ANN-based Shear Capacity of Steel Fiber-Reinforced Concrete Beams Without Stirrups

Miguel Abambres, E Lantsoght

ANN-based Shear Capacity of Steel Fiber-Reinforced Concrete Beams Without Stirrups

M. Abambres GS, email, E. Lantsoght GS, email

Abstract: Comparing experimental results on the shear capacity of steel fiber-reinforced concrete (SFRC) beams without mild steel stirrups, to the ones predicted by current design equations and other available formulations, still shows significant differences. In this paper we propose the use of artificial intelligence to estimate the shear capacity of these members. A database of 430 test results reported in the literature is used to develop an artificial neural network-based formula that predicts the shear capacity of SFRC beams without shear reinforcement. The proposed model yields maximum and mean relative errors of 0.0% for the 430 data points, which represents a better prediction (mean V_{test} / V_{ANN} = 1.00 with a coefficient of variation of 1×10^{-15}) than the existing expressions, where the best model yields a mean value of V_{test} / V_{pred} = 1.01 and a coefficient of variation of 27%.

Keywords: Experimental Data, Artificial Neural Networks, Design Formula, Concrete Beams, Steel Fiber-Reinforced Concrete, Shear Capacity.

id: hal-02293627

Important Note: The first author has been proposing several ANN-based models, in each case designed and tested for a fairly limited amount of data (especially when empirical). Regardless the high quality of the predictions yielded by some model for the used data, the reader should not **blindly** accept that model as accurate for any other instances falling inside the input domain of the design dataset. Any analytical approximation model must undergo extensive validation before it can be taken as reliable (the more inputs, the larger the validation process). Models proposed until that stage are part of a learning process towards excellence.

1. Introduction

Since concrete is strong in compression but weak in tension, adding steel fibers to the material can be a solution to the limited strength in tension – they keep crack widths small (Amin et al. 2016). In structural applications, steel fiber-reinforced concrete is combined with regular steel reinforcement.

A failure mode where crack shape and width are key factors is shear failure. When steel fibers are added to a concrete mix, all shear-carrying mechanisms are affected (Lantsoght 2019a). Those mechanics are still not fully understood, which makes it quite meaningful to research the behavior of steel fiber-reinforced concrete (SFRC) with longitudinal reinforcement and no stirrups. Such an approach can study the contribution of steel fibers to the shear capacity of structural concrete (e.g., Torres and Lantsoght 2019), and optimal combinations of steel fiber reinforcement and regular stirrups can be searched. Such combinations are particularly useful for (i) joints where rebar congestion makes concreting difficult (Singh and Jain 2014), and (ii) bridge girders where the addition of steel fibers can lead to a more durable structure, since cracks will be less wide and more distributed.

Even though the mechanics of the shear resistance of SFRC is not fully understood, several formulas are proposed in the literature and current design codes, as described in Lantsoght 2019b. In these expressions, the properties of the fibers are usually measured by the fiber factor $F = V_f l_f \rho_f / d_f$ (Narayanan and Kareem-Palanjian 1984), where V_f is the fiber volume fraction with respect to the concrete volume, l_f the length of the fiber, d_f the diameter of the fiber, and ρ_f the fiber bond factor (depends on the fiber type). Lantsoght (2019b) assessed the performance of the aforementioned formulas against experimental results reported in the literature, and it was concluded that work still needs to be done to effectively predict the shear capacity of SFRC beams without stirrups. Note that some of those analytical approaches have been developed by means of artificial intelligence (AI) (e.g., Sarveghadi et al. 2015, Greenough and Nehdi 2008, Hossain et al. 2016, Kara 2013). The differences for the model proposed in this work, also derived from an AI technique (called artificial neural network), are the following: (i) the dataset used herein, comprising 430 lab tests, is significantly larger than the ones used in the previous studies; (ii) a larger number of ANNs were simulated for this work; and (iii) the model proposed in this paper yields smaller mean and maximum errors for the aforementioned 430 experimental instances.

2. Data Gathering

A database comprising 430 test results (outcomes of repeated tests were averaged) reported in the literature (see Lantsoght 2019b) was used to feed all ANN models. Tab. 1 shows the input and output variables/ranges considered in this study. Geometrical variables include the beam width (b) and effective depth (d), and the clear shear span to effective depth ratio (a_v/d), as depicted in Fig. 1. The reinforcement ratio $\rho = A_s / (bd)$, where A_s is the rebar area, and the steel yield strength (f_y), characterize the longitudinal reinforcement. Concrete mix is characterized by the maximum aggregate size (d_a) and the average concrete compressive strength ($f_{c,cyl}$, taken from cylinders). Lastly, the fiber factor (F) described before and the tensile strength of the steel fibers (f_{tenf}) were also used as inputs. The output is the sectional shear capacity (V_{utot}) as shown in Fig. (b), which includes the beam self-weight. In total, nine input variables and one output variable were adopted. The dataset considered is available in Developer (2019a).

Tab.1. Input and output variables considered in the dataset, including ranges of values.

VARIABLES			ANN INPUT	Min	Max
Geometry	b (mm)	beam width	1	50	610
	d (mm)	beam effective depth	2	85.3	1118
	a_v/d (-)	clear shear span to effective depth ratio	3	0.20	5.95
Rebar properties	ρ (-)	reinforcement ratio	4	0.0037	0.0572
	f_y (MPa)	yield strength of steel	5	257.9	900
Concrete properties	d_a (mm)	maximum aggregate size	6	0.4	22
	$f_{c,cyl}$ (MPa)	average concrete compressive strength	7	9.8	215
Fiber properties	F (-)	fiber factor	8	0.08	2.86
	f_{tenf} (MPa)	tensile strength of fiber	9	260	4913
Output	V_{utot} (kN)	sectional shear capacity	1	12.89	1480.85

The majority of SFRC shear experiments were carried out on heavily longitudinally reinforced beams, and small geometries. These large reinforcement ratios are not commonly used in practice. Concrete mix features show that both mortars and low to ultra-high strength concretes were used. Many fiber types were employed, namely (i) hooked, (ii) crimped, (iii) straight

smooth, (iv) mixed (hooked + straight), (v) with a flat end, (vi) flat, (vii) round , (viii) mill-cut, (ix) made of straight mild steel, (x) made of brass-coated high-strength steel, (xi) chopped with butt ends, (xii) recycled, and (xiii) corrugated. Most experiments (63%) used hooked-end fibers, and in most cases a fiber factor within 0.5-1 was adopted (higher values result in concrete mixes with low workability) – for further details on fibers, please check Lantsoght 2019a.

