ANN-based Fatigue Strength of Concrete Under Compression

Miguel Abambres [1] and Lantsoght E [2]

[1]Num3ros
[2]Affiliation not available

October 30, 2023

Abstract

When concrete is subjected to cycles of compression, its strength is lower than the statically determined concrete compressive strength. This reduction is typically expressed as a function of the number of cycles. In this work, we predict the reduced capacity as function of a given number of cycles by means of artificial neural networks (ANN). A 203-point experimental dataset gathered from the literature was used. The proposed ANN model results in a maximum relative error of 5.1% and a mean counterpart of 1.2% for the whole dataset. It's shown that the proposed analytical model outperforms the existing design code expressions.

ANN-based Fatigue Strength of Concrete Under Compression

Miguel Abambres [a], Eva Lantsoght [b]

a Num3ros, 1600-275 Lisbon, Portugal; amgg@mailfence.com
Escola de Tecnologias e Engenharia, Instituto Superior de Educação e Ciências (ISEC), Lisboa, Portugal

b Politécnico, Universisad San Francisco de Quito, Ecuador; elantsoght@usfq.edu.ec
Concrete Structures, Department of Civil Engineering and Geosciences, TU Delft, The Netherlands

Important Notes:

1. The first author has been proposing ANN-based models in former publications, 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.
2. If you can't find any of my papers referred as references of this work, feel free to email me and I´ll send you the PDF right away.

Abstract

When concrete is subjected to cycles of compression, its strength is lower than the statically determined concrete compressive strength. This reduction is typically expressed as a function of the number of cycles. In this work, we predict the reduced capacity as function of a given number of cycles by means of artificial neural networks (ANN). A 203-point experimental dataset gathered from the literature was used. The proposed ANN model results in a maximum relative error of 5.1% and a mean counterpart of 1.2% for the whole dataset. It's shown that the proposed analytical model outperforms the existing design code expressions.

Keywords: Fatigue, Dataset, Compressed Concrete, Artificial Neural Networks, Design Formula.

1. Introduction

When concrete is subjected to cycles of compression, its strength is lower than the statically determined concrete compressive strength (CEB Committee GTG 15 1988, Suzuki et al. 2007). The practical implication of that behavior is the need to consider a

lower concrete compressive strength in the design of structures subjected to loading cycles, such as bridges (e.g., Lantsoght et al. 2019a, b, c). The most fundamental approach to study fatigue isolates the different material contributions in the cross-section (Blasón et al 2019, Tilly 1979). As such, the effect of fatigue on a concrete cross-section under compression is studied by testing concrete cylinders, from different section parts, under cyclic loading (Bazant and Hubler 2014, Bennett and Muir 1967).

In a classic fatigue test of a concrete specimen (most often a cylinder) under compression, the load is applied as a sine wave between a fixed lower and upper value. These loads induce stresses in the concrete specimen that fluctuate between $S_{min}\ f_c$ and $S_{max}\ f_c$ ($0 \leq S_{min}, S_{max} \leq 1$). The focus of this work is only on constant amplitude loading. When the stress ratios S_{min} and S_{max} are chosen as the input values for an experiment, the outcome of the experiment then is the number of cycles to failure (N). For the design of a new structure, we usually have the number of cycles the structure needs to withstand (for example, N = 1 000 000). Thus, the design or assessment is based on the reduced strength associated with that number of cycles. That said, the number of cycles N is one of the input (independent) variables in this work, being the strength ratio (S_{max}) the sole output variable.

2. Data Gathering

The dataset used in this work was constructed from the database published by Lantsoght et al (2016). To have unique input data points, the geometric average of the number of cycles (N) was employed in cases of repeated tests. Furthermore, (i) experimental results on ultra-high-performance concrete were excluded, since it could not be ensured a good continuum of input values of the concrete compressive strength, and (ii) experiments on heat-treated specimens were left out because Lohaus and Anders (2006) reported that their fatigue performance was different from the regular specimen counterpart.

