Research Article

Framework for the development of artificial neural networks for predicting the load carrying capacity of RC members

Afaq Ahmad[1] · Demitrios M. Cotsovos[2] · Nikos D. Lagaros[3]

Received: 14 August 2019 / Accepted: 26 February 2020 / Published online: 3 March 2020

Abstract
This paper aims at establishing a framework for the development of artificial neural networks (ANNs) capable of realistically predicting the load-carrying capacity of reinforced concrete (RC) members. Multilayer back propagation neural networks are developed through the use of MATLAB and enriched databases which contain information describing the variation of load-carrying capacity in relation to key design parameters associated with the RC specimens (i.e. beams) considered. This work forms the basis for the development of a knowledge-based structural analysis tool capable of predicting RC structural response. A detailed discussion is provided on the different aspects of the proposed framework which include (1) the formation and analysis of the relevant (experimental) data, (2) the architecture of the ANNs, (3) the training/calibration process they undergo and finally, (4) ways of extending their applicability enabling them to predict the behaviour of RC structural forms with design parameters not represented in the available experimental database. Non-linear finite element analysis is used for validating the predictions of the ANN models developed. The comparative study reveals that the ANN models developed through the proposed framework are capable of effectively predicting the load-carrying capacity s of the RC structural forms considered quickly, accurately and without requiring significant computational resources.

Keywords Artificial neural network · Database · Sampling method · Ultimate limit state · Reinforced concrete · Training process · Finite element analysis · Failure · Latin hypercube sampling

List of symbols

α_v	Shear span
b	Width of the beam specimen cross-section
d	Effective depth of the beam specimen cross-section
A_s	Area of longitudinal reinforcement acting in tension
A_{sw}	Area of transverse reinforcement
α_v/d	Shear span to depth ratio
f_c	Uniaxial compressive strength of concrete
f_{yl}	Yield stress of longitudinal reinforcement bars
f_{yw}	Yield stress of transverse reinforcement bars
s	Spacing between shear links
ρ_l	Ratio of tensile reinforcement ($\rho_l = A_s/b \cdot d$)
ρ_w	Ratio of transverse reinforcement ($\rho_l = A_{sw}/b \cdot s$)
V_u	Shear strength

Abbreviations

CFP	Compressive force path
ANN	Artificial neural network
ULS	Ultimate limit state
LHS	Latin hypercube sampling

✉ Afaq Ahmad, afaq.ahmad@uettaxila.edu.pk; Demitrios M. Cotsovos, D.Cotsovos@hw.ac.uk; Nikos D. Lagaros, nlagaros@central.ntua.gr | [1]University of Engineering and Technology, Taxila, Taxila, Pakistan. [2]Heriot-Watt University, Edinburgh EH14 4AS, UK. [3]National Technical University of Athens, Athens, Greece.

SN Applied Sciences (2020) 2:545 | https://doi.org/10.1007/s42452-020-2353-8

1 Introduction

Progress in structural analysis is achieved through the formulation of new analysis techniques capable of providing more accurate solutions to increasingly complex problems without requiring additional computational resources. This can be achieved through the use of Soft Computing (SC) methods. Such methods initially appeared three decades ago as a new family of computational algorithms based on heuristic approaches, which do not strictly adhere to the principles of theoretical mechanics, unlike the traditional analysis procedures, [1–3]. Although initially treated with suspicion, they are presently employed to form surprisingly powerful computation tools, the applicability of which is constantly being extended to different fields of engineering [1–3]. Artificial neural networks (ANNs), genetic algorithms and fuzzy logic are SC methods widely employed [4]. In recent years ANNs have emerged as a powerful tool capable of providing accurate solutions to a range of problems without requiring the high computational resources and lengthy analysis time required compared by the more traditional numerical procedures employed in structural analysis based on the finite element (FE) method.

ANNs are increasingly being used for the structural assessment of simple RC configurations (i.e. beams, bridge decks and columns) [5–8]. Although, ANNs have also been employed to predict the response of more intricate RC structural forms (e.g. RC frames) [9–11] with the whole structure modelled as a single ANN, the latter approach lacks generality as it is case-dependent and its application to other problems requires re-training of the ANN employed. In order to successfully formulate a scheme suitable for the analysis of any RC structure, it is imperative that the ANNs employed are capable of realistically predicting the nonlinear behaviour of the individual members (e.g. beams, columns, slab and walls) comprising the structure considered. To accomplish this, a good understanding of the mechanics underlying RC structural response is essential, since, although ANNs depend on heuristic approaches rather than on strict mechanics, their calibration process has to realistically account for the most important aspects of the problem at hand (i.e. the effect of key design parameters) that influence the exhibited response.

Information concerning RC structural behaviour is usually obtained from tests carried out on simple structural configurations as well as the use of nonlinear finite element analysis and the available assessment methods [12–14]. The specimens studied experimentally are usually scaled models of structural members comprising actual RC structures. The available test data collected [15] is processed and used to form databases which describe the variation of specific aspects of specimen behaviour (e.g. the load-carrying capacity) in respect to key design parameters. However, the range of values of the design parameters considered in the databases are not always representative of their counterparts adopted in actual (full-scale) RC members. Therefore, the ANNs developed purely on the basis of the available test data are not guaranteed to provide accurate predictions for the behaviour of the specimens considered or of full-scale structural members. Nonlinear finite element analysis (NLFEA) is frequently employed for investigating the behaviour of RC structural elements and the numerical predictions obtained provide information that complements the available test data. However, these predictions are often dependent on parameters essential for fully defining the NLFEA models developed. The values of such parameters may vary depending on the characteristics of the problem considered and often require recalibration. This affects the generality and objectivity of the numerical predictions obtained. Furthermore, NLFEA requires significant computational resources and time when conducting detailed parametric studies in order to establish the effect of specific design parameters on specimen behaviour. Predictions concerning RC structural response can also be obtained from physical models adopted by the available RC design codes [12, 13] and alternative assessment methods [14]. Such models offer an interpretation of the available test data and describe the physical condition of the structural elements at the ultimate limit state (ULS). As a result, the predictions obtained are based on specific assumptions concerning the mechanics underlying RC structural response.

The present study aims at establishing a framework for the development of ANNs capable of realistically predicting certain important aspects of the behaviour of RC structural members (e.g. the load-carrying capacity and the exhibited mode of failure) when approaching ULS which accounts for both the limitations and benefits associated with the available sources of information concerning RC structural response. This work forms the basis for the development of a knowledge-based structural analysis tool for predicting RC structural response. Multilayer Back Propagation (MBP) Neural Networks are used in combination with MATLAB [16, 17] to form an open source analysis tool which permits the user to change the parameters of the problem considered thus allowing more flexibility for solving a wider range of engineering problems. The most important aspects of the proposed framework include: (1) the analysis of the relevant test data and the formation of databases upon which the development of the ANN is based, (2) the architecture of the ANN, (3) the training/

calibration process that the ANN undergoes and finally (4) methods of extending the applicability of the ANNs to predict the behaviour of RC structural forms with design parameters not included (or not well represented) in the available (purely experimental) databases. For the comparative the comparison of ANN models with the empirical equations, author already published this in [18].

The proposed framework is presently implemented in detail for the case of simply supported RC beams with and without transverse reinforcement. The limitations of the existing experimental databases and their impact on the generality of the predictions obtained from the ANNs are discussed in detail. In an attempt to overcome the above shortcomings hybrid databases are formed in which the experimental data is enriched through the use of additional information that can be obtained either from the available structural analysis packages or the available assessment methods. It is demonstrated that the ANNs developed based on the hybrid databases are capable of realistically predicting the behaviour of a wider range of RC structural members characterised by design parameters which are either not well represented or not included in the initial experimental database. Finally, NLFEA is also employed in combination with the available test data to validate the predictions of the ANNs. The comparative study reveals that the ANNs are capable of effectively predicting certain aspects of RC structural response accurately, quickly and without requiring high computational resources.

2 Fundamental characteristics of ANNs

ANNs mimic the basic functions of their biological counterparts in the central nervous system and the brain of animals and humans (see Fig. 1a) [19–21]. They are used to estimate or approximate functions that depend on a large number of input parameters, the effect of which is not clearly established or quantified. ANNs have the ability to learn, generalize, categorize and predict values due to their adaptive nature and their ability to "*remember*" information introduced to them during the training/calibration process. They consist of a number of consecutive layers and, in turn, each layer consists of a system of interconnected "neurons". Each link (forming between two neurons) is assigned a specific coefficient (weight) which is multiplied by the values generated by the neurons. The values obtained from all neurons of a specific layer are then transferred through the links and summed with a bias value (see Fig. 1b). The latter sum (analytically expressed by Eq. 2) is then introduced into a predefined activation function (*f*) representing the relationship between the neurons of the consecutive layers. The output values obtained from the activation functions of a specific layer form the input values for the neurons of the next layer (see Fig. 1). The weights are initially randomly assigned to their corresponding links and their final values are obtained through an iterative training (calibration) () process based on the use of the available data included in the relevant databases.

$$\boldsymbol{x_j} = \sum(\boldsymbol{y_k w_{kj}}) + \boldsymbol{b_j} \tag{1}$$

$$\boldsymbol{y_j} = \boldsymbol{f}(\boldsymbol{x_j}) \tag{2}$$

where $\boldsymbol{x_j}$ represents the output from the neural network $\boldsymbol{y_k}$, are the values obtained from the activation function $f(x_j)$, $\boldsymbol{w_{kj}}$ are the weight coefficients, $\boldsymbol{b_j}$ is the bias value and $\boldsymbol{f}$ is the activation, k represents the number of the neuron within a specific layer and j represent the number of the layer.

2.1 Types of activation functions

The activation functions [22] incorporated within the ANNs form links between successive layers of neurones

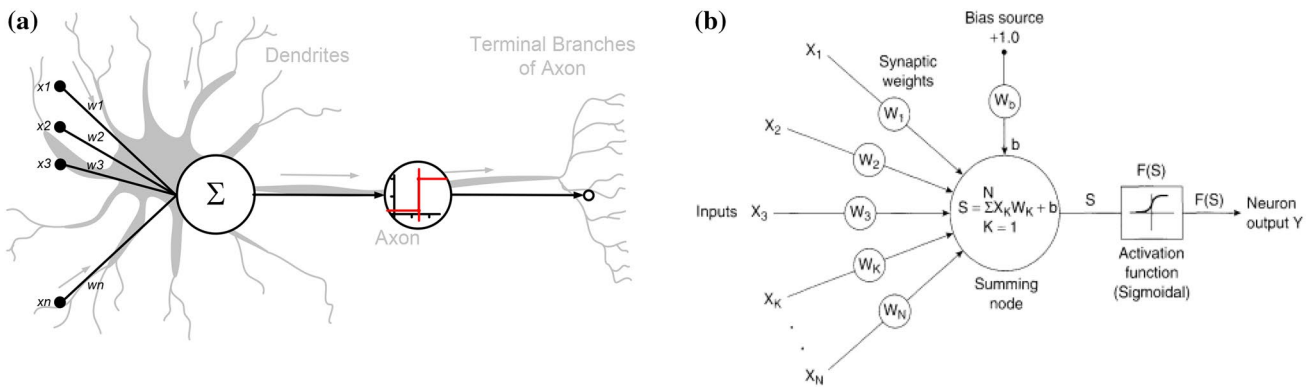


Fig. 1 **a** Analogy between artificial and biological neural networks, **b** Mathematical representation of an artificial neural networks model

transforming the value of the sum provided by Eq. (1) into output values that are fed into the following layer. ANNs employ a range of activation functions depending on the type of problem considered as shown in Table 1. The activation functions can be generally classified as linear or nonlinear. Previous studies show that the type of the activation function employed between different layers depends on how the input parameters are processed and normalised (e.g. the form in which they are introduced into the ANN) which in turn depends on the type of problem considered. If all values associated with the input and output parameters are normalized between 0 and 1, as is often the case for problems associated with identification and regression analysis [22] (then the "*log-sigmoid*" function (see Table 1) is often used for linking the first two layers of the ANN and the "*linear*" activation function (see Table 1) for linking the last two layers (the last hidden layer and the output layer) of the ANN. If the values resulting from the normalization process are positive or negative, ranging between − 1 and 1, as is the case for prediction (e.g. stock exchange and rain fall prediction) and pattern recognition problems [23, 24], then the "*hyperbolic tangent*" function (see Table 1) can be used between all consecutive layers of the ANN. When considering decision making (i.e. stock recommendation and rate handling) problems [25] then the "*Gaussian*" activation function (see Table 1) is often employed for the output layer and the "*hyperbolic tangent*" function (see Table 1) for the hidden layers. In spite the above practices, there are no fixed rules concerning the choice of activation function and the