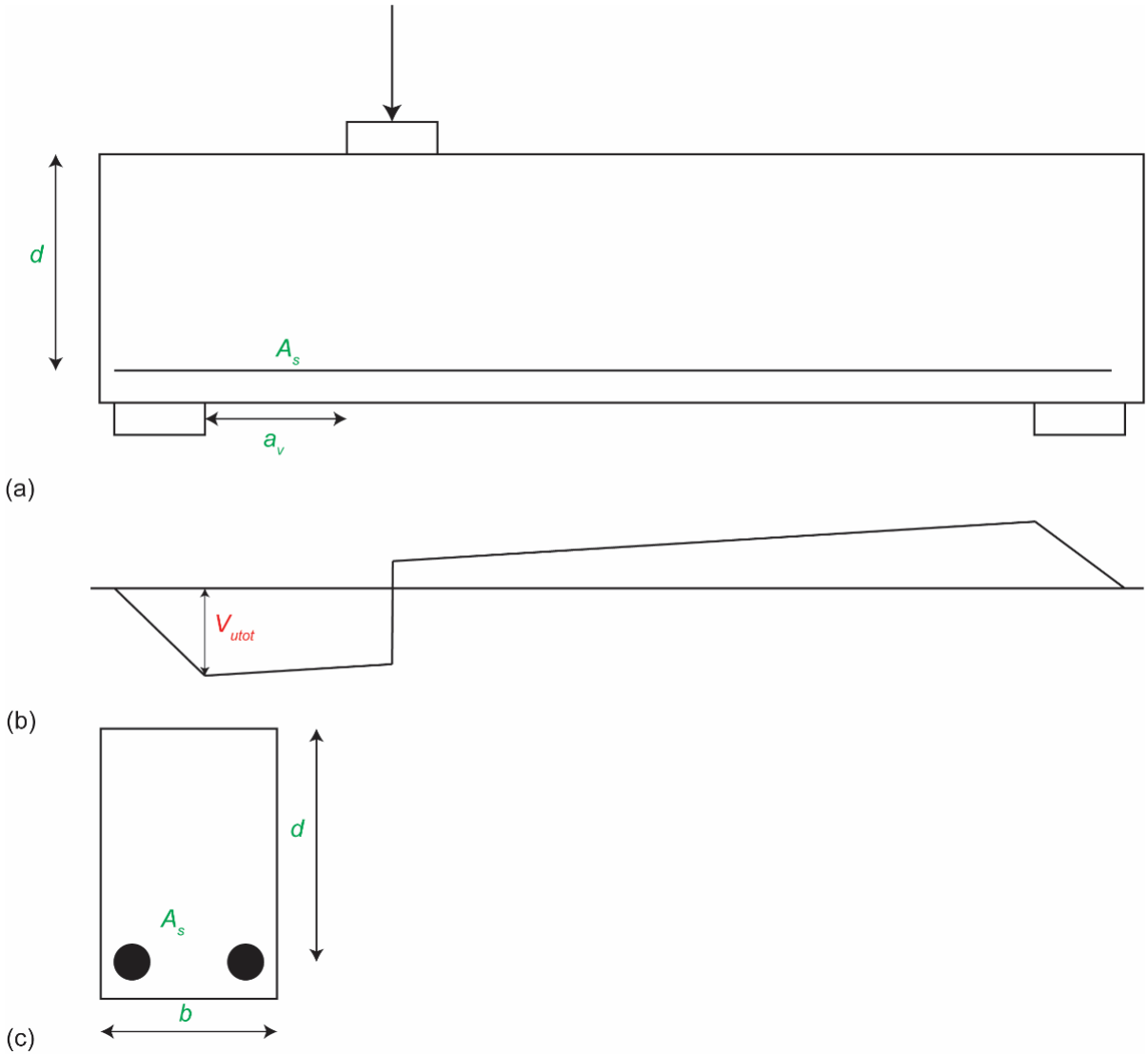

Fig. 1. Problem illustration: (a) beam side view, (b) sectional shear diagram, showing maximum value V_{utot} (self-weight included), and (c) beam cross-section.

3. Artificial Neural Networks

3.1 Introduction

Machine learning, one of the six disciplines of Artificial Intelligence (AI) without which the task of having machines acting humanly could not be accomplished**,** allows us to 'teach' computers how to perform tasks by providing examples of how they should be done (Hertzmann and Fleet 2012). The decision about which modelling technique to use in an arbitrary problem depends primarily on the availability of both the theory explaining the underlying phenomena and the data. When there is abundant data (also called examples or patterns) explaining a certain phenomenon, but its theory richness is poor, machine learning (e.g., Artificial Neural Networks) can be a perfect tool. An illustration of the several possible scenarios is presented in Basheer and Hajmeer (2000), as shown in Fig 2 (shadowed areas represent regions where any of the contiguous tools might me used).

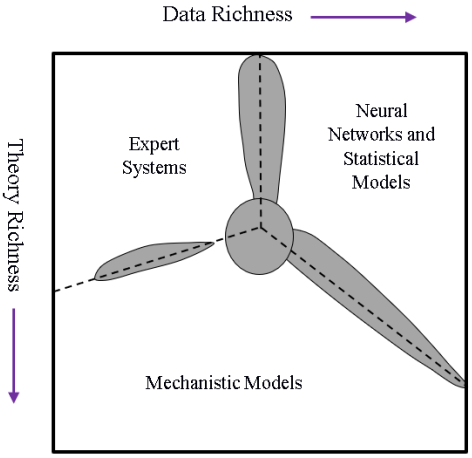

Fig. 2. Suitable modelling techniques as function of theory and data richness (Basheer & Hajmeer 2000).

The world is quietly being reshaped by machine learning, being the Artificial Neural Network (also referred in this manuscript as ANN or neural net) its (i) oldest (McCulloch and Pitts 1943) and (ii) most powerful (Hern 2016) technique. ANNs also lead the number of

practical applications, virtually covering any field of knowledge (Wilamowski and Irwin 2011, Prieto et. al 2016). In its most general form, an ANN is a mathematical model designed to perform a particular task, based in the way the human brain processes information, i.e. with the help of its processing units (the neurons). ANNs have been employed to perform several types of real-world basic tasks. Concerning functional approximation, ANN-based solutions are frequently more accurate than those provided by traditional approaches, such as multi-variate nonlinear regression, besides not requiring a good knowledge of the function shape being modelled (Flood 2008).

The general ANN structure consists of several nodes disposed in L vertical layers (input layer, hidden layers, and output layer) and connected between them, as depicted in Fig. 3. Associated to each node in layers 2 to L, also called neuron, is a linear or nonlinear transfer (also called activation) function, which receives the so-called net input and transmits an output . All ANNs implemented in this work are called feedforward, since data presented in the input layer flows in the forward direction only, i.e. every node only connects to nodes belonging to layers located at the right-hand-side of its layer, as shown in Fig. 3. ANN's computing power makes them suitable to efficiently solve small to large-scale complex problems.