Tab. 1 describes the independent and dependent variables considered in the dataset, being illustrated in Fig. 1. These are the most important parameters to consider when testing concrete specimens under fatigue compression (CEB

Committee GTG 15 1988). The influence of testing frequency on the fatigue life is still a topic of discussion (e.g., CEB Committee GTG 15 1988, Hsu 1981, Lohaus and Anders 2006). Moreover, the authors wanted to propose a model that depends on the same parameters as the design code counterpart.

The 203-point dataset used for ANN simulations is available in Developer (2019a). The reader should keep in mind that the proposed ANN model (to be described in section 3) is only valid within the range of values of each input variable in the dataset, which are shown in Tab. 1.

Tab. 1. Independent (input) and dependent (output or target) variables in the dataset, including ranges of values.

VARIABLES			ANN NODE NUMBER	MIN	MAX
Material	$f_{c,cyl}$ (MPa)	average concrete compressive strength (statically determined)	1	24.03	170
Loading	S_{min} (-)	minimum stress ratio	2	0	0.836
	N (-)	number of cycles to failure	3	3	63 841 046.87
Output	S_{max} (-)	strength ratio or maximum stress ratio	1	0.465	0.960

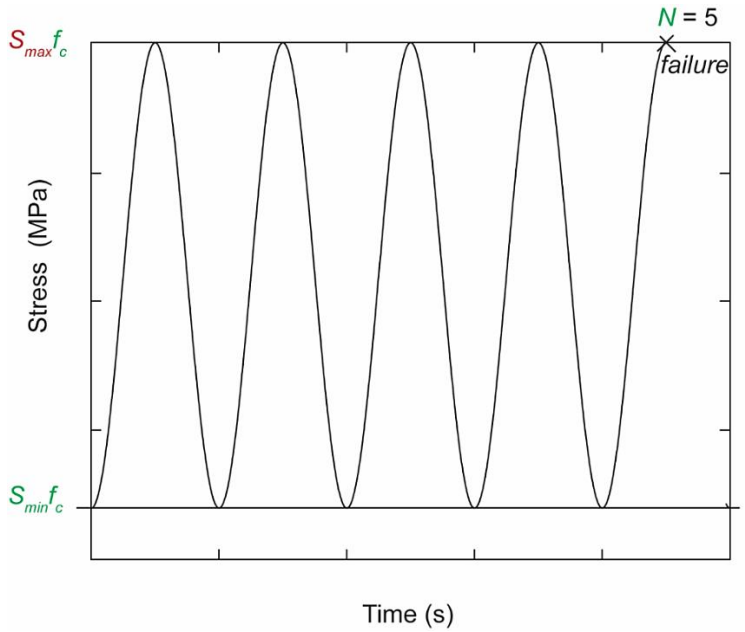

Fig. 1. Input and output variables, shown on an example of an experimental loading scheme.

3. Artificial Neural Networks

3.1 Brief Introduction

Machine learning (ML), one of the six disciplines of Artificial Intelligence (AI), allows us to 'teach' computers how to perform tasks by providing examples of how they should be done (Hertzmann and Fleet 2012). The Artificial Neural Network (also referred in this manuscript as ANN or neural net) is ML's (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). An ANN is a mathematical model inspired by the way a brain processes information, i.e. with the help of its processing units (the neurons). ANNs have been employed to perform several types of tasks. Concerning nonlinear regression, the task adopted in this work, 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).

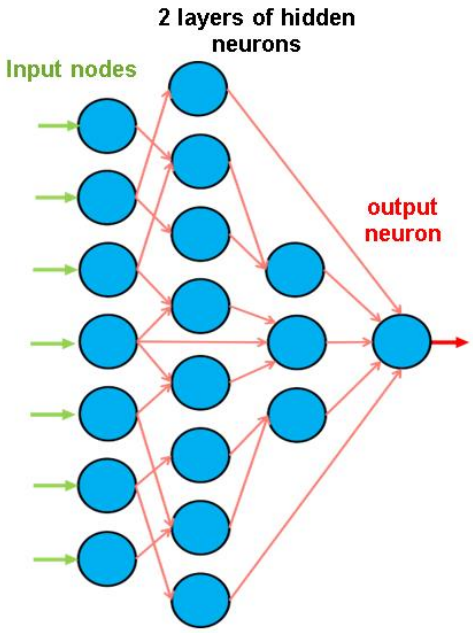

Fig. 2. Example of a feedforward neural network.