Table 1 Different activation function for ANN

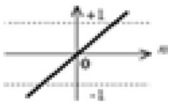
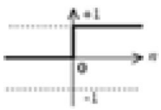
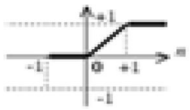
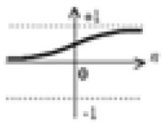
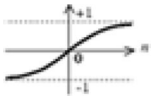
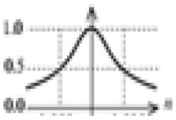
S. no	Activation function	Formulae	Range	Graph
1	Identity (linear)	$f(x) = x$	$(-\infty, +\infty)$	
2	Step (threshold)	$f(x) = 0 \quad if\ 0 > x$ $f(x) = 1 \quad if\ x \geq 0$	[0, + 1]	
3	Piecewise Linear	$f(x) = 0 \quad if\ x \leq x_{min}$ $f(x) = mx + b \quad if\ x_{max} > x > x_{min}$ $f(x) = 1 \quad if\ x \geq x_{max}$	[0, + 1]	
4	Sigmoid (Logistic)	$f(x) = \frac{1}{1-e^{-x}}$	(0, + 1)	
5	Hyperbolic	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	(− 1, + 1)	
6	Gaussian	$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{a}$ $a = \frac{-(x-\mu)^2}{2\sigma^2}$	(0, + 1)	

Table 2 Statistics analysis for BWOS

	Units	Min	Max	Mean	SD	COV
b	mm	50	1000	197.23	128.3	0.66
d	mm	65	2000	320.42	209.34	0.66
a_v/d		0.38	11.42	3.57	1.58	0.45
ρ_l	%	0.14	7.46	2.09	1.11	0.54
f_y	MPa	35	1779	423.83	158.27	0.38
f_c	MPa	12.2	110.9	38.71	21.62	0.56
V_u	KN	7	683	87.66	77.34	0.89

SD standard deviation, *COV* Coefficient of variance

method in which the data should be analysed and processed when developing ANNs.

2.2 Architecture of multi-layered ANN

Multilayer back propagation (MBP) artificial neural networks (ANNs) have been widely used for solving structural engineering problems [21, 26–33]. The structure of the subject ANNs consists of at least three successive layers: an input layer, one or more hidden layers and an output layer (as shown in Fig. 2).

The function of the MBP process consists of two phases: (1) the free-forward (*input signal*) phase and (2) the back-propagation (*error signal*) phase. In the case of the free-forward phase, the information is processed from the input layer towards the output layer. During this phase, the input values are initially introduced in the neurons of the first (**input**) layer of the ANN and the information generated is then process through the consecutive hidden layers. The last of the hidden layers is connected to the **output** layer (see Fig. 2) the latter providing the predicted solution to the problem considered.

As already discussed, the values introduced into the neurons of the **input** layer represent information usually included in the relevant database. These values are multiplied with the corresponding weights (assigned to the associated links) and their sum is finally added to the bias value (see Eq. 1). The resulting value is then introduced into the activation functions (see Eq. 2) which generate the input values of the neurons of the following layer. This process is repeated for every consecutive layer and at the end of the free forward phase, the values provided by the output neurons represent the predicted solution provided by the ANN for the problem considered. These predictions are then compared against their corresponding target values included in the relevant database. The error (obtained from Eq. 3) is then established expressing the level of difference between the predicted and target values.

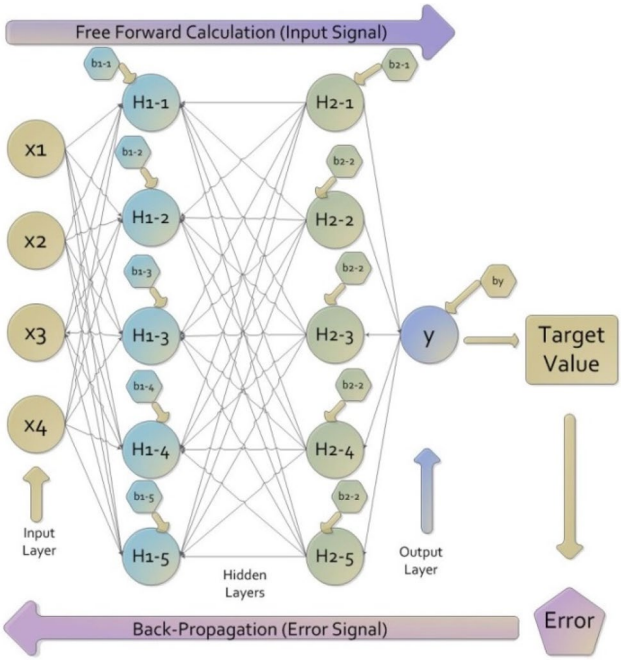


Fig. 2 The free forward and back propagation phases in an ANN

$$E = \frac{1}{2}\sum(t_j - y_j)^2 \tag{3}$$

The performance of the ANN (i.e. its ability to provide accurate predictions) is assessed based on the level of error calculated from Eq. 3. This error can be minimized by adjusting (calibrating) the weights and bias values of the ANN model. This is achieved through the back-propagation phase of the MBP ANN. The term "*back-propagation*" refers to the direction in which the data flows during this phase. Information flows backwards, from the output layer through the consecutive hidden layer(s) towards the input layer. The aim of this process is to correct/adjust the values of the weights and biases so that the output values in order for the output values obtained from the ANN to converge to the target values provided by the relevant database. This process is carried out by taking the derivative of the activation function and repeated until the errors become acceptable. This is achieved through the "*gradient descent*" method (see Fig. 3a). During each iteration the change in the value of the weight Δw_{kj} (expressed by Eq. 4) depends on: (1) the type of error function E adopted and (2) the learning rate (α). The values of the weights are corrected/adjusted based on Eq. 4.

$$\Delta w_{kj} = -\alpha\frac{\partial E}{\partial w_{kj}} \tag{4}$$

The use of high values of learning rate (α) suggest that the values of the weights (w_{jk}) change more drastically (large values of Δw_{jk}) during each iteration. Large values of Δw_{jk} can result in the training process not identifying the most optimum combination of values of weights. On the other hand, small values of α result in the values of the weights being adjusted slowly (small values of Δw_{jk}) during each iteration. In this case, although more training time is required (due to the larger number of iterations that have to be carried out to satisfy the convergence criteria), the iterative process is able to identify more effectively the optimum combination of values of the weights allowing the resulting ANN to provide more accurate predictions. However, during this iterative process two main problems should be considered associated with local minima and over fitting [20, 34–36] (see Fig. 3b). The problem of local

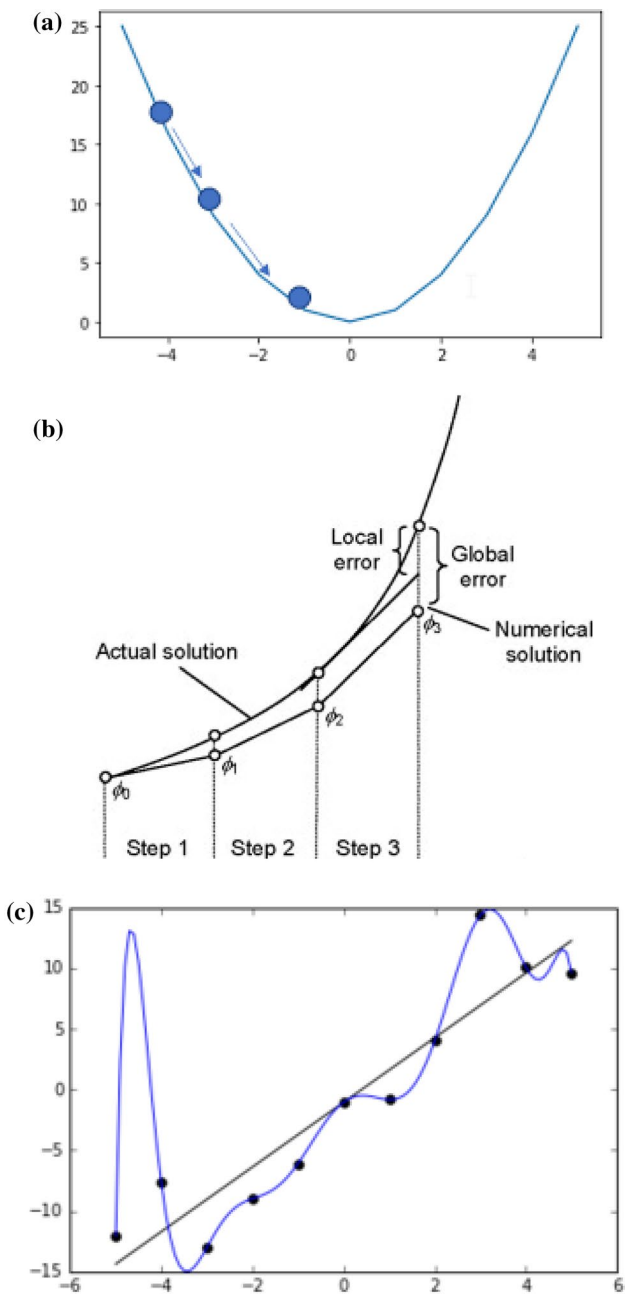


Fig. 3 **a** Gradient descent, **b** local and global error and **c** over fitting problem [19–21]

minima is associated with the convergence of the iterative process to solution which is not representative for the whole database. As a result, the training/calibration process stops early, without processing the whole database and caused the problem of over fitting or over-prediction of the ANN model compared to their counterpart target values. To avoid such problems, during each iteration (n) the correction of the weights Δw_{kj} is provided as a function (see Eq. 5) of the corresponding adjustment carried out in the previous iteration (n – 1) and the momentum factor (η) which obtains values between 0 and 1 (where n refers to the epoch/iteration number). To avoid problems associated with local minima and over-fitting the database is divided into three subsets used for training, validation and testing purposes instead of just two (used for training and testing purposes only) [19–21]. This process is discussed in the following section.

$$\Delta w_{kj}^{n} = \alpha y_k + \eta \Delta w_{kj}^{n-1} \quad (5)$$

3 Developing procedure of ANN

In the present study, MBP ANNs are developed for predicting the load-carrying capacity of RC members at ultimate limit state (ULS), based on the available test data [15]. The flowchart in Fig. 4 provides an outline of the process adopted for the development of the subject ANNs.