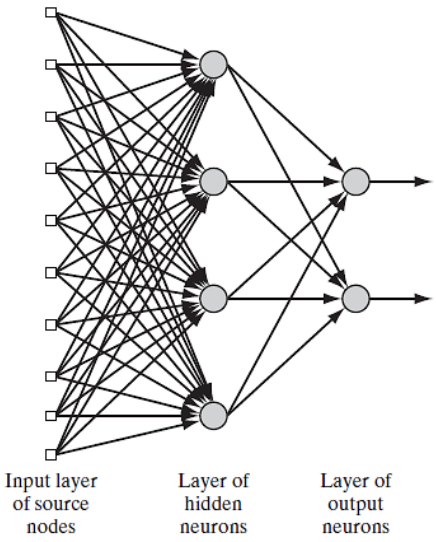

Fig. 3. Example of a feedforward neural network (Haykin 2009).

3.2 Learning

Each connection between 2 nodes is associated to a synaptic weight (real value), which, together with each neuron's bias (also a real value), are the most common types of neural net unknown parameters that will be determined through learning. Learning is nothing else than determining network unknown parameters through some algorithm in order to minimize network's performance measure, typically a function of the difference between predicted and target (desired) outputs. When ANN learning has an iterative nature, it consists of three phases: (i) training, (ii) validation, and (iii) testing. From previous knowledge, examples or data points are selected to train the neural net, grouped in the so-called training dataset. Those examples are said to be 'labelled' or 'unlabeled', whether they consist of inputs paired with their targets, or just of the inputs themselves – learning is called supervised (e.g., functional approximation, classification) or unsupervised (e.g., clustering), whether data used is labelled or unlabeled, respectively. During an iterative learning, while the training dataset is used to tune network unknowns, a process of cross-validation takes place by using a set of data completely distinct from the training counterpart (the validation dataset), so that the generalization performance of the network can be attested. Once 'optimum' network parameters are determined, typically associated to a minimum of the validation performance curve (called early stop – see Fig. 3 in Abambres et al. 2018), many authors still perform a final assessment of model's accuracy, by presenting to it a third fully distinct dataset called 'testing'. Heuristics suggests that early stopping avoids overfitting, i.e. the loss of ANN's generalization ability. One of the causes of overfitting might be learning too many input-target examples suffering from data noise, since the network might learn some of its features, which do not belong to the underlying function being modelled (Haykin 2009).

3.2.1 The Universal Approximation Theorem

For a nonlinear input-output mapping, this theorem states (Haykin 2009) that a single hidden layer multi-layer perceptron network (MLPN), with (i) any bounded, monotone-increasing and continuous activation function for the hidden neurons, and (ii) an identity transfer function for

the output neurons, is sufficient to compute an arbitrarily good approximation of any continuous function in a general n-dimensional space – the absolute difference between any estimated and target outputs can be less than any $\varepsilon > 0$, for all input space values. However, the theorem does not say that the aforementioned network features are optimal in the sense of learning time or generalization.

3.3 Implemented ANN features

The 'behavior' of any ANN depends on many 'features', having been implemented 15 ANN features in this work (including data pre/post processing ones). For those features, it is important to bear in mind that no ANN guarantees good approximations via extrapolation (either in functional approximation or classification problems), i.e. the implemented ANNs should not be applied outside the input variable ranges used for network training. Since there are no objective rules dictating which method per feature guarantees the best network performance for a specific problem, an extensive parametric analysis (composed of nine parametric sub-analyses) was carried out to find 'the optimum' net design. A description of all methods/formulations implemented for each ANN feature (see Tabs. 2-4) – they are a selection from state of art literature on ANNs, including both traditional and promising modern techniques, can be found in previous published works (e.g., Abambres and Lantsoght 2018) – the reader might need to go through it to fully understand the meaning of all variables and acronyms reported in this manuscript. The whole work was coded in MATLAB (The Mathworks, Inc. 2017), making use of its neural network toolbox when dealing with popular learning algorithms (1-3 in Tab. 4). Each parametric sub-analysis (SA) consists of running all feasible combinations (also called 'combos') of pre-selected methods for each ANN feature, in order to get performance results for each designed net, thus allowing the selection of the best ANN according to a certain criterion. The best network in each parametric SA is the one exhibiting the smallest average relative error (called performance) for all learning data.

Tab. 2. Implemented ANN features (F) 1-5.

FEATURE METHOD	F1	F2	F3	F4	F5
	Qualitative Var Represent	Dimensional Analysis	Input Dimensionality Reduction	% Train-Valid-Test	Input Normalization
1	Boolean Vectors	Yes	Linear Correlation	80-10-10	Linear Max Abs
2	Eq Spaced in]0,1]	No	Auto-Encoder	70-15-15	Linear [0, 1]
3	-	-	-	60-20-20	Linear [-1, 1]
4	-	-	Ortho Rand Proj	50-25-25	Nonlinear
5	-	-	Sparse Rand Proj	-	Lin Mean Std
6	-	-	No	-	No

Tab. 3. Implemented ANN features (F) 6-10.

FEATURE METHOD	F6	F7	F8	F9	F10
	Output Transfer	Output Normalization	Net Architecture	Hidden Layers	Connectivity
1	Logistic	Lin [a, b] = 0.7[φ_{min}, φ_{max}]	MLPN	1 HL	Adjacent Layers
2	-	Lin [a, b] = 0.6[φ_{min}, φ_{max}]	RBFN	2 HL	Adj Layers + In-Out
3	Hyperbolic Tang	Lin [a, b] = 0.5[φ_{min}, φ_{max}]	-	3 HL	Fully-Connected
4	-	Linear Mean Std	-	-	-
5	Bilinear	No	-	-	-
6	Compet	-	-	-	-
7	Identity	-	-	-	-

Tab. 4. Implemented ANN features (F) 11-15.

FEATURE METHOD	F11	F12	F13	F14	F15
	Hidden Transfer	Parameter Initialization	Learning Algorithm	Performance Improvement	Training Mode
1	Logistic	Midpoint (W) + Rands (b)	BP	-	Batch
2	Identity-Logistic	Rands	BPA	-	Mini-Batch
3	Hyperbolic Tang	Randnc (W) + Rands (b)	LM	-	Online
4	Bipolar	Randnr (W) + Rands (b)	ELM	-	-
5	Bilinear	Randsmall	mb ELM	-	-
6	Positive Sat Linear	Rand [-Δ, Δ]	I-ELM	-	-
7	Sinusoid	SVD	CI-ELM	-	-
8	Thin-Plate Spline	MB SVD	-	-	-
9	Gaussian	-	-	-	-
10	Multiquadratic	-	-	-	-
11	Radbas	-	-	-	-

With respect to the ANN formulation used in Abambres and Lantsoght (2018), a few changes were carried out for this work. They were (i) the elimination of performance improvements (feature 14), although that feature is still integrated in the code for eventual future use, and (ii) the algorithm used in feature 4. The latter is described next.