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. 2. 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, as exemplified in Fig. 2.

Further information on Artificial Neural Networks might be found in previous publications or 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. 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), 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.

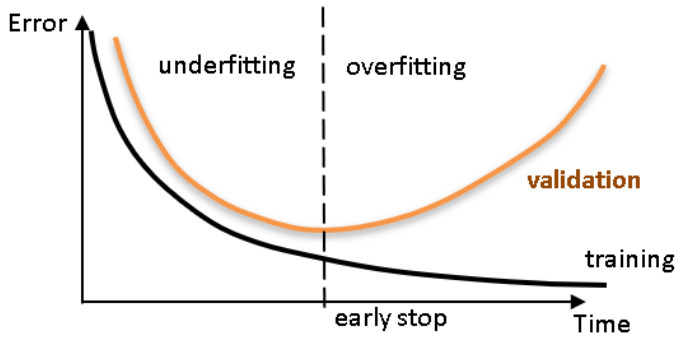

Fig. 3. Cross-validation - assessing network's generalization ability.

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. 2018), 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	-	-	-	-

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.

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	-	-	-	-

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. Validation results are not presented herein but they were made public in Researcher (2018). Moreover, several papers involving the successful application of this software have already been published and can be downloaded here (a centralized platform) or here (a decentralized platform – you should try it, according to this).

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 219 combos were run for this work.

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	F15
1	1	2	6	2	5	1	1	1	1	1	3	2	3	3
2	1	2	6	2	6	7	1	1	1	1	3	2	7	3
3	1	2	6	1	5	1	1	1	1	1	3	2	3	3
4	1	2	6	3	5	1	2	1	1	1	3	2	3	3
5	1	2	6	2	5	1	3	1	1	1	3	2	3	3
6	1	2	6	1	5	7	4	1	1	1	3	2	3	3
7	1	2	6	3	5	7	5	1	1	1	3	2	3	3
8	1	2	6	3	5	7	5	1	1	1	3	2	3	3
9	1	2	6	3	5	7	5	1	3	3	3	2	3	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	18.2	3.0	39.9	12	2.67E-04
2	51.2	10.7	79.8	53	4.70E-05
3	21.2	2.9	35.5	12	8.39E-05
4	18.1	3.0	37.9	12	5.06E-05
5	21.5	3.0	39.4	12	6.13E-05
6	21.3	2.9	36.5	12	4.10E-05
7	23.1	2.9	39.4	12	4.40E-05
8	23.0	3.0	36.0	12	6.95E-05
9	5.1	1.2	10.3	12	7.09E-05

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.

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, 3, 5, 7, 5, 1, 3, 3, 3, 2, 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 MLPN with 5 layers and a distribution of nodes/layer of 3-4-4-4-1. Concerning connectivity, the network is fully-connected, and the hidden and output transfer functions are all Hyperbolic Tangent and Identity, respectively. The network was trained using the Levenberg-Marquardt (LM) algorithm (1500 epochs). After design, the average network computing time concerning the presentation of a single example (including data pre/postprocessing) is 7.09×10^{-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.

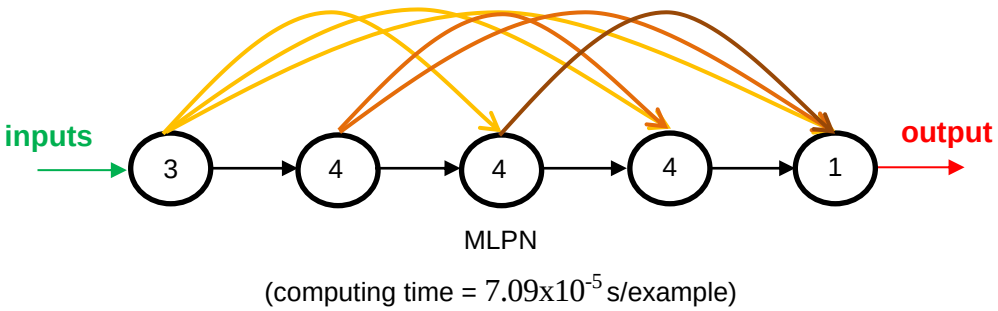

Fig. 4. Proposed 3-4-4-4-1 fully-connected MLPN – simplified scheme.

