, J., *Advances in Neural Networks in Computational Mechanics and Engineering*, in *Advances of Soft Computing in Engineering*, Z. Waszczyszyn, Editor. 2010, Springer Vienna: Vienna. p. 191-236. DOI: 10.1007/978-3-211-99768-0_4.

[15] Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning*. 2016: MIT Press, ISBN: 9780262035613.

[16] Buscema, M., *Back Propagation Neural Networks.* Substance Use & Misuse, 1998. **33**(2): p. 233-270 DOI: 10.3109/10826089809115863.
[17] Rosenblatt, F., *The perceptron: A probabilistic model for information storage and organization in the brain.* Psychological Review, 1958. **65**(6): p. 386–408 DOI: 10.1037/h0042519.
[18] Keskar, N.S., et al., *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*, in *5th International Conference on Learning Representations (ICLR 2017)*. 2017: Toulon, France.