3.3.1 Training, Validation and Testing Datasets (feature 4)

Four distributions of data (methods) were implemented, namely p_t-p_v-p_{tt} = {80-10-10, 70-15-15, 60-20-20, 50-25-25}, where p_t-p_v-p_{tt} represent the amount of training, validation and testing examples as % of all learning data (P), respectively. Aiming to divide learning data into training, validation and testing subsets according to a predefined distribution p_t-p_v-p_{tt}, the following algorithm was implemented (all variables are involved in these steps, including qualitative ones after converted to numeric):

1) Reduce p_t-p_v-p_{tt} values by 10 units each.
2) For each variable q (row) in the complete input dataset, compute its minimum and maximum values.
3) Select all patterns (if some) from the learning dataset where each variable takes either its minimum or maximum value. Those patterns must be included in the training dataset, regardless what p_t is. However, if the number of patterns is lower than the rounding of p_t * P/100, more patterns should be added to the training set in the following way:
    a. Compute the number of patterns (Lp_t) that need to be added to the initially selected training patterns to equal round(p_t * P/100).
    b. Randomly select 10.000 combinations of Lp_t patterns from all those not included in the training set defined prior a).
    c. For each combination/scenario in b), add those Lp_t patterns to the set of training patterns defined prior a), and label all remaining learning patterns as “validation + testing”.
    d. For each scenario in c), and for each pattern labeled as “validation + testing”, check if that pattern has at least one input variable that takes a value not taken

by any pattern in the training set. If it hasn´t, then that pattern should be moved to the training set.

e. Among all 10.000 scenarios of training and "validation + testing" subsets addressed in b) till d), the "winner" should be the one guaranteeing the amount of training data (P_{t*}) closest to round(p_t * P/100).

f. If the winning training set selected in e) guarantees | P_{t*} / P - p_t | ≤ 0.2, then that becomes the training data to be taken for simulation. Otherwise, the training data should be selected according to step 2 in subsection 3.3.4 of Abambres et al. (2018).

4) Increase p_t-p_v-p_{tt} values by 10 units each (to re-obtain the original input values – recall step 1).

5) In order to select the validation patterns, randomly select p_v / (p_v + p_{tt}) of those patterns not belonging to the previously defined training dataset. The remainder defines the testing dataset.

It might happen that the actual distribution p_t-p_v-p_{tt} to be used in the simulation is not equal to the one imposed *a priori* (before step 1).

3.4 Network Performance Assessment

Several types of results were computed to assess network outputs, namely (i) maximum error, (ii) % errors greater than 3%, and (iii) performance, which are defined next. All abovementioned errors are relative errors (expressed in %) based on the following definition, concerning a single output variable and data pattern,

$$e_{qp} = 100 \left| \frac{d_{qp} - y_{qLp}}{d_{qp}} \right| \quad , \quad (1)$$

where (i) d_{qp} is the q^{th} desired (or target) output when pattern p within iteration i (p=1,…, P_i) is presented to the network, and (ii) y_{qLp} is net's q^{th} output for the same data pattern. Moreover,

denominator in eq. (1) is replaced by 1 whenever $|d_{qp}| < 0.05 - d_{qp}$ in the nominator keeps its real value. This exception to eq. (1) aims to reduce the apparent negative effect of large relative errors associated to target values close to zero. Even so, this trick may still lead to (relatively) large solution errors while groundbreaking results are depicted as regression plots (target vs. predicted outputs).

3.4.1 Maximum Error

This variable measures the maximum relative error, as defined by eq. (1), among all output variables and learning patterns.

3.4.2 Percentage of Errors > 3%

This variable measures the percentage of relative errors, as defined by eq. (1), among all output variables and learning patterns, that are greater than 3%.

3.4.3 Performance

In functional approximation problems, network performance is defined as the average relative error, as defined in eq. (1), among all output variables and data patterns being evaluated (e.g., training, all data).

3.5 Software Validation

Several benchmark datasets/functions were used to validate the developed software, involving low- to high-dimensional problems and small to large volumes of data. Due to paper length limit, validation results are not presented herein but they were made public by Researcher (2018). Moreover, several papers involving the successful application of this software have already been published and can be downloaded here.

3.6 Parametric Analysis Results

Aiming to reduce the computing time by cutting in the number of combos to be run – note that all features combined lead to hundreds of millions of combos, the whole parametric simulation was divided into nine parametric SAs, where in each one feature 7 only takes a single value. This measure aims to make the performance ranking of all combos within each 'small' analysis more 'reliable', since results used for comparison are based on target and output datasets as used in ANN training and yielded by the designed network, respectively (they are free of any postprocessing that eliminates output normalization effects on relative error values). Whereas (i) the 1st and 2nd SAs aimed to select the best methods from features 1, 2, 5, 8 and 13 (all combined), while adopting a single popular method for each of the remaining features (F_3: 6, F_4: 2, F_6: {1 or 7}, F_7: 1, F_9: 1, F_{10}: 1, F_{11}: {3, 9 or 11}, F_{12}: 2, F_{14}: 1, F_{15}: 1 – see Tabs. 2-4) – SA 1 involved learning algorithms 1-3 and SA 2 involved the ELM-based counterpart, (ii) the 3rd – 7th SAs combined all possible methods from features 3, 4, 6 and 7, and concerning all other features, adopted the methods integrating the best combination from the aforementioned SAs 1-2, (iii) the 8th SA combined all possible methods from features 11, 12 and 14, and concerning all other features, adopted the methods integrating the best combination (results compared after postprocessing) among the previous five sub-analyses, and lastly (iv) the 9th SA combined all possible methods from features 9, 10 and 15, and concerning all other features, adopted the methods integrating the best combination from the previous analysis. Summing up the ANN feature combinations for all parametric SAs, a total of 475 combos were run for this work.

ANN feature methods used in the best combo from each of the abovementioned nine parametric sub-analyses, are specified in Tab. 5 (the numbers represent the method number as in Tabs 2-4). Tab. 6 shows the corresponding relevant results for those combos, namely (i) maximum error, (ii) % errors > 3%, (iii) performance (all described in section 3, and evaluated for all learning data), (iv) total number of hidden nodes in the model, and (v) average computing time per example (including data pre- and post-processing). All results shown in Tab. 6 are based on target and output datasets computed in their original format, i.e. free of any transformations due to output normalization and/or dimensional analysis. The microprocessor used in this work has the

following features: OS: Win10Home 64bits, RAM: 48 GB, Local Disk Memory: 1 TB, CPU: Intel® Core™ i7 8700K @ 3.70-4.70 GHz.

Tab. 5. ANN feature (F) methods used in the best combo from each parametric sub-analysis (SA).