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 $Y_{1,sim}$ (3 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 no dimensional analysis (*d.a.*) nor dimensionality reduction (*d.r.*) were 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

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

$$\mathrm{INP} = \begin{bmatrix} 70.7417339901478 & 44.7805334345334 \\ 0.206873201970443 & 0.192556137441602 \\ 1122471.30034975 & 5728851.48603107 \end{bmatrix} \quad , \quad (3)$$

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 $\{Y_{1,sim}\}_n^{after}$ (3 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

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

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 and Original formats

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

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

since no output normalization nor dimensional analysis were carried out.

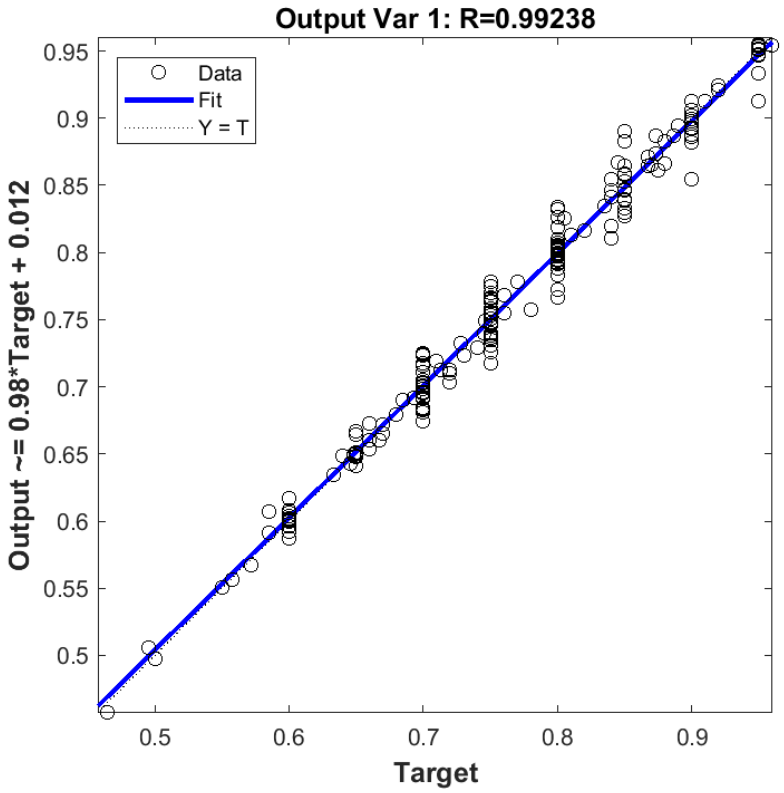

Fig. 5. Regression plot for the proposed ANN (see output variable in Fig. 1) – one should bear in mind that the lower value of both axis is not null.

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.