The efficiency of the training process depends on how the database is processed and divided into the training, validation and testing subsets, the normalization process adopted for processing the input data, the number of hidden layers included in the ANN, the number of hidden neurons in each layer, the type of activation function adopted between consecutive layers as well as the type of error functions and the training and learning algorithms employed [20]. No fixed rules exist concerning the above parameters. As a result, the development of the ANN is often achieved through a trial and error process. When developing ANN previous relevant studies carried out to date [19–21] suggest that:

- A sufficient number of input samples should be included in the database, so that each subset has an adequate number of samples for training, validation and testing.
- All weight and bias values are either initially randomly assigned values between – 0.5 and 0.5 or are all set to one (1).
- The number of neurons included in the hidden layers should be double the number of the neurons included in the input layer.
- Sigmoid activation functions are normally used between the first two layers (input and the first hidden layer) of the ANN whereas the hyperbolic tangent activation function is used in the output layer.

3.1 Enrichment of existing experimental databases

As already discussed, the data included in the existing purely experimental databases are usually obtained from tests carried out on scaled specimens. As a result, the full

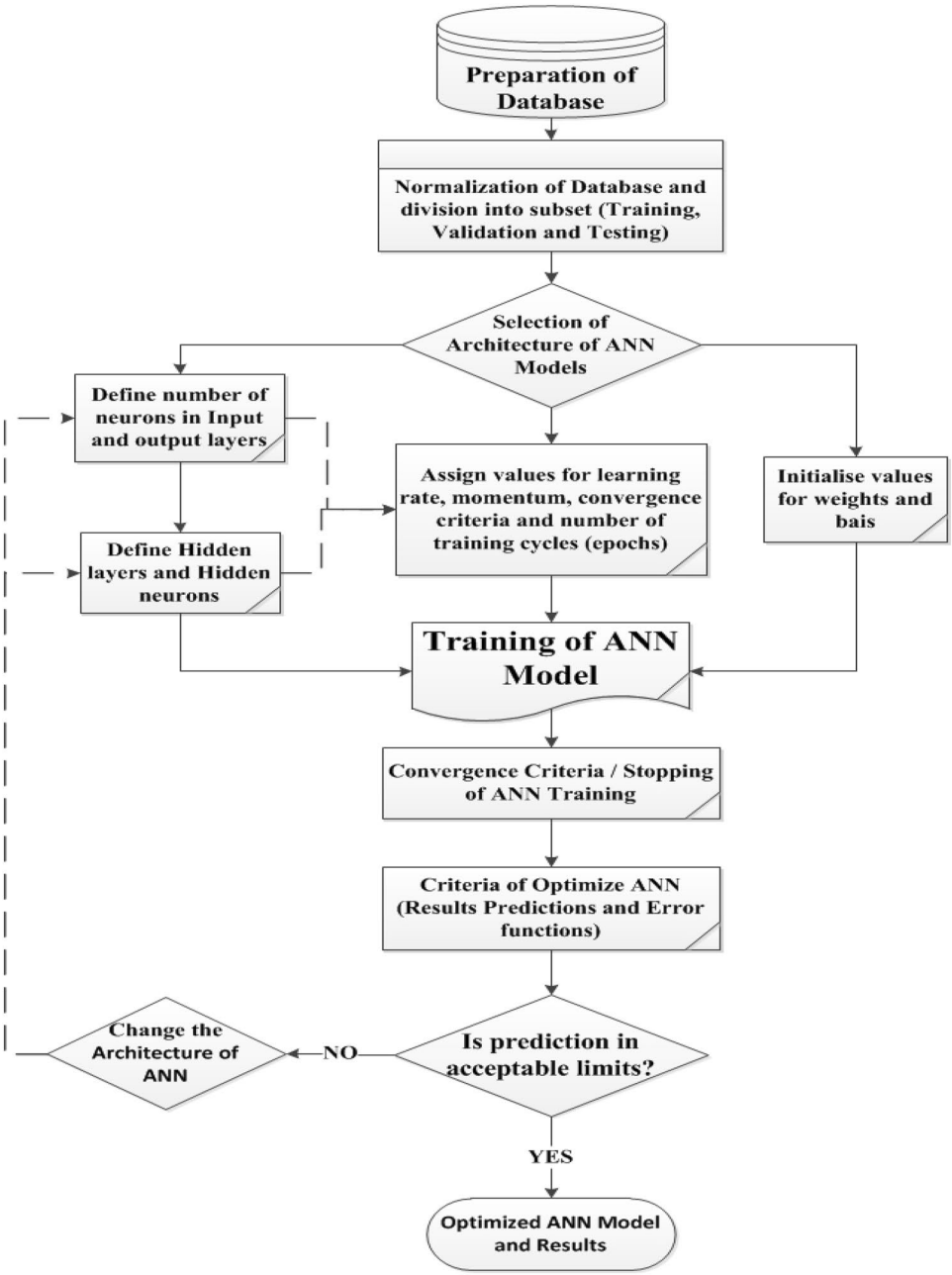


Fig. 4 Developing process for ANN model

range of values that the design parameters can potentially take in practice may not be fully represented by the number of samples included in the database. Furthermore, the values included in the databases may not be representative of their counterparts adopted for full-scale RC members. As a consequence, the applicability of the ANNs developed on the basis of the available test data is limited. Therefore, it is often necessary to enrich the existing experimental databases so that the full range of values of the design parameters are considered allowing for the development of ANN capable of providing accurate predictions for a wider range of cases. The enrichment process can be achieved by introducing more samples into the available experimental databases by (a) carrying out additional tests, (b) employing Artificial Generating Techniques (i.e. a form of interpolation method for creating additional samples on the basis of the existing ones), (c) using the available assessment methods and their underlying physical models and (d) conducting detailed parametric numerical studies through the use of NLFEA. However, it is important to mention that each of these methods are characterised by advantages and disadvantages with respect to the cost, computational resources required, the objectivity

and generality as well as the accuracy of the predictions obtained.

A large number of experiments have been conducted to investigate the behaviour of simply supported RC beam specimens without stirrups in order to assess the effect of various design parameters on certain aspects of the exhibited behaviour (as shown in Fig. 5). Data selected from 608 RC beams without stirrups subjected to 3 or 4 point bending tests (describing the effect of various design parameters on the load-carrying capacity and mode of failure of the specimens) has resulted in the formation of a database [15]. Close consideration of the data included in the database reveals some regions that are poorly populated (i.e. regions associated with certain values of the design parameters for which a large number of samples is not available). Figures 6 and 7 shows the distribution of the available data in respect to the range of values associated with certain key design parameters ($b, d, a_v/d, fc, fy, \rho_l = A_s/bh$) considered in the database revealing regions which are densely populated and others which are not. The range of values associated with each design parameter considered in the database can be divided into: (1) a strong range (densely-populated region), which has a sufficient number of samples to accurately describe the effect of the design parameters considered on the load-carrying capacity of the beams and (2) a weak range (poorly/sparsely-populated region) which contains little (or even no) samples. For example, for 50 mm < b < 200 mm the database is densely populated (including 550 samples out of the 604 samples) representing a strong range. However, for 200 mm < b < 510 mm the database includes a low number of samples forming a weak range. Similar observations can be made for the other key design parameters considered in the database (see Figs. 6 and 7).

Prior to enriching the experimental database, it is essential to identify the regions which are poorly populated and find the required number of additional samples that are missing. To achieve this the Latin hypercube sampling (LHS) method [37] (i.e. randomization of data sampling) was used in this study. The LHS method is employed to find the number of additional samples required for the case of each parameter considered (see Figs. 8 and 9). A relevant assessment method is then employed to generate the required number of samples in order to enrich the database enabling it to account for the effect of the subject parameter on the load-carrying

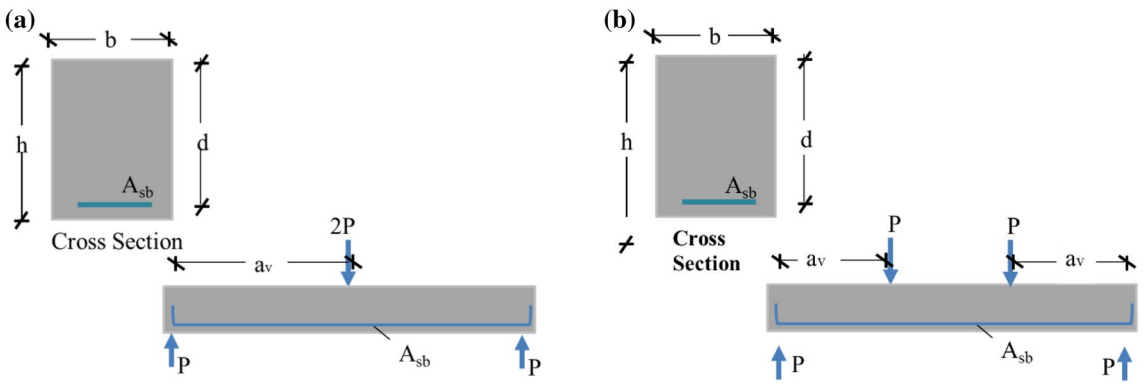


Fig. 5 Experimental setup for **a** Three-point and **b** four-point flexural tests for RC beams (BWOS)

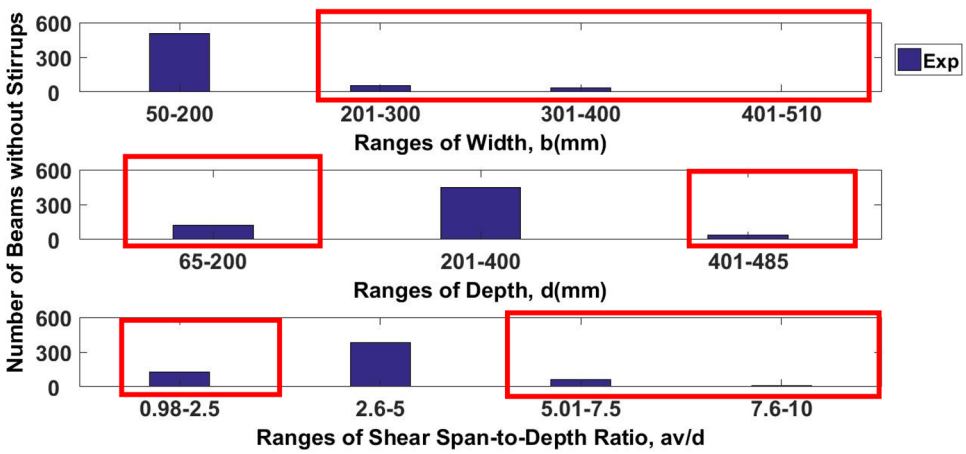


Fig. 6 Distribution of samples within database BWOS for different values of cross sectional parameters associated with the width, depth and shear span to depth ratio of the beam specimens