SA	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
1	1	2	6	2	5	7	1	1	1	1	3	2	3	1	3
2	1	2	6	2	3	7	1	1	1	1	3	2	5	1	3
3	1	2	1	1	5	3	1	1	1	1	3	2	3	1	3
4	1	2	6	2	5	1	2	1	1	1	3	2	3	1	3
5	1	2	6	3	5	1	3	1	1	1	3	2	3	1	3
6	1	2	6	3	5	7	4	1	1	1	3	2	3	1	3
7	1	2	6	4	5	7	5	1	1	1	3	2	3	1	3
8	1	2	6	4	5	7	5	1	1	1	1	5	3	1	3
9	1	2	6	4	5	7	5	1	3	3	1	5	3	1	3

Tab. 6. Performance results for the best design from each parametric sub-analysis.

SA	ANN				
	Max Error (%)	Performance All Data (%)	Errors > 3% (%)	Total Hidden Nodes	Running Time / Data Point (s)
1	24.2	0.7	5.6	36	1.88E-04
2	1375.2	21.6	83.7	120	9.96E-05
3	15.4	0.5	4.0	36	1.31E-04
4	11.7	0.5	4.0	36	1.14E-04
5	15.9	0.7	7.0	36	1.06E-04
6	12.7	0.5	3.0	36	9.58E-05
7	67.0	5.3	40.0	36	1.07E-04
8	90.0	4.0	24.0	36	1.10E-04
9	0.0	0.0	0.0	36	9.72E-05

3.7 Proposed ANN-Based Model

The proposed model is the one, among the best ones from all parametric SAs, exhibiting the lowest maximum error (SA 9). That model is characterized by the ANN feature methods {1, 2, 6, 4, 5, 7, 5, 1, 3, 3, 1, 5, 3, 1, 3} in Tabs. 2-4. Aiming to allow implementation of this model by any user, all variables/equations required for (i) data preprocessing, (ii) ANN simulation, and (iii) data

postprocessing, are presented in 3.7.1-3.7.3, respectively. The proposed model is a single MLPN with 5 layers and a distribution of nodes/layer of 9-12-12-12-1. Concerning connectivity, the network is fully-connected, and the hidden and output transfer functions are all Logistic and Identity, respectively. The network was trained using the Levenberg-Marquardt (LM) algorithm (859 epochs). After design, the average network computing time concerning the presentation of a single example (including data pre/postprocessing) is 9.72 x 10^{-5} s – Fig. 4 depicts a simplified scheme of some of network key features. Lastly, all relevant performance results concerning the proposed ANN are illustrated in 3.7.4. The obtained ANN solution for every data point can be found in Developer (2019a), making it possible to compute the exact (with all decimal figures) approximation errors.

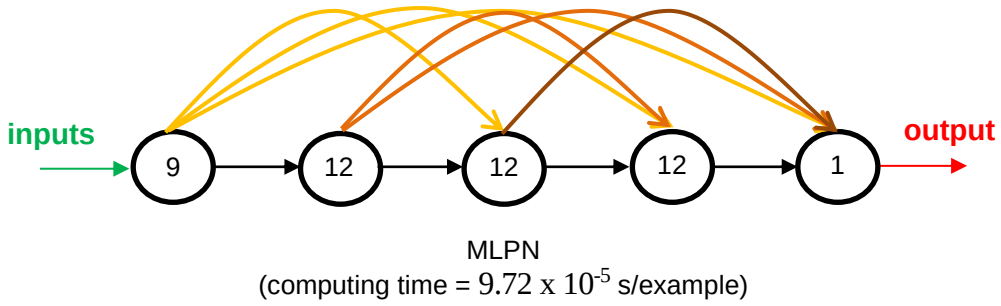

Fig. 4. Proposed 9-12-12-12-1 fully-connected MLPN – simplified scheme.

It is worth recalling that, in this manuscript, whenever a vector is added to a matrix, it means the former is to be added to all columns of the latter (valid in MATLAB).

3.7.1 Input Data Preprocessing

For future use of the proposed ANN to simulate new data $Y_{1,sim}$ (9 x P_{sim} matrix), concerning P_{sim} patterns, the same data preprocessing (if any) performed before training must be applied to the input dataset. That preprocessing is defined by the methods used for ANN features 2, 3 and 5 (respectively 2, 6 and 5 – see Tab. 2), which should be applied after all (eventual) qualitative

variables in the input dataset are converted to numerical (using feature 1's method). Next, the necessary preprocessing to be applied to $Y_{1,sim}$, concerning features 2, 3 and 5, is fully described.

Dimensional Analysis and Dimensionality Reduction

Since neither dimensional analysis (*d.a.*) nor dimensionality reduction (*d.r.*) were not carried out, one has

$$\left\{Y_{1,sim}\right\}_{d.r.}^{after} = \left\{Y_{1,sim}\right\}_{d.a.}^{after} = Y_{1,sim} \quad . \quad (2)$$

Input Normalization

After input normalization, the new input dataset $\{Y_{1,sim}\}_{n}^{after}$ is defined as function of the previously determined $\{Y_{1,sim}\}_{d.r}^{after}$, and they have the same size, reading

$$\left\{Y_{1,sim}\right\}_{n}^{after} = \left(\left\{Y_{1,sim}\right\}_{d.r}^{after} - \text{INP}(:,1)\right) ./ \text{INP}(:,2)$$

$$\text{INP} = \begin{bmatrix} 149.674651162791 & 66.7396705241561 \\ 262.705930232558 & 153.595523149765 \\ 2.53778427906977 & 0.958947876821730 \\ 0.0244412479069768 & 0.0104087109797821 \\ 480.789007209302 & 90.8692510105096 \\ 11.0945581395349 & 4.95663746228321 \\ 49.7411932558140 & 26.2694346953084 \\ 0.555063267441860 & 0.364880290571624 \\ 1261.49069767442 & 476.799170124293 \end{bmatrix} \quad , \quad (3)$$

where one recalls that operator './' divides row *i* in the numerator by INP(*i*, 2).