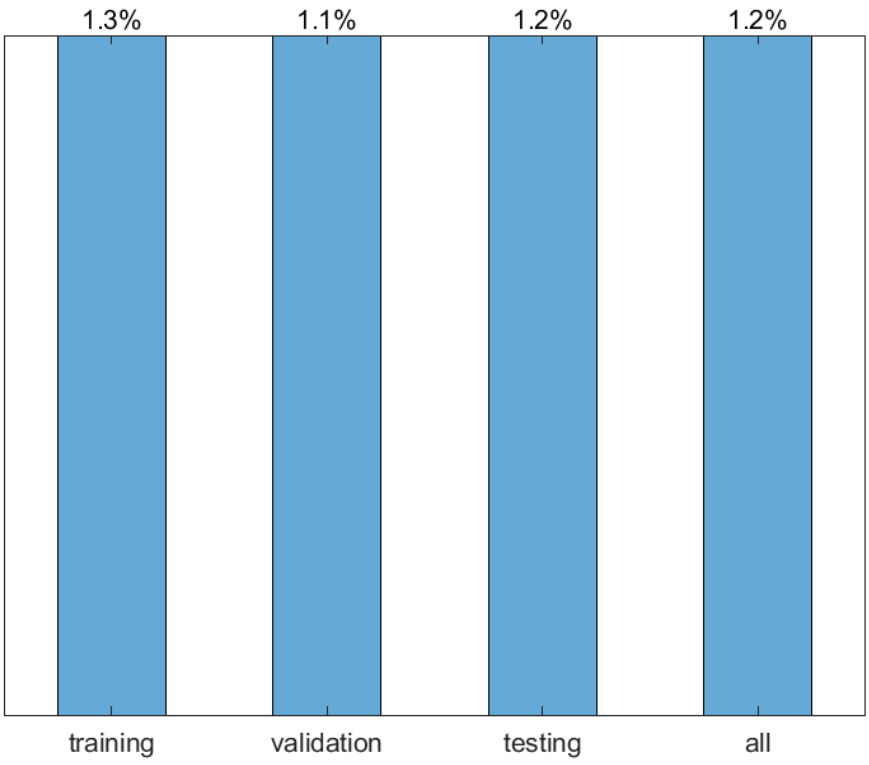

Fig. 6. Performance plot (mean errors) for the proposed ANN.

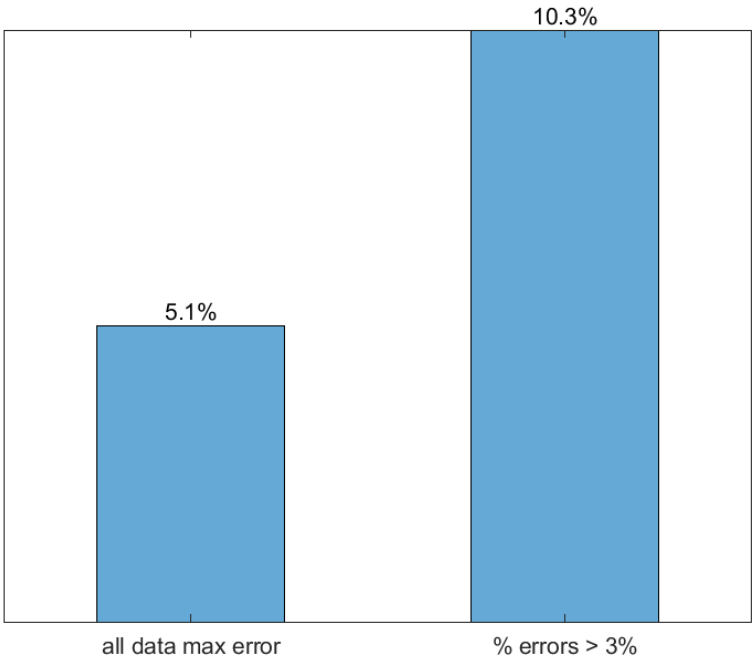

Fig. 7. Error plot for the proposed ANN.

4. ANN-based vs. Design Code Models

Tab. 1 in URL (peer-reviewed version of this work – worst in Abambres' opinion, and not fully reviewed by that author) gives an overview of some currently and formerly used design code equations to assess the fatigue life of concrete under compression. Some of these models are used next for performance comparison against the proposed ANN in predicting S_{max}. The 203 experimental data points were used for that purpose – they are available in Developer (2019a), along with the ANN predictions.

Fig. 1 shows the comparison between the experimental and predicted values – the following three codes were considered: (i) NEN 6723 (Code Committee 351-001 "Technical Foundations for Structures" 2009), (ii) NEN-EN 1992-2+C1 (CEN 2011), and (iii) the Model Code 2010 (fib 2012). Details on how the variables in those code equations were computed are available on URL. We can see that a few code predictions yield negative S_{max}, which is physically impossible.

Tab. 7 gives the statistical properties of the experimental-to-predicted S_{max} ratios for all the 203 data points, concerning all predictive models assessed. Besides Fig. 8, Tab. 7 makes it clear that the proposed ANN is the most effective analytical model for the 203 examples used in this research (in second place comes the Model Code 2010).