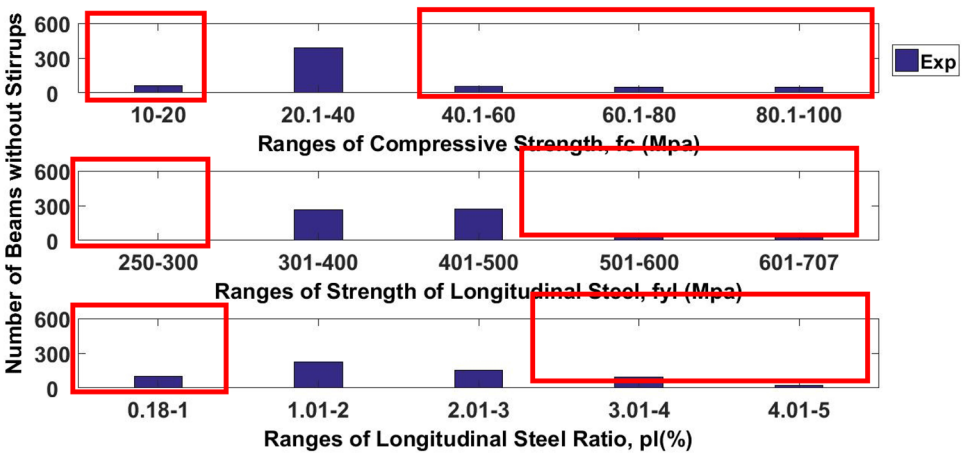


Fig. 7 Distribution of samples within database BWOS for different values of material parameters associated with the compressive strength, longitudinal strength of the steel and the longitudinal steel ratio of the beam specimens

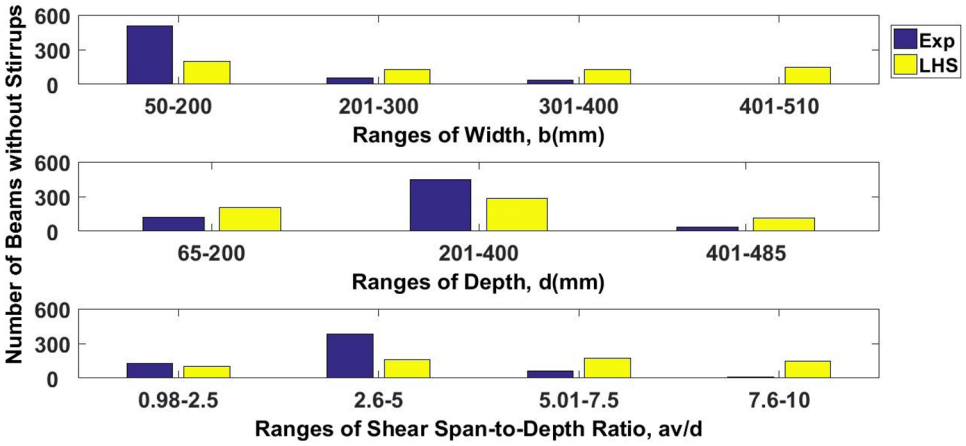


Fig. 8 Number of samples of the experimental database BWOS and its counterpart developed through LHS when considering design parameters associated with the geometry of specimen

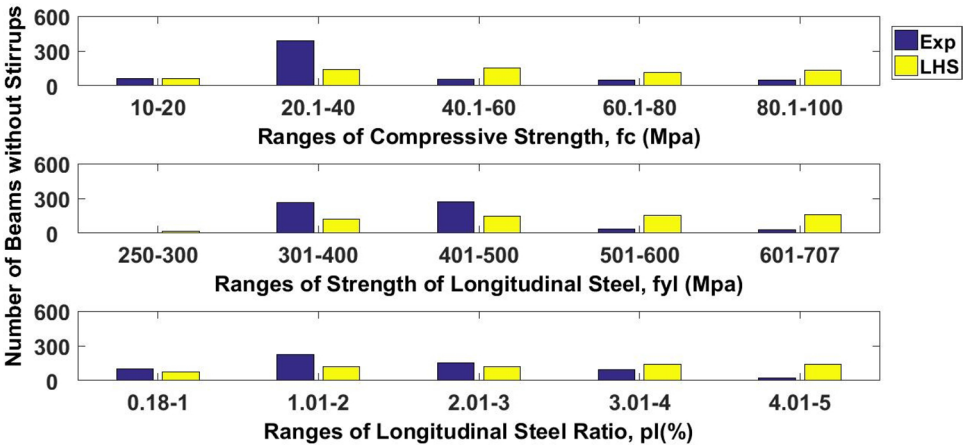


Fig. 9 Number of samples of the experimental database BWOS and its counterpart developed through LHS when considering design parameters associated with material properties and longitudinal reinforcement

capacity. For the present case study, the Compressive Force Path (CFP) [15] method is employed for generating the required number of samples. This process is repeated for all the critical parameters. By combining the predictions obtained from the CFP method, and the available tests data, an enriched (artificial) uniform and well-populated database is formed.

To solve this problem, an effort is made in this investigation. The CFP method, which shows the good level of accuracy [15], with the experimental values, an artificial database has been created. For which the recognition of the critical parameter is important because the ranges are set for this critical parameter only. The critical parameter is that which has a most effective property with

respect to the shear strength. For this, all the remaining parameters are in the ratio of this parameter. So, that with uniform interval values for the critical parameter, the well distributed database can be generated. However, for the critical parameter and the ratio, the problem cannot be solved by the previous design codes [12, 13] and NLFEA [38].

Author previous work [15] shows the alternative physical model i.e. CFP give the better predictions against the current design codes and closer to experimental values. To overcome the above mentioned problem CFP method is used to developed new database only targeting the poorly populated regions (CFP). This new database is then merged with the previously developed experimental database and knows as Hybrid Experimental database (HEXP). Figures 10 and 11 described that the HEXP has sufficient number of samples throughout the region of different parameters for the case of BWOS. This HEXP database is then used to train the ANN model for RC BWOS. For the validation purpose of the CFP method, limited case studies are carried out in NLFEA (i.e. ABAQUS) for the mentioned case.

3.2 Normalization of database

The training process of the ANNs becomes more efficient when the values (both input and target data) included in the database is normalized and at the end of the training process, all the outputs can de-normalized for the comparative studies. The performance of ANN depends upon the quality of the database. The normalization of the parameters considered in the database has significant impact on the ANN procedure. Considering that the various input combinations are associated with different units the normalization process allows their conversion to unit-less parameters. To avoid problems associated with low learning rates of the ANN [39, 40] it is better to normalize the values of the parameters between an appropriate upper and lower limit value of the subject parameter. In this work, instead of using MATLAB automated functions of normalization, the normalization is done by own. This is because to control the whole process of ANN by user instead of by code. All parameters associated with the 608 beams without stirrups are normalised between [0.1, 0.9] instead of [0, 1], by using following Eq. 6.

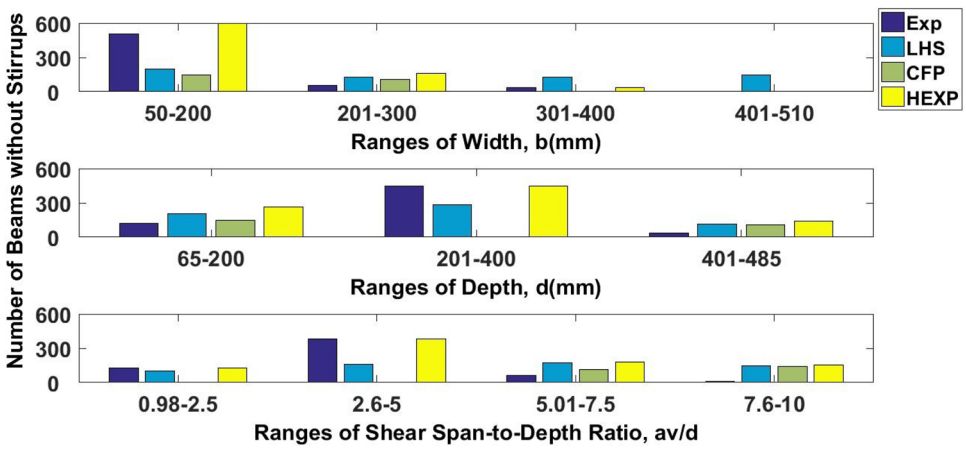


Fig. 10 Number of samples of experimental database BWOS and its counterparts developed through LHS and the CFP method when considering design parameters associated with the geometry of specimens

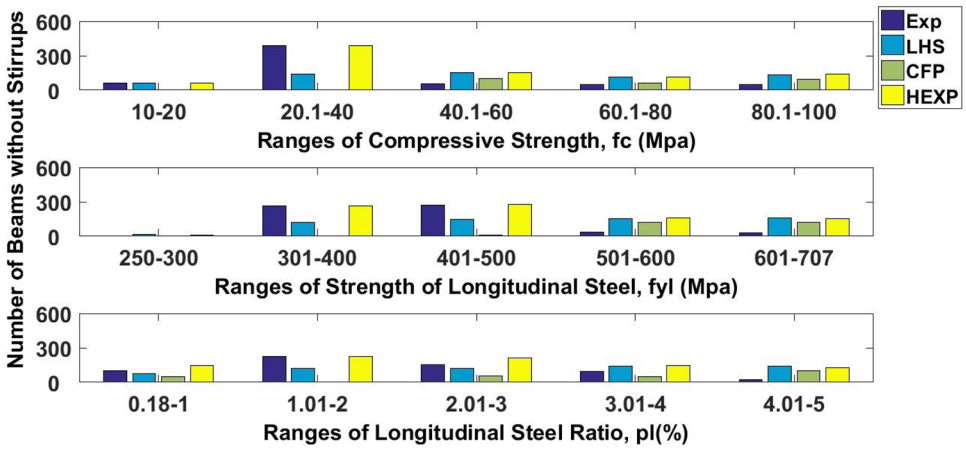


Fig. 11 Number of samples of experimental database BWOS and its counterpart developed through LHS and CFP of design parameters associated with the Material Strength and Steel Ratio of specimen Table 2 provides the statistical information concerning the variation of certain parameters associated with the specimens considered

$$\mathbf{X} = \left(\frac{0.8}{\mathbf{x}_{max} - \mathbf{x}_{min}}\right)\mathbf{x} + (0.9 - \left(\frac{0.8}{\mathbf{x}_{max} - \mathbf{x}_{min}}\right)\mathbf{x}_{max} \quad (6)$$

3.3 Architecture of ANN models

MATLAB [16, 17] is used for the development of the ANNs and Table 3 shows the different parts of MATLAB code.

3.4 Division of database into training, validation & testing sub-sets

To improve ANN Generalization and avoid over fitting, it is very useful to divide the database into further three sub sets i.e. training, validation and testing subsets (**TVT**). The training subset is used for computing the gradient and updating the network weights and biases. The *validation* subset which is used to monitor the error during the training process. The value of the latter error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to over-fit the data, the value of the error calculated from the validation subset typically begins to rise. The network weights and bias values are saved once the minimum value of the error calculated on the basis of the validation subset is attained. The testing subset is used to compare different models. It is also useful to monitor the error calculated on the basis of the test subset during the training process since if it attains its minimum at a significantly different iteration number than the error calculated based on the validation subset, this can potentially indicate a poor division of the database. In MATLAB, four functions provided for dividing the database into the training, validation and test subsets which are shown in Table 4 with their advantages and disadvantages. In the present work the "dividerand" and "divideind" functions are sued to compare their effect on the performance of ANN [16, 17].