3.7.2 ANN-Based Analytical Model

Once determined the preprocessed input dataset $\{Y_{1,sim}\}_n^{after}$ (9 x P_{sim} matrix), the next step is to present it to the proposed ANN to obtain the predicted output dataset $\{Y_{5,sim}\}_n^{after}$ (1 x P_{sim} vector), which will be given in the same preprocessed format of the target dataset used in learning. In order to convert the predicted outputs to their 'original format' (i.e., without any transformation due to normalization or dimensional analysis – the only transformation visible will be the (eventual) qualitative variables written in their numeric representation), some postprocessing is needed, as described in detail in 3.7.3. Next, the mathematical representation of the proposed ANN is given, so that any user can implement it to determine $\{Y_{5,sim}\}_n^{after}$, thus eliminating all rumors that ANNs are 'black boxes'.

$$\begin{aligned}
Y_2 &= \varphi_2\left(W_{1-2}^T\left\{Y_{1,sim}\right\}_n^{after} + b_2\right)\\
Y_3 &= \varphi_3\left(W_{1-3}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-3}^T Y_2 + b_3\right)\\
Y_4 &= \varphi_4\left(W_{1-4}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-4}^T Y_2 + W_{3-4}^T Y_3 + b_4\right)\\
\left\{Y_{5,sim}\right\}_n^{after} &= \varphi_5\left(W_{1-5}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-5}^T Y_2 + W_{3-5}^T Y_3 + W_{4-5}^T Y_4 + b_5\right)
\end{aligned}, \quad (4)$$

where

$$\begin{aligned}
\varphi_2(s) = \varphi_3(s) = \varphi_4(s) &= \frac{1}{1+e^{-s}}\\
\varphi_5(s) &= s
\end{aligned}. \quad (5)$$

Arrays $W_{j\text{-}s}$ and b_s are stored online in Developer (2019b), aiming to avoid an overlong article and ease model's implementation by any interested reader.

3.7.3 Output Data Postprocessing

In order to transform the output dataset obtained by the proposed ANN, $\{Y_{5,sim}\}_n^{after}$ (1 x P_{sim} vector), to its original format ($Y_{5,sim}$), i.e. without the effects of dimensional analysis and/or output normalization (possibly) taken in target dataset preprocessing prior training, the postprocessing addressed next must be performed.

Non-normalized (just after dimensional analysis) and Original formats

Once obtained $\{Y_{5,sim}\}_n^{after}$, the following relations hold for its transformation to its non-normalized and original formats, respectively $\{Y_{5,sim}\}_{d.a.}^{after}$ and $Y_{5,sim}$:

$$Y_{5,sim} = \left\{Y_{5,sim}\right\}_{d.a.}^{after} = \left\{Y_{5,sim}\right\}_{n}^{after} \quad , \quad (6)$$

since neither output normalization nor dimensional analysis were carried out.

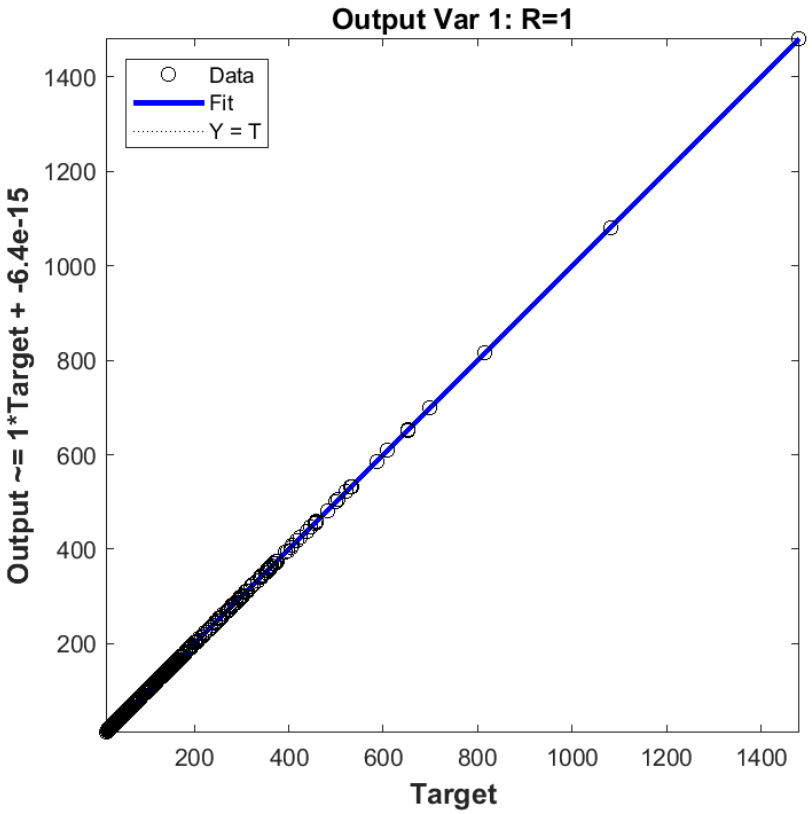

Fig. 5. Regression plot for the proposed ANN (see output variable in Fig. 1).

3.7.4 Performance Results

Finally, results yielded by the proposed ANN, in terms of performance variables defined in sub-section 3.4, are presented in this section in the form of several graphs: (i) a regression plot (Fig. 5), where network target and output data are plotted for each data point, as *x*- and *y*-coordinates respectively – a measure of linear correlation is given by the Pearson Correlation Coefficient (*R*); (ii) a performance plot (Fig. 6), where performance (average error) values are displayed for several learning datasets; and (iii) an error plot (Fig. 7), where values concern all data (iii_1) maximum error and (iii_2) % of errors greater than 3%. It´s worth highlighting that all graphical results just mentioned are based on effective target and output values, i.e. computed in their original format (free of any transformations due to output normalization and/or dimensional analysis).

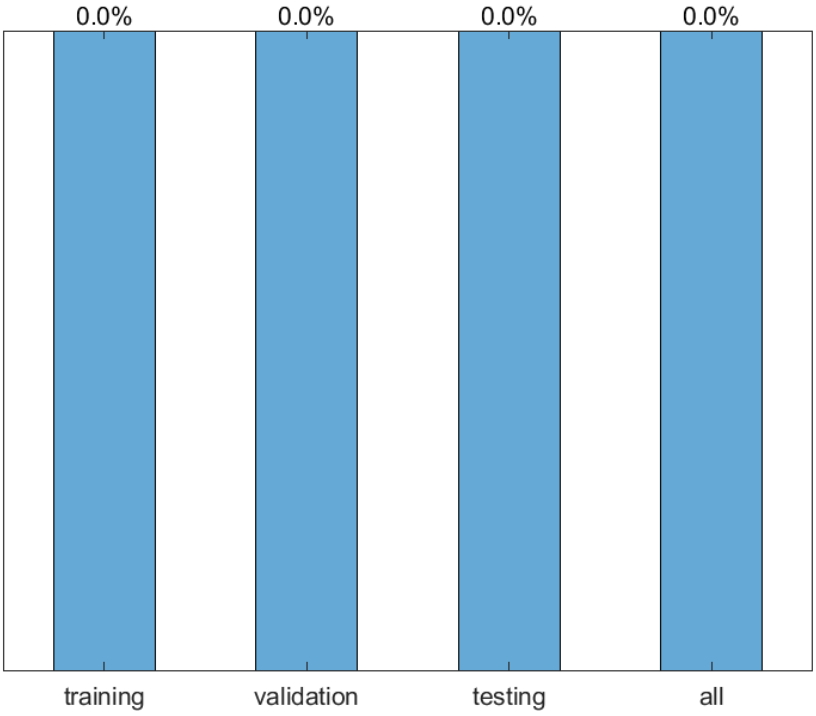

Fig. 6. Performance plot (mean errors) for the proposed ANN.