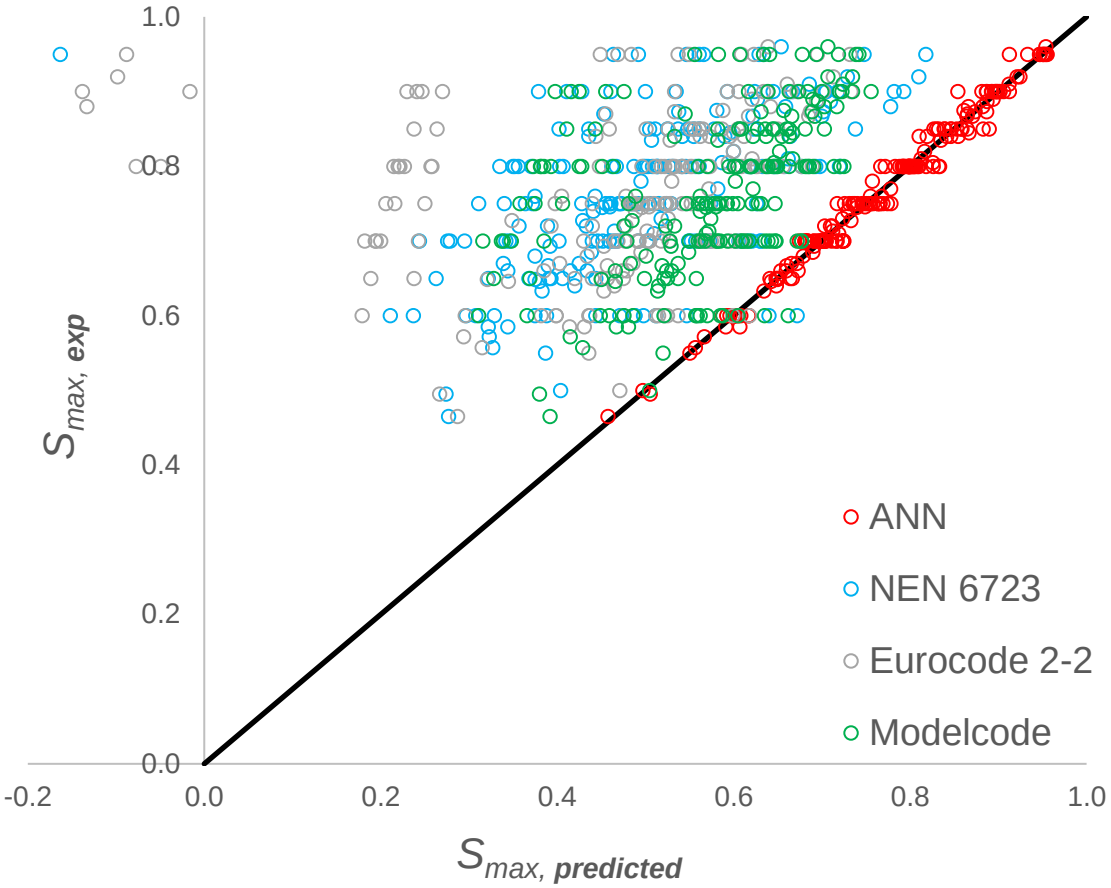

Fig. 1. Comparison between experimental and predicted values for the 203 examples.

Tab. 7. Statistics of $S_{max,\,exp}$ / $S_{max,\,p}$ for all the 203 data points (AVG = average, STD = standard deviation, COV = coefficient of variation).

Model	AVG	STD	COV	Min	Max
ANN	1.0	0.02	1.69%	0.96	1.05
NEN 6723	1.6	0.63	40.53%	-5.83	2.87
Eurocode 2-2	1.1	4.59	430.61%	-56.25	3.91
Model Code 2010	1.4	0.28	20.46%	0.91	2.26

5. Conslusions

- A 203-point experimental dataset gathered from the literature was used to develop an analytical model that predicts the (reduced) compressive strength of concrete under fatigue compression.
- The proposed “optimum” model was found through extensive parametric artificial neural network simulations, resulting in a maximum relative error of 5.1% and a mean counterpart of 1.2% for all the 203 data points.
- It’s shown that the proposed ANN outperforms the three design code equations used for comparison purposes: (i) NEN 6723, (ii) NEN-EN 1992-2+C1 (Eurocode 2-2), and (iii) the Model Code 2010 – Fig. 8 and Tab. 7 (shown just above) make it clear.
- The computational time of the proposed model is tinny – 0.07 milliseconds per data point.
- Further experimental results, on high-cycle fatigue, are necessary to broad the scope of the proposed model. Given the required time for such experiments, perhaps highly robust numerical analyses can be used to generate data for N > 64 M cycles (e.g., up to 250 or 500 million).