3.5 Training & learning algorithms for ANN

As described earlier, the learning process of the back propagation (BP) involves sending the input values forward through the network and then computing the difference between the calculated output and the corresponding desired output from the training dataset. In this algorithm, the value of the error function is minimized using a gradient-descent technique. The necessary corrections to the weights and bias values of the network for each moment are obtained by calculating the partial derivative of the error function in relation to each weight. The BP which based on gradient- descent or Jacobian method, is derived using the chain rule of calculus. The BP learning has become the standard method and process in adjusting weights and biases for training an ANN in many engineering applications. However, BP (gradient-descent) has three limitations, (a) the suitable architectures of ANN is not easy to find (b) the generated multifaceted error-planes with multiple local minima, the BP fall into local minima instead of a global minimum [21, 34, 41, 42]. Another problem during BP training, is that each time ANN is trained, can result in a different solution due to different initial weight and bias values and different divisions of data into training, validation, and test sets. Thus, different ANN trained on the same problem can give different outputs for the same input. To ensure that ANN has good degree of accuracy, it should be trained at least three times. While training the MLFFBP, there are 12 training algorithms which based on different defining criteria, in MATLAB. Table 5 provides the information about these 12 training algorithms. Following are some training algorithms based on gradient or Jacobian method, which are available in Neural Network Toolbox software of Matlab [16, 17];

As mentioned earlier, for the robust and efficient training, the second order learning algorithms must be used. For this the effective method is trainlm, Levenberg–Marquardt (LM) algorithm [43–45] which is a derivative of the

Table 3 Different part of the MATLAB code used for ANN

	Database	Hidden layers	activation function	Training & learning algorithms	Error function	User control	Division
net = newff	(Input, Trg,	{H1, H2}	{'logsig','logsig','tansig'},	'trainlm','learngdm'	,'mse'	,{},{}	'dividerand')

Table 4 Four division functions

Functions	Description	Merits/demerits
Dividerand	Divides the data randomly (default)	User has no control
Divideblock	Divides the data into contiguous blocks	User has no control
Divideint	Divide the data using an interleaved selection	User defined
Divideind	Divide the data by index	User defined-Constant

Table 5 Different 12 Training Algorithms [16, 17]

S. no	Notation	Description
1	trainlm	Levenberg–Marquardt
2	trainbr	Bayesian regularization
3	trainbfg	BFGS Quasi-Newton
4	trainrp	Resilient back propagation
5	trainscg	Scaled conjugate gradient
6	traincgb	Conjugate gradient with Powell/Beale restarts
7	traincgf	Fletcher–Powell conjugate gradient
8	traincgp	Polak–Ribiére conjugate gradient
9	trainoss	One step secant
10	traingdx	Variable learning rate gradient descent
11	traingdm	Gradient descent with momentum
12	traingd	Gradient descent

Newton method. The following Table 6 describe the different input parameters for the Levenberg–Marquardt training in Matlab. In this work, different above mentioned 12 training functions are used on the case study. The purpose of doing this is to evaluate the previous guidelines and to experience the different between these training functions. The next sections are about these 12 training functions and their effect on the performance of ANN.

3.6 Criteria for optimized ANN model

The most efficient/optimal ANN model, capable of providing the most accurate predictions, is usually assumed to be the ANN characterised by the highest value of correlation factor (*R*) [46]. In the present study, in addition to the correlation factor (*R*), two more error functions—the Mean Square Error (*MSE*) and the Mean Absolute Error (*MAE*)—are also considered [46]. The most optimized ANN model should be associated with the highest value of *R* and the smallest values of *MSE* and *MAE*. Following are the three different error types are considered (a) correlation factor (R) as explained by Eq. 7, (b) Mean Squared Error (MSE) as in Eq. 8 and (c) Mean Absolute Error (MAE) in Eq. 9 [40, 47–49].

$$R = 1 \Big/ (N-1) \sum_{i=1}^{N} \left[\frac{(Y_i - \bar{Y}_i)}{S_y} \frac{(X_i - \bar{X}_i)}{S_x} \right] \tag{7}$$

$$MSE = \frac{\sum_{i=1}^{N} |Y_i - X_i|^2}{N} \tag{8}$$

$$MAE = \frac{\sum_{i=1}^{N} |Y_i - X_i|}{N} \tag{9}$$

where N = Total Number of samples, X_i = actual values, Y_i = predicted values, $\bar{X}_i = \sum_{i=1}^{N} \frac{(X_i)}{N}$ and $S_x = \sqrt{\frac{(X_i - \bar{X}_i)^2}{N-1}}$, same for the $\bar{Y}_i$ and S_y.

4 Modelling example for ANN

In order to do the investigation on different above-mentioned parameters, different type of investigations is carried out during this work. However, it is not possible to investigate the parameter at the same time. For this, different combinations of parameters are used. Table 7 provides the information about these combinations. Against these developed cases, above considered database is used to establish the guidelines for ANN MATLAB tool.

4.1 IC-1: random division and training functions

For enhance the performance and accurate predictions of the ANN, the above-mentioned normalized data is initially divided into three sub-sets: (training, validation and

Table 6 Parameters for the Levenberg–Marquardt [16, 17]

Parameters	Default value	Description
net.trainParam.epochs	1000	Maximum number of epochs to train
net.trainParam.goal	0	Performance goal
net.trainParam.max_fail	6	Maximum validation failures
net.trainParam.min_grad	1e−7	Minimum performance gradient
net.trainParam.mu	0.001	Initial mu
net.trainParam.mu_dec	0.1	Mu decrease factor
net.trainParam.mu_inc	10	Mu increase factor
net.trainParam.mu_max	1e10	Maximum mu
net.trainParam.show	25	Epochs between displays (NaN for no displays)
net.trainParam.showCommandLine	False	Generate command-line output
net.trainParam.showWindow	True	Show training GUI
net.trainParam.time	Inf	Maximum time to train in seconds

Table 7 Different combination of Investigative Cases (IC)

	Database	Hidden layers	Hidden neurons	A.F	Training & learning algorithms	E.F	Division
net = newff	(Input, Trg,	{H1, H2}	{H1, H2}	{'logsig', 'tan-sig'},	'trainlm', 'learngdm'	–	–
IC-1	Input = 06 Output = 01	Double	12,12	STT	12 training algorithms	MSE, MAE, R	Randomly Division (60%, 20%, 20%)
IC-2	Input = 06 Output = 01	Double	12,12	STT	12 training algorithms	MSE, MAE, R	Constant Division
IC-3	Input = 04-06 Output = 01	Single	06-12, 06-12	ST	'trainlm', 'learngdm'	R	Randomly Division (60%, 20%, 20%)
IC-4	Input = 04-06 Output = 01	Double	06-12, 06-12	STT	'trainlm', 'learngdm'	R	Randomly Division (60%, 20%, 20%)
IC-5	Input = 04-06 Output = 01	Double	2xIN, 2xIN	STT	'trainlm', 'learngdm'	MSE, MAE, R	Randomly Division (60%, 20%, 20%)

testing purposes). Matlab [16, 17] is used to develop the ANN models and to randomly divide the database into three sub-sets: **60%** of each database is used for training, **20%** for validation and another **20%** for testing purposes. Following Fig. 12a, b, show the three error functions (MSE, MAE, and R) against the above mentioned 12 training function by using the Database (i.e. beams without stirrups) with *random division* of data samples. While doing this investigation, following parameters of architecture of ANN is used as constant.

a. The input parameters are six i.e. b, d, a_v/d, ρ_l, f_{yl}, f_c.
b. The number of hidden layers is **two.**
c. The number of hidden neurons in each layer are 12 (twice the number of input neurons).
d. The activation function between first hidden layer and input layer is sigmoid and the two-activation function with next three layers are tanh.

4.2 IC-2: constant division and training functions

After using the *randomly division* data, another investigation is carried out with constant division. Matlab [16, 17] is used to develop the ANN models and to divide the database into constant three sub-sets: the **1:400** samples belongs to the training subset, **401:504** samples belongs for validation and **505:608** samples belongs for testing purposes. Following Fig. 13a, b, show the three error functions (MSE, MAE, and R) against the above mentioned 12 training function by using the Database (i.e. beams without stirrups) with *constant division*.

As from above Figures, in both the case is proved that trainlm (LM) is the most accurate and useful training for engineering applications. This training algorithms also used in preciously conducted studies. However, the performance of *randomly division* is much better than the *constant division*. Hence, the output of this investigation is that to use the randomly division with trainlm training algorithms. From this comparative study following two guidelines are established.

(a) Randomly Division database should be used for subsets (i.e. training, validation and testing).
(b) LM-training algorithms should be used for MBP.

4.3 IC-3: single hidden layers (SHL)

Now in this phase of investigation we change the above constant parameters and use the above output. The above data is divided into three subsets as; **60%** of each database is used for training, **20%** for validation and another **20%** for testing purposes and the training function of train is used. But this time single hidden layer (SHL) and different number of hidden neurons with different activation functions are used. A total of 84 different ANN models were created for the case of the RC beams without stirrups (see Table 8). From each case, the ANN model exhibiting the maximum value of *r* between the output and target values is selected as shown in Table 9.

4.4 IC-4: double hidden layers (DHL)

The same above practice is repeated with double hidden layers (DHL) and different number of hidden neurons with different activation functions are used, as described in Table 10. Table 11 described the R values for all three subsets and overall performance of DHL ANN models.

From both these comparative studies, ANNs exhibiting higher value of *r* are the most optimal. Form Tables 10 and 11, for the case of RC beams without stirrups, **ST-6-12-12-1** (employing a combination of input parameters: *b, d,* a_v*/d,* f_c, ρ_l, f_{yl}) is the most optimum ANN

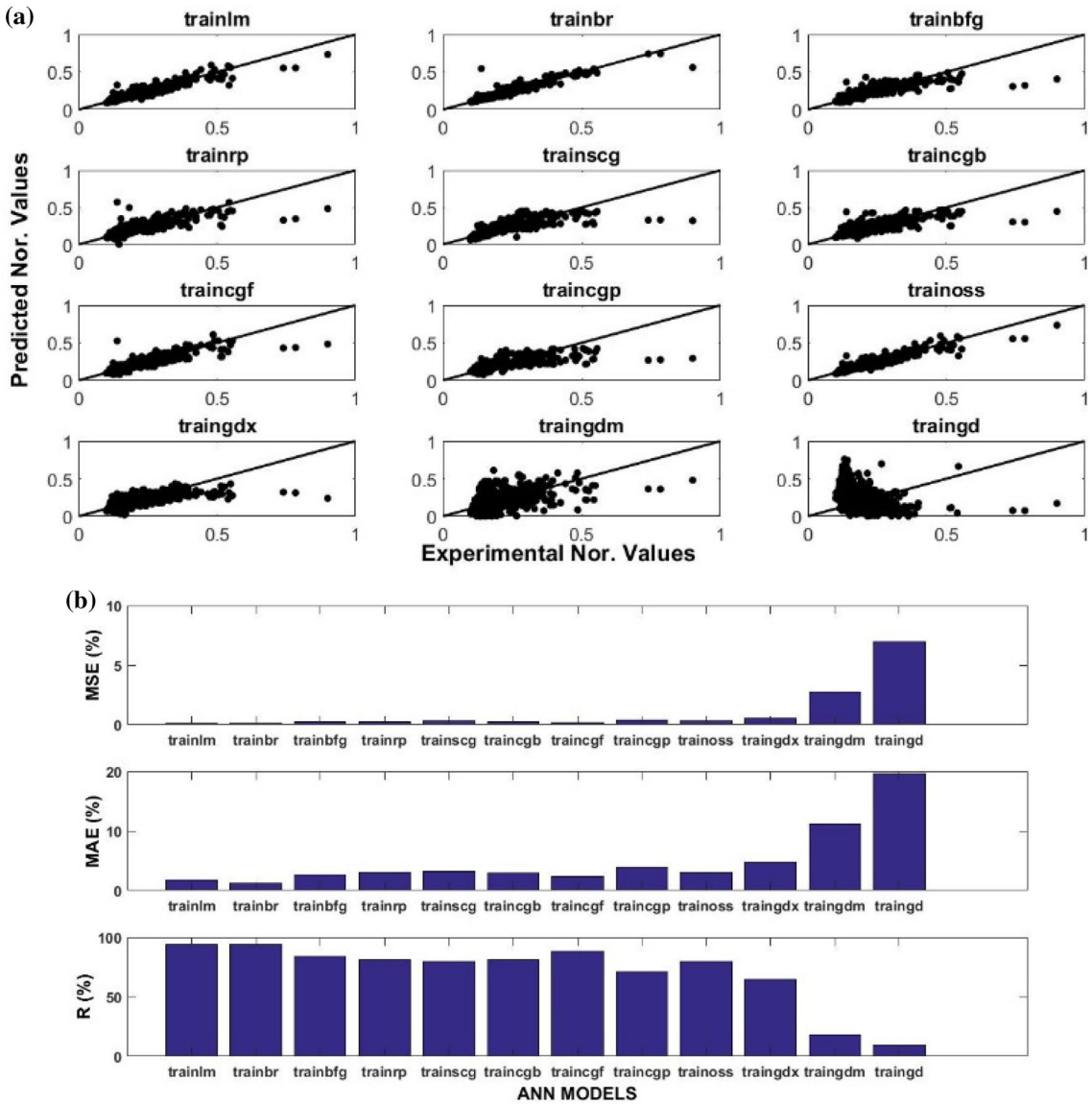


Fig. 12 **a** Performance of 12 training algorithms with random division, **b** error functions