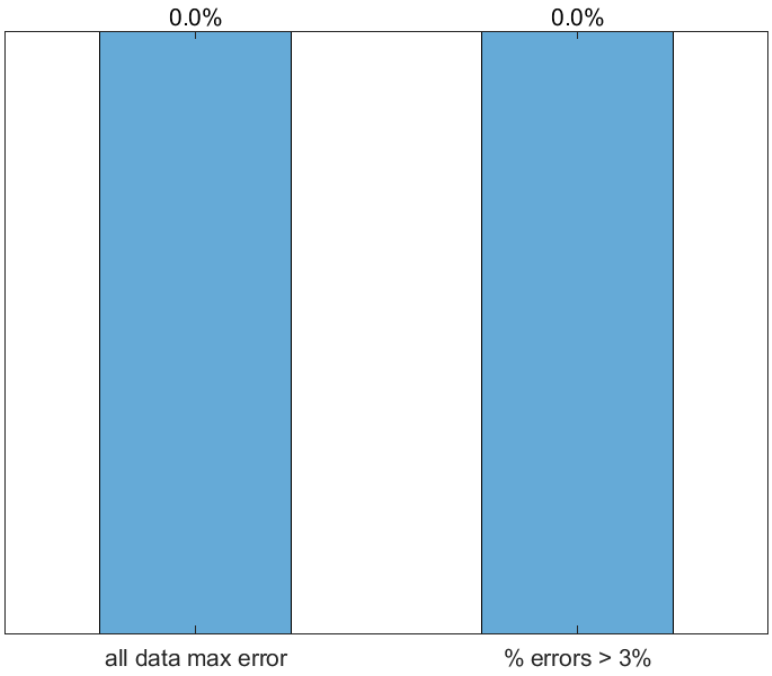

Fig. 7. Error plot for the proposed ANN.

4. ANN-based vs. Existing Models

In this section, the performance of the ANN-based model (results available in Developer 2019a) is compared against existing analytical schemes from the literature (described in Lantsoght 2019b), for all 430 examples adopted in this study. The comparison is illustrated through two figures, one for models proposed in the scientific literature by other authors (Fig.), and another for results from existing design code provisions. Tab. 7 gives the statistical results of V_{utot}/V_{pred} for all methods, where it´s possible to conclude that the ANN-based model significantly improves predictions for the 430 data points.

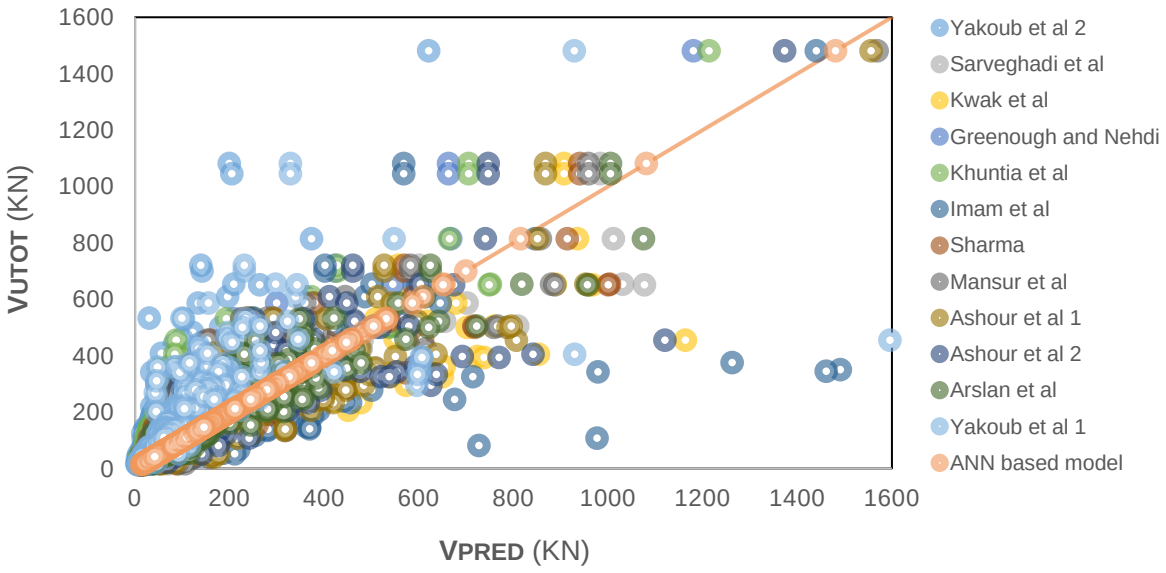

Fig. 8. Comparison (for all 430 data points) between the ANN-based model and existing formulas from the literature.

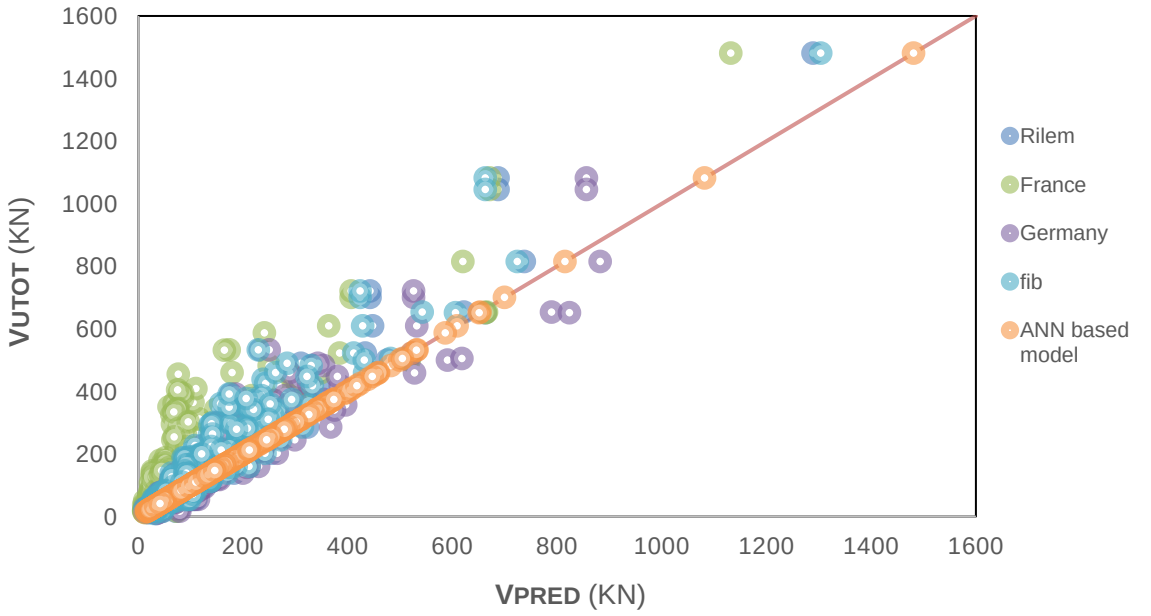

Fig. 9. Comparison (for all 430 data points) between the ANN-based model and currently available code provisions.