Acknowledgements

To Universidad San Francisco de Quito (Ecuador) for its funding through *Collaboration Grants 2019.*

Author Contributions

Abambres was in charge of section 3 (Artificial Neural Networks) and was the editor of this eprint; Lantsoght was in charge of all sections but section 3. 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

Bazant ZP, Hubler MH (2014). Theory of cyclic creep of concrete based on Paris law for fatigue growth of subcritical microcracks. Journal of the Mechanics and Physics of Solids, 63, 187-200.

Bennett EW, Muir SESJ (1967). Some fatigue tests of high-strength concrete in axial compression. Magazine of Concrete Research, 19(59), 113-117.

Blasón S, Poveda E, Ruiz G, Cifuentes H, Fernández Canteli A (2019). Two-fold normalization of the cyclic creep curve of plain and steel-fiber reinforced concrete and its application to predict fatigue failure. International Journal of Fatigue, 120, 215-227.

CEB Committee GTG 15 (1988). Fatigue of Concrete Structures, V. CEB Bulletin d' Information No 188.

Code Committee 351-001 "Technical Foundations for Structures" (2009). Regulations for concrete – Bridges – Structural requirements and calculation methods (NEN 6723).

Comité Européen de Normalisation (CEN) (2011). Eurocode 2: Design of concrete structures - Concrete bridges – Design and detailing rules (NEN-EN 1992-2+C1), Brussels, Belgium.

Developer (2019a). dataset + tabled results [Data set], downloadable

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

fib (2012). Model Code 2010: final draft. International Federation for Structural Concrete, Lausanne, Switzerland.

Flood I (2008). Towards the next generation of artificial neural networks for civil engineering, Advanced Engineering Informatics, 22(1), 4-14.

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.

Hsu TTC (1981). Fatigue of plain concrete. Journal of the American Concrete Institute, 78(4), 292-305.

Lantsoght EOL, Koekkoek R, van der Veen C, Sliedrecht H (2019a). Fatigue Assessment of Prestressed Concrete Slab-Between-Girder Bridges. Applied Sciences, 9(11).

Lantsoght EOL, van der Veen C, de Boer A (2016). Proposal for the fatigue strength of concrete under cycles of compression. Construction and Building Materials, 107(15), 138-156.

Lantsoght EOL, van der Veen C, Koekkoek R, Sliedrecht H (2019b). Fatigue testing of transversely prestressed concrete decks. ACI Structural Journal, 116(4), 143-154.

Lantsoght EOL, van der Veen C, Koekkoek R, Sliedrecht H (2019c). Punching capacity of prestressed concrete bridge decks under fatigue. ACI Structural Journal, 116(4), 209-2018.

Lohaus L, Anders S (2006). High-Cycle Fatigue of Ultra-High-Performance Concrete – Fatigue Strength and Damage Development. Proceedings of the 2nd International Congress, Naples, Italy.

McCulloch WS, Pitts W (1943). A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5(4), 115–133.

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(November), 242-268.

Researcher T (2018). ANNSoftwareValidation-report.pdf, 10.6084/m9.figshare.6962873.

Suzuki T, Ohtsu M, Shigeishi M (2007). Relative damage evaluation of concrete in a road bridge by AE rate-process analysis, Materials and Structures, 40, pp. 221-227.

The Mathworks, Inc (2018). MATLAB R2018a, User's Guide, Natick, USA.

Tilly GP (1979). Fatigue of steel reinforcement bars in concrete: a review. Fatigue & Fracture of Engineering Materials & Structures, 2(3), 251-268.

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