model. As the DHL models show the better performance than SHL ANN models. The importance of the database in defining the architecture of ANNs becomes evident when comparing to other ANNs developed in the previous studies [15]. Form this comparison it is evident that the use of different databases can lead to differences in the structure (architecture) of the ANN which, however, does not necessarily result in more accurate predictions. Changes in the architecture of the ANN may also occur when extending the database upon which the training process of the ANNs is based. From this comparative study following three guidelines are established.

(a) Double hidden layers should be used for ANN models.
(b) Each hidden layer should have twice number of input parameters.
(c) Sigmoid and Tanh activations function should be used between the layers of MBP.

4.5 IC-5: different input parameters

For this step, different combinations of the input parameters are tried to investigate their effect on the ANN predictions. The selection of these combinations is based on the available physical models developed to date for describing the mechanics underlying RC structural response at the ULS. A total of 6 different ANN models are developed

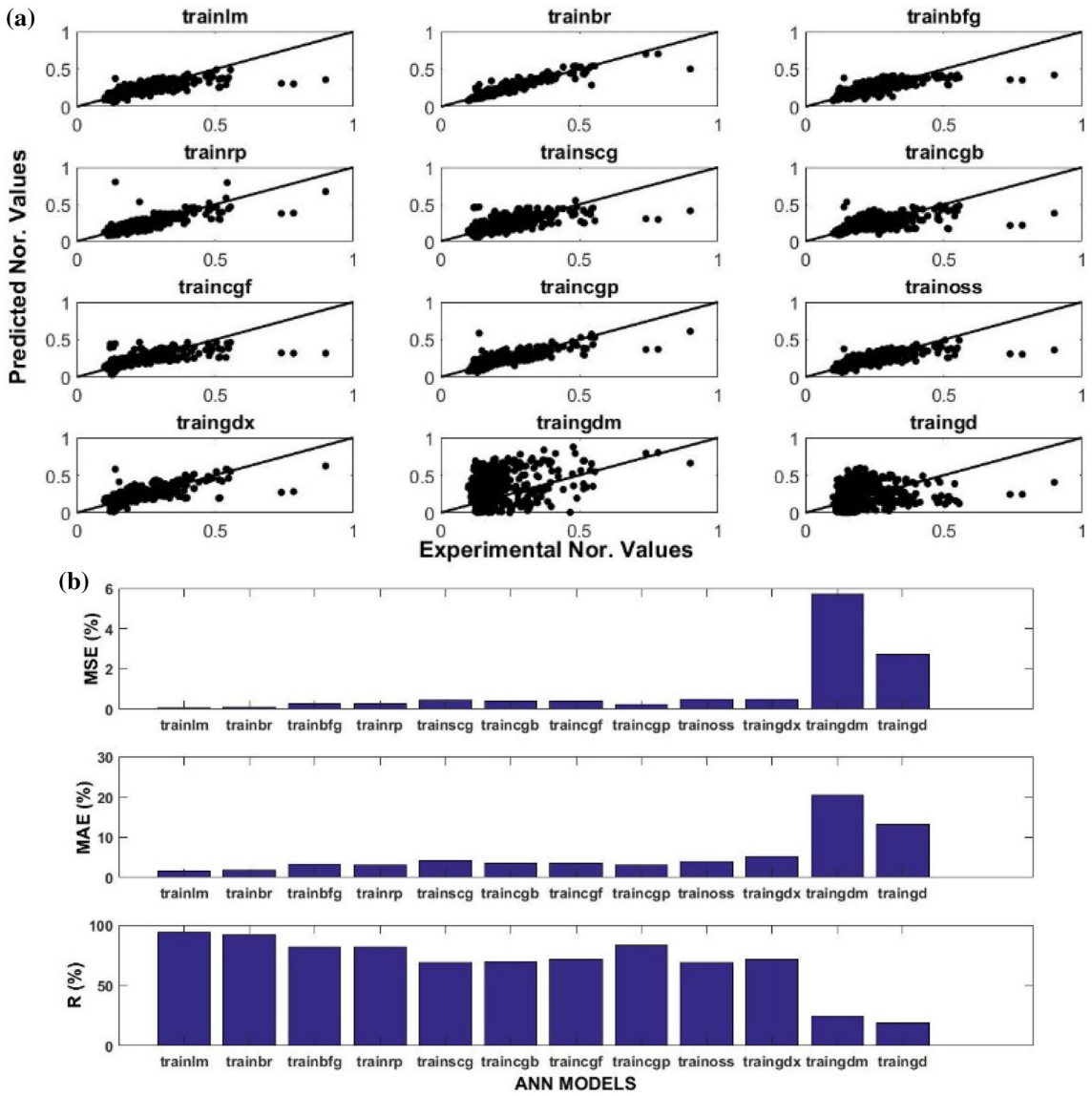


Fig. 13 **a** Performance of 12 training algorithms with constant division, **b** error functions

Table 8 Different input parameters for ANN models with SHL

S. no	Model	Input	Function of hidden	No of hidden neurons	Output
1	ST-4-H-1	$b, d, a_v/d, f_c$	Sigmoid	5–10	[50]
2	TT-4-H-1	$b, d, a_v/d, f_c$	tanghn	5–10	[50]
3	ST-5-H-1	$b, d, a_v/d, f_c, \rho_l$	Sigmoid	6–12	[50]
4	TT-5-H-1	$b, d, a_v/d, f_c, \rho_l$	tanghn	6–12	[50]
5	ST-6-H-1	$b, d, a_v/d, f_c, \rho_l, fy$	Sigmoid	7–14	[50]
6	TT-6-H-1	$b, d, a_v/d, f_c, \rho_l, f_y$	tanghn	7–14	[50]

for the case of the RC beams without stirrups (BWOS) as described in Table 12.

Table 12 also presents statistical information concerning the level of correlation achieved between the predicted values of load-bearing capacity obtained from the ANN models and their experimentally established counterparts. The most efficient/optimal ANN model, capable of providing the most accurate predictions, is usually assumed

Table 9 R^2 value for training, validation and testing for ANN models with SHL

Sr.	ANN models	Training 60%	Validation 20%	Testing 20%	All 100%
1	ST-4-9-1	0.955	0.947	0.932	0.948
2	TT-4-10-1	0.969	0.957	0.933	0.956
3	ST-5-7-1	0.991	0.968	0.977	0.984
4	TT-5-9-1	0.984	0.979	0.957	0.977
5	ST-6-9-1	0.986	0.984	0.976	0.983
6	TT-6-10-1	0.983	0.972	0.977	0.98

Table 11 R^2 value for training, validation and testing for NN models for DHL

S. no	ANN models	Training 60%	Validation 20%	Testing 20%	All 100%
1	ST-4-8-8-1	0.966	0.932	0.932	0.95
2	TT-4-8-8-1	0.96	0.958	0.939	0.956
3	ST-5-8-8-1	0.986	0.98	0.949	0.981
4	TT-5-9-9-1	0.991	0.964	0.972	0.98
5	ST-6-12-12-1	0.989	0.984	0.975	0.985
6	TT-6-9-9-1	0.992	0.982	0.972	0.984

to be the ANN characterised by the highest value of correlation factor (*R*). In the present study, in addition to the correlation factor (*R*), two more error functions—the Mean Square Error (*MSE*) and the Mean Absolute Error (*MAE*)—are also considered. The most optimized ANN model should be associated with the highest value of *R* and the smallest values of *MSE* and *MAE*. Figure 14a, b provide the values of the three error functions presently considered by the different ANN models developed for above mentioned DB. Based on Fig. 14a, b it is observed the *BWOS-1*, exhibited the highest value of *R* in combination with the smallest values of *MSE* and *MAS* while at the same time employing the smallest number of parameters. This study established that by using all available parameters, ANN model gives best performance.

4.6 Established guidelines for ANN models

From the above discussion following guidelines are established.

(a) Randomly Division database should be used for subsets (i.e. Training, Validation and Testing).
(b) LM-training algorithms should be used for MBP.
(c) Double hidden layers should be used for ANN models.
(d) Each hidden layer should have twice number of input parameters.
(e) Sigmoid and Tanh activations function should be used between the layers of MBP.
(f) By using all available parameters, ANN model gives best performance.

Bases on these guidelines an ANN model is trained for the case study i.e. BWOS, as shown in Fig. 14a, b, BWOS-1 is the optimized model and it is further used for the parametric study in the next section. Furthermore, the validation of this ANN model is carried out by using NLFEA package i.e. ABAQUS.

4.7 Parametric studies for BWOS

The effect V_{Exp}/V_{ANN} against all critical parameters are plotted in Fig. 15 for the case of RC beams without stirrups (BWOS). As expected from the comparative study sections, the predictions of the ANN (V_{ANN}) are in good agreement with their experimentally established counterparts. For only one case having width and depth 150 mm showed some deviation from the unit line. However, for all other samples, the ratio is closed to the unit line. That case has av/d ratio at 4 and fc value above 80 MPa.