Tab. 7. Statistical properties of V_{utot}/V_{pred} for all data points (AVG = average, STD = standard deviation, COV = coefficient of variation).

Model	AVG	STD	COV	Min	Max
Sarveghadi et al	1.03	0.29	28%	0.23	2.49
Kwak et al	1.01	0.28	27%	0.27	2.39
Greenough and Nehdi	1.34	0.48	36%	0.31	3.11
Khuntia et al	1.81	0.85	47%	0.18	6.53
Imam et al	0.97	0.36	37%	0.06	2.51
Sharma	1.24	0.49	39%	0.18	3.59
Mansur et al	1.30	0.60	46%	0.15	3.85
Ashour et al 1	1.08	0.38	35%	0.24	3.14
Ashour et al 2	1.29	0.37	29%	0.31	3.22
Arslan et al	1.17	0.37	31%	0.43	3.24
Yakoub et al 1	1.90	0.76	40%	0.28	7.50
Yakoub et al 2	2.97	1.37	46%	0.51	17.48
French code	1.85	0.88	48%	0.22	5.95
German code	1.12	0.31	27%	0.21	2.13
fib	1.24	0.36	29%	0.30	2.33
Rilem	1.16	0.33	29%	0.23	2.28
ANN (proposed model)	1.00	1.08E-15	1.08E-15	1.00	1.00

5. Conclusions

This paper shows how artificial neural networks (ANN) can be used to predict the shear capacity of steel fiber-reinforced concrete (SFRC) beams without stirrups. For this purpose, a database of 430 test results gathered from the literature was adopted. Nine input variables were taken to describe the problem, whereas the maximum sectional shear force at collapse (including beam self-weight) was the selected target variable. After an extensive ANN-based parametric analysis, the resulting 'optimal' model yielded maximum and mean relative errors of 0.0% for all the 430 data points, which outperforms (for those 430 instances) the currently available formulas and code provisions.

One limitation of this study is that the proposed model can only be used within the variable ranges of the dataset. While it covers the practical ranges of all material properties, it does not

cover large-sized beams. As such, we recommend the performance of further tests covering the missing realistic scenarios, so that more robust and versatile data-driven analytical models (based on larger and richer datasets) can be developed. This study has not yet allowed a full description of the mechanics underlying the shear behavior of SFRC members without stirrups, but parametric studies by means of accurate and robust ANN-based models will facilitate the evaluation and improvement of existing and future mechanistic models.

Acknowledgements

This research was funded by the program of Collaboration Grants 2019 of Universidad San Francisco de Quito. There are no conflicts of interest to disclose.

Author Contributions

Abambres was in charge of section 3 (Artificial Neural Networks), and Lantsoght of all remaining sections. Both authors equally contributed to the Conclusions section.

References

Abambres M, Lantsoght E (2018). Neural network-based formula for shear capacity prediction of one-way slabs under concentrated loads, hal-02074675

Abambres M, Marcy M, Doz G (2018). Potential of Neural Networks for Structural Damage Localization, hal-02074844

Amin A, Foster SJ, Watts M (2016). Modelling the tension stiffening effect in SFR-RC. Magazine of Concrete Research, 68(7), doi: 10.1680/macr.15.00188.

Basheer I, Hajmeer M (2000). Artificial neural networks: Fundamentals, computing, design, and application, Journal of Microbiological Methods, 43(1).

Developer (2019a). SFRC dataset + target vs output [Data set], downloadable

Developer (2019b). W and b arrays [Data set], downloadable

Flood I (2008). Towards the next generation of artificial neural networks for civil engineering, Advanced Engineering Informatics, 22(1).

Greenough T, Nehdi M (2008). Shear Behavior of Fiber-Reinforced Self-Consolidating Concrete Slender Beams. ACI Materials Journal, 105(5), 468-77.

Haykin SS (2009). Neural networks and learning machines, Prentice Hall/Pearson, New York.

Hern A (2016). Google says machine learning is the future. So I tried it myself. Available at: www.theguardian.com/technology/2016/jun/28/all (Accessed: 2 November 2016).

Hertzmann A, Fleet D (2012). Machine Learning and Data Mining, Lecture Notes CSC 411/D11, Computer Science Department, University of Toronto, Canada.

Hossain KMA, Gladson LR, Anwar MS (2016). Modeling shear strength of medium- to ultra-high-strength steel fiber-reinforced concrete beams using artificial neural network. Neural Computing and Applications, 28(1), doi: 10.1007/s00521-016-2417-2.

Kara IF (2013). Empirical modeling of shear strength of steel fiber reinforced concrete beams by gene expression programming. Neural Comput and Applications, 23(3-4), doi: 10.1007/s00521-012-0999-x.

Lantsoght EOL (2019a). How do steel fibers improve the shear capacity of reinforced concrete beams without stirrups? Composites Part B: Engineering, 175(October), doi: 10.1016/j.compositesb.2019.107079.

Lantsoght EOL (2019b). Database of Shear Experiments on Steel Fiber Reinforced Concrete Beams without Stirrups. Materials, 12(6), doi: 10.3390/ma12060917.

McCulloch WS, Pitts W (1943). A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5(4).

Narayanan R, Kareem-Palanjian AS (1984). Effect of Fibre Addition on Concrete Strengths. Indian Concrete Journal, 58(4), 100-103.

Prieto A, Prieto B, Ortigosa EM, Ros E, Pelayo F, Ortega J, Rojas I (2016). Neural networks: An overview of early research, current frameworks and new challenges, Neurocomputing, 214(Nov).

Researcher, The (2018). "Annsoftwarevalidation-report.pdf", figshare, doi: 10.6084/m9.figshare.6962873.

Sarveghadi M, Gandomi AH, Bolandi H, Alavi AH (2015). Development of prediction models for shear strength of SFRCB using a machine learning approach. Neural Computing and Applications, 31(7), doi: 10.1007/s00521-015-1997-6.

Singh B, Jain K (2014). An appraisal of steel fibers as minimum shear reinforcement in concrete beams (with Appendix). ACI Structural Journal, 111(5), doi: 10.14359/51686969.

The Mathworks, Inc (2017). MATLAB R2017a, User's Guide, Natick, USA.

Torres JA, Lantsoght EOL (2019). Influence of Fiber Content on Shear Capacity of Steel Fiber Reinforced Concrete Beams. Preprints, 2019080301, doi: 10.20944/preprints201908.0301.v1.

Wilamowski BM, Irwin JD (2011). The industrial electronics handbook: Intelligent Systems, CRC Press, Boca Raton.