4.8 Validation with FEA (BWOS)

The validation of ANN predictions is also achieved through the use of a well as established commercial package (i.e. ABAQUS) which is capable of carrying out three-dimensional (3-D) nonlinear finite element analysis (NLFEA) while realistically accounting for the brittle nature and triaxiality

Table 10 Different input parameters with DHL

S. no	Model	Input	Function of hidden	No of hidden neurons	Output
1	ST-4-H-H-1	*b, d, a_v/d, fc*	Sigmoid	5–10	[50]
2	TT-4-H-H-1	*b, d, a_v/d, fc*	Tanghn	5–10	[50]
3	ST-5-H-H-1	*b, d, a_v/d, fc, ρ_l*	Sigmoid	6–12	[50]
4	TT-5-H-H-1	*b, d, a_v/d, fc, ρ_l*	Tanghn	6–12	[50]
5	ST-6-H-H-1	*b, d, a_v/d, fc, ρ_l, fy*	Sigmoid	7–14	[50]
6	TT-6-H-H-1	*b, d, a_v/d, fc, ρ_l, fy*	Tanghn	7–14	[50]

Table 12 Different input parameters for ANN models of BWOS

Vu		Min	Max	Diff	Avg.	SD	COV
DB-I	$b,d,a_v/d,\rho l,f_{yf},f_c$	7	585	578	72.05	56.39	0.78
BWOS-1	$b,d,a_v/d,\rho l,f_{yf},f_c$	−3	554	557	73.87	55.1	0.75
BWOS-2	$b,d,a_v/d,M_f,f_c$	6	578	572	70.97	55.15	0.78
BWOS-3	$b/d,a_v/d,M_f/f_cbd^2$	5	577	572	78.17	67.31	0.86
BWOS-4	$d,a_v/d,M_f/bd^2,f_c$	3	552	549	74.48	57.09	0.77
BWOS-5	$d,b/d,a_v/d,M_f/f_cbd^2$	6	595	589	76.36	62.39	0.82
BWOS-6	$d,b/d,a_v/d,Mf/f_cbd^2,f_c$	7	584	577	71.93	56.83	0.79

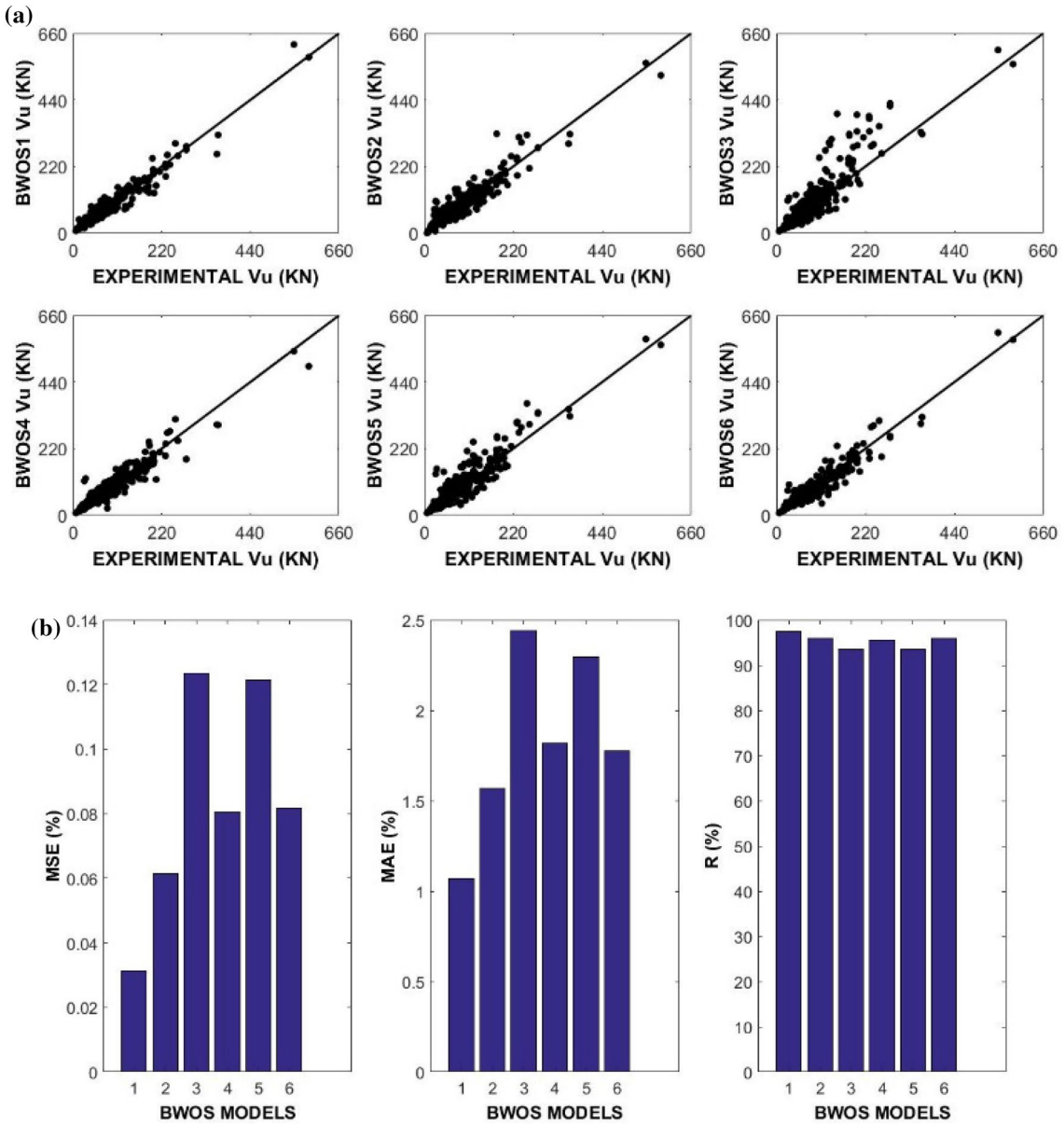


Fig. 14 **a** Predictions of BWOS ANN models, **b** error functions

characterising concrete material behaviour. A damage plasticity model is employed for describing concrete material behaviour and a simple bilinear elasto-plastic hardening model is used for describing the behaviour of steel. All essential parameters are trained on the experimental cylinder samples for concrete. Full bond is assumed

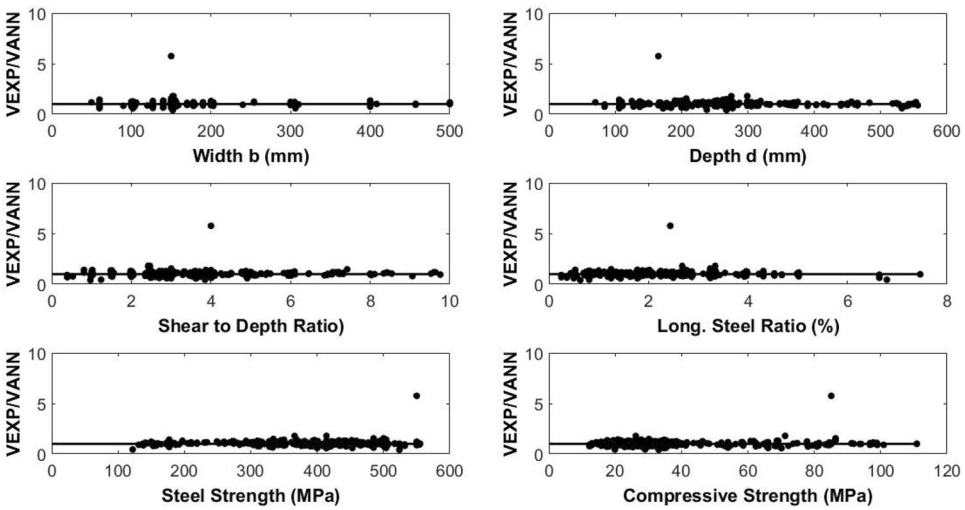


Fig. 15 Parametric study for BWOS model

in order to describe the interaction between steel and concrete. The static problem is solved through an iterative procedure. For the case of static loading the load is applied monotonically until failure in the form of displacement increments (displacement control). The concrete is modelled through the use of 3D 8 node elements whereas the steel reinforcement is modelled through the use if steel elements. For the calibration of ABAQUS model authors used the previous experiences as described [51]. Table 13 shows the information of all parameters for the considered cases for the BWOS.

Figure 16 shows the comparison of experimental Load–deflection curve with NLFEA i.e. ABAQUS, describing the response of OA1, OA2 and OA3 [52]. These three case studies are selected from above described database i.e. BWOS. The NLFEA with ABAQUS shows that the ABAQUS load–deflection curve are in good agreement with the experimental results. Also Fig. 17 shows the comparison between the ANN predicted values and Abaqus against their respective counterpart experimental values. For both the cases ANN values and very close to the experimental and Abaqus.

5 Conclusions

The objective of the present work is to establish the undefined rules of ANN architecture. These proposed guidelines are also discussed with the previous established models for predicting the response of RC members at ULS. The most important conclusions derived from this comparative study are:

1. Development and enrichment of experimental database for predicting the structural response of RC members at ULS. For the better prediction of ANN models, new samples are required to enrich the poorly populated region of individual parameter in each the databases i.e. EXP by using realistic method.
2. The randomized division of enriched database into further sub-sets i.e. Training, Validation and Testing, with the percentages of 60% 20% and 20% respectively. Among the available 12 training algorithm, LM is the best one in respect to the time and accuracy.
3. The MBP models having double layers have better performance than single layer ANN models. Also, by using the sigmoid activation function in the first two layers and tanh activation functions in last two layers is more beneficial for ANN models.

Table 13 Summary of case studies for BWOS

Name	b (mm)	h (mm)	d (mm)	a_v/d	ρ_l (%)	f_{yl} (MPa)	f_c (MPa)	ρ_w (%)	f_{yw} (MPa)	N (kN)
Bresler and Scordelis [52]										
OA-1	310	556	461	3.97	1.81	555	22.6	0	0	0
OA-2	305	561	466	4.91	2.27	555	23.7	0	0	0
OA-3	307	556	462	6.93	2.73	552	37.6	0	0	0

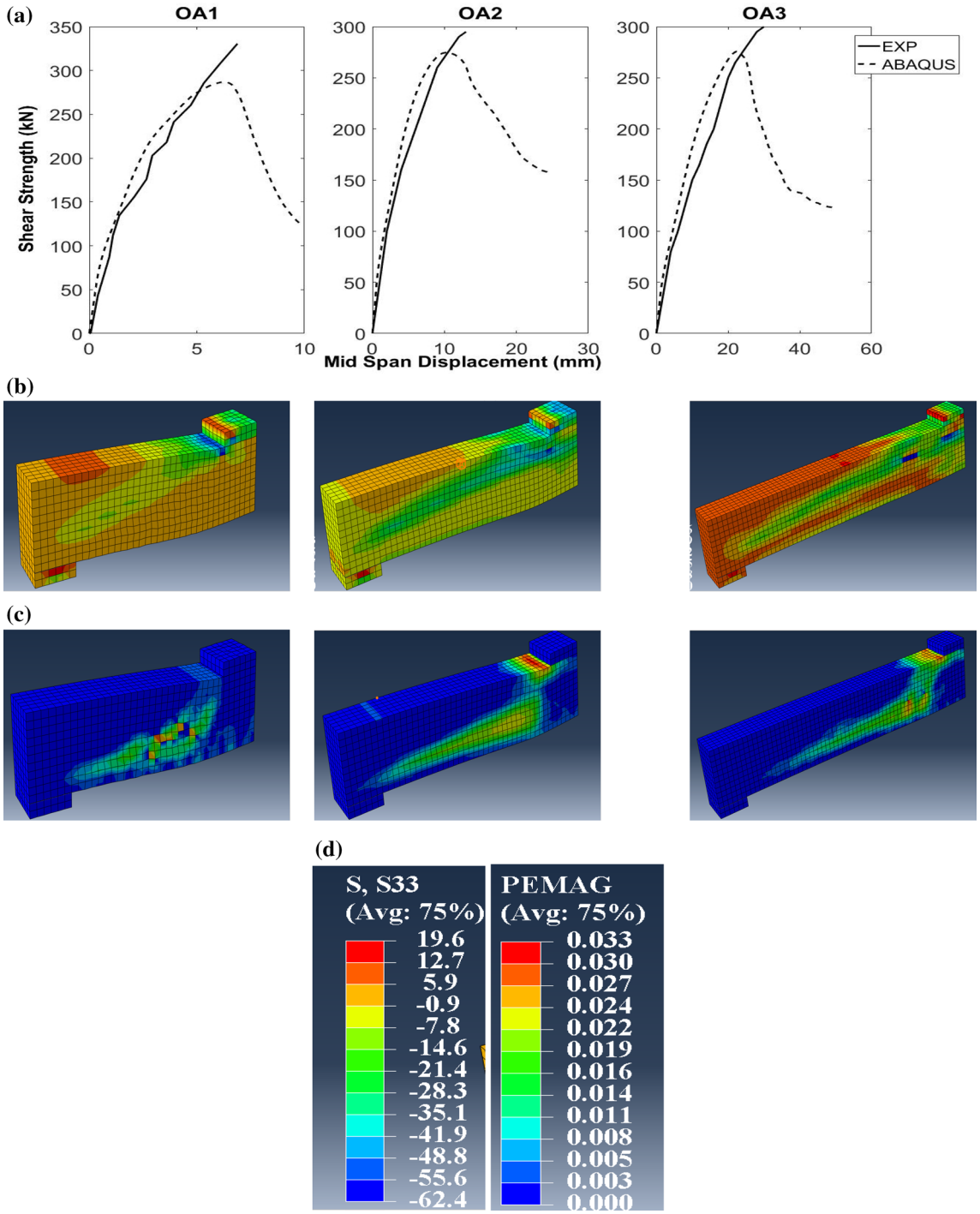


Fig. 16 **a** Comparison of experimentally and numerically established load–deflection curves accompanied by the numerical predicted, **b** flow of compressive stresses and **c** damage developing along the beam span for the case of BWOS specimens OA1, OA2 and OA3 (**d**) values for stress and strain

4. Based on these above guidelines, ANNs is based on heuristic approaches which do not strictly adhere to the principles of theoretical mechanics, the predictions obtained provide a closer fit to the available experimental test data.
5. Furthermore, the validation of ANN models is achieved by the selected cases studies of NLFEA. The predictions obtained via NLFEA results are in good agreement to their counterparts provided by the ANN. ANN models

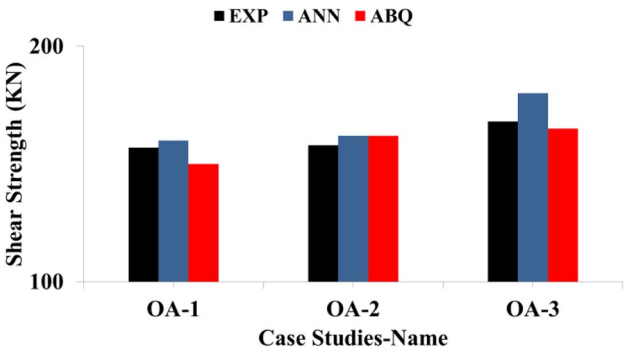


Fig. 17 Results of case studies of BWOS with ABAQUS

can predict the accurate solution for variety of problems i.e. RC beams without and with stirrups (BWOS).

As based on the outcome of this paper, authors will develop ANN models or RC beam, RC columns, RC wall and RC slabs. These models then used as the failure criteria during the pushover analysis, as described in the conferences papers "Prediction of RC Frames through the use of Hybrid Artificial Neural Network Finite Element (ANN-FE) Model", (Infrastructure and Environment Scotland 4th Postgraduate Conference 30th May 2017 University of Edinburgh, Edinburgh).

Acknowledgements Not Applicable.

Funding This research was supported by the James Watt Scholarship scheme titled: Analysis of RC Structures Employing Neural Networks (ARCSENN).

Compliance with ethical standards

Conflict of interest The authors whose names are listed in this paper certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

1. Das SK et al (2013) On soft computing techniques in various areas. Comput Sci Inf Technol 3(2):59–68
2. Chandwani V, Agrawal V, Nagar R (2013) Applications of soft computing in civil engineering: a review. Int J Comput Appl 81(10):13–20
3. Talatahari S et al (2015) Soft computing methods in civil engineering. Sci World J 2015:605871
4. Zadeh LA (1994) Fuzzy-logic, neural networks, and soft computing. Commun ACM 37(3):77–84
5. Schueremans L, Gemert VD (2004) Assessing the safety of existing structures: reliability based assessment framework, examples and application. J Civ Eng Manag 10(2):131–141
6. Mehrjoo M et al (2008) Damage detection of truss bridge joints using artificial neural networks. Expert Syst Appl 35(3):1122–1131
7. Gonzalez MP, Zapico JL (2008) Seismic damage identification in buildings using neural networks and modal data. Comput Struct 86(3–5):416–426
8. Chang CC et al (2000) Structural damage detection using an iterative neural network. J Intell Mater Syst Struct 11(1):32–42
9. Arslan MH (2010) An evaluation of effective design parameters on earthquake performance of Rc buildings using neural networks. Eng Struct 32(7):1888–1898
10. Niu L (2012) Monitoring of a frame structure model for damage identification using artificial neural networks. In: 2nd international conference on electronic & mechanical engineering and information technology. Atlantis Press, Paris, France
11. Dehkordi BZ et al (2012) Reinforced concrete frame failure prediction using neural network algorithm. J Appl Sci 12(5):498–501
12. ACI, Building Code Requirements for Structural Concrete (ACI 318-08) and Commentary. In: aci-318-08. 2008: American Concrete Institute 38800 Country Club Drive Farmington Hills, MI 48331, pp 1–471
13. EC2, Eurocode 2: design of concrete structures—part 1-1: general rules and rules for buildings. In: EN 1992-1-1. 2004: Management Centre: Avenue Marnix 17, B-1000 Brussels
14. Kotsovos MD (2013) Compressive force-path method. Springer, Heidelberg
15. Ahmad A, Cotsovos DM, Lagaros ND (2015) Assessing the reliability of RC code predictions through the use of artificial neural networks. In: The first international conference on structural safety under fire & blast. Glasgow, UK: CONFAB
16. Beale MH, Hagan MT, Demuth HB (2015) Neural network toolbox™-user's guide. The MathWorks Inc, Natick
17. Koivo HN (2008) Neural networks: basics using MATLAB neural network toolbox, pp 1–59
18. Ahmad A et al (2018) Assessing the accuracy of RC design code predictions through the use of artificial neural networks. Int J Adv Struct Eng 10(4):349–365
19. Anderson D, McNeill G (1992) Artificial neural networks technology. Kaman Sciences Corporation, New York, pp 1–83
20. Basheer IA, Hajmeer M (2000) Artificial neural networks: fundamentals, computing, design, and application. J Microbiol Methods 43(1):3–31
21. Svozil D, Kvasnicka V, Pospichal J (1997) Introduction to multilayer feed-forward neural networks. Chemom Intell Lab Syst 39(1):43–62
22. Özkan C, Erbek FS (2003) The comparison of activation functions for multispectral Landsat TM image classification. Photogramm Eng Remote Sens 69(11):1225–1234
23. Leahy P, Kiely G, Corcoran G (2008) Structural optimisation and input selection of an artificial neural network for river level prediction. J Hydrol 355(1–4):192–201
24. Abhishek K et al (2012) Weather forecasting model using artificial neural network. Proc Technol 4:311–318
25. Looney CG (1997) Pattern recognition using neural networks: theory and algorithms for engineers and scientists. Oxford University Press, Inc, Oxford
26. Hadi MNS (2003) Neural networks applications in concrete structures. Comput Struct 81(6):373–381
27. Jenkins WM (1999) A neural network for structural re-analysis. Comput Struct 72(6):687–698

28. Yun CB, Yi JH, Bahng EY (2001) Joint damage assessment of framed structures using a neural networks technique. Eng Struct 23:425–435
29. Günaydın HM, Doğan SZ (2004) A neural network approach for early cost estimation of structural systems of buildings. Int J Project Manag 22(7):595–602
30. Zapico JL, Gonzalez MP (2006) Numerical simulation of a method for seismic damage identification in buildings. Eng Struct 28(2):255–263
31. Bakhary N, Hao H, Deeks AJ (2007) Damage detection using artificial neural network with consideration of uncertainties. Eng Struct 29(11):2806–2815
32. Caglar N et al (2008) Neural networks in 3-dimensional dynamic analysis of reinforced concrete buildings. Constr Build Mater 22(5):788–800
33. Nyarko MH, Nyarko EK, Moric D (2011) A neural network based modelling and sensitivity analysis of damage ratio coefficient. Expert Syst Appl 38(10):13405–13413
34. LeCun Y et al (1998) Efficient backprop. Red Bank, NJ 07701-703, USA, pp 1–44
35. Giordano F, Rocca ML, Perna C (2014) Input variable selection in neural network models. Commun Stat Theory Methods 43(4):735–750
36. Sarle WS (1995) Stopped training and other remedies for overfitting. In: Proceedings of the 27th symposium, pp 1–10
37. Olsson A, Sandberg G, Dahlblom O (2003) On latin hypercube sampling for structural reliability analysis. Struct Saf 25(1):47–68
38. Abaqus, Abaqus 6.12 documentation. Providence, Rhode Island, US. 2012: Simulia, Dassault Systemes
39. Rafiq MY, Bugmann G, Easterbrook DJ (2001) Neural network design for engineering applications. Int J Comput Struct 79(17):1541–1552
40. Krogh A, Vedelsby J (1995) Neural network ensembles, cross validation, and active learning. Adv Neural Inf Process Syst 7:21–238
41. Rojas R (1996) The backpropagation algorithm. In: Neural networks, pp 151–184. Springer, Berlin
42. Wilson DR, Martinez TR (2003) The general inefficiency of batch training for gradient descent learning. Neural Netw 16(10):1429–1451
43. Sapna S, Tamilarasi A, Kumar MP (2012) Backpropagation learning algorithm based on Levenberg Marquardt algorithm, pp 393–398
44. Gupta JND, Sexton RS (1999) Comparing backpropagation with a genetic algorithm for neural network training. Omega-Int J Manag Sci 27(6):679–684
45. Nawi NM, Khan A, Rehman MZ (2013) A new Levenberg Marquardt based back propagation algorithm trained with cuckoo search. Proc Technol 11:18–23
46. Mojtaba N, Bali M, Naeej MR, Amiri JV (2013) Prediction of lateral confinement coefficient in reinforced concrete columns using m5′ machine learning method. KSCE J Civ Eng 17(7):1714–1719
47. Utans J, Moody J, Rehfuss S, Siegelmannt H (1995) Input variable selection for neural networks: application to predicting the U.S. Business Cycle. IEEE Trans Knowl Data Eng, pp 118–122
48. Castellano G, Fanelli AM (2000) Variable selection using neural-network models. Neurocomputing 31(1–4):1–13
49. Saxen H, Pettersson F (2006) Method for the selection of inputs and structure of feedforward neural networks. Comput Chem Eng 30(6–7):1038–1045
50. Hwang C-L, Tran V-A (2015) A study of the properties of foamed lightweight aggregate for self-consolidating concrete. Constr Build Mater 87:78–85
51. Raza A, Khan QUZ, Ahmad A (2019) Numerical investigation of load-carrying capacity of gfrp-reinforced rectangular concrete members using CDP model in ABAQUS. Advances in Civil Engineering, 2019, pp 1–21
52. Bresler B, Scordelis AC (1963) Shear strength of reinforced concrete beams. J Am Concr Inst 60(4):51–74