DOI: http://doi.org/10.31224/osf.io/wg7hd

Neural Network-based formula for the buckling load prediction of I-section cellular steel beams

Miguel Abambres [a], Komal Rajana [b], Konstantinos Tsavdaridis [b], Tiago Ribeiro [c]

[a] *Research & Development, Abambres' Lab, 1600-275 Lisbon, Portugal; abambres@netcabo.pt*
[b] *School of Civil Engineering, University of Leeds, LS2 9JT, Leeds, United Kingdom*
[c] *Tal Projecto, Lisbon, Portugal*

Abstract

Cellular beams are an attractive option for the steel construction industry due to their versatility in terms of strength, size, and weight. Further benefits are the integration of services thereby reducing ceiling-to-floor depth (thus, building's height), which has a great economic impact. Moreover, the complex localised and global failures characterizing those members have led several scientists to focus their research on the development of more efficient design guidelines. This paper aims to propose an artificial neural network (ANN)-based formula to estimate the critical elastic buckling load of simply supported cellular beams under uniformly distributed vertical loads. The 3645-point dataset used in ANN design was obtained from an extensive parametric finite element analysis performed in ABAQUS. The independent variables adopted as ANN inputs are the following: beam's length, opening diameter, web-post width, cross-section height, web thickness, flange width, flange thickness, and the distance between the last opening edge and the end support. The proposed model shows a strong potential as an effective design tool. The maximum and average relative errors among the 3645 data points were found to be 3.7% and 0.4%, respectively, whereas the average computing time per data point is smaller than a millisecond for any current personal computer.

Keywords: Elastic Buckling, Cellular Steel Beams, FEA, Artificial Neural Networks, Design Formula

1. Introduction

The use of cellular beams (i.e., perforated beams with circular web openings) in the construction sector has significantly increased over the past decade on account of the distinct

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

and discreet advantages they offer. Cellular beams are applicable for long span structures, where integration of services such as ventilation ducts and lightings systems within the beam is attained, but also for short spans, where spatial interference among concentrated mechanical devices and structural elements may require a compromised solution. Cellular beams allow reducing the height of buildings to fit a required number of floors, otherwise fitting more floors in a given height limit having a significant economic impact to the whole structure's budget. Furthermore, cellular beams offer practical advantages such as the possibility of (i) fixing the ceilings directly to the beams' lower flanges instead of requiring additional suspension elements, and (ii) allowing future equipment addition or replacement within the existent void holes. In fact, with the wider adoption of Building Information Modelling (BIM), the knowledge of those expansion possibilities is becoming a valuable asset for the building management.

Long span and lightweight structures also benefit from flexible designs with the fewer number of columns and foundations, and thus from the reduced construction time (Tsavdaridis 2010). The increase in beam depth due to the castellation process (i.e., profile cutting manufacturing) also provides greater flexural stiffness having a final section with larger section modulus (Morkhade and Gupta 2015). However, the presence of web openings significantly influences the structural performance of the beams, which in particular is dependent on the geometry (shape, diameter, and critical opening length), location (shear-moment interaction), and spacing (closely and widely spaced) between perforations. The perforations lead to complex structural behaviours, attributed to the distribution of forces and stresses in the vicinity

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

of the openings. This results in rather complicated and conservative design procedures (Morkhade and Gupta 2015, Akrami and Erfani 2016).

The first analytical model proposed by Uenoya and Redwood (1978) provided critical baseline data relating to the behaviour of perforated beams by studying the in-plane stress distribution pattern using perforated plates. Later, Lucas and Darwin (1990) proposed a design process based on the identification of the maximum bending and shear capacities at the web openings. It was afterwards suggested that the nominal capacities for the combinations of the bending moment and shear at each opening were determined. This method was accepted by the AISC (Darwin, 1990) and the ASCE 23-97 (SEI/ASCE, 1998). However, the method only provided a reasonable accurate load estimate for beams with small height, whereas for greater heights becomes conservative. The method is also restrictive to a maximum opening height of 0.7h, since the average errors were found to increase significantly above this range. Also, in 1990, Ward proposed a simplified semi-empirical web-post model using finite element modelling; however, this model was restrictive as it was based on a limited number of geometric configurations and best results were found with an error of 30%. Following, Chung et al. (2001) studied the Vierendeel mechanism and derived moment-shear interaction curves for various types of perforated beams which ultimately led to the development of a generalised moment-shear curve to assess the load carrying capacities of beams with various openings. In 2003, Chung at al. further analysed this moment-shear interaction curve and concluded that different shapes and sizes of openings can affect a beam differently and that this curve is more relevant for beams with large openings. In the same year, Chung et al. reported that the

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

moment-shear curve can be somewhat conservative when analysing beams with small web openings. However, Chung et al.'s findings were identified as a relatively good method as it provided a good approximation with mean errors of 15-25% (for various shapes of web openings), where cellular beams indicated an error of 15.8%. Since 2009, Tsavdaridis et al. published studies (Tsavdaridis and D'Mello 2009; Tsavdaridis 2010; Tsavdaridis and D'Mello 2011; Tsavdaridis and D'Mello 2012; Tsavdaridis et al. 2015) on thin-walled perforated beams with circular and other novel non-standard web openings investigating the web-post buckling failure mode for closely spaced web openings as well as the Vierendeel mechanism when large isolated web openings. The studies revealed that the Vierendeel bending is influenced by both the shape and size of an opening and the load carrying capacity of beams with large web openings can be found by examining the formation (the order and position) of plastic hinges. The extreme opening diameter (0.8h) as opposed to the maximum value of 0.75h presented in earlier literature, was introduced by Tsavdaridis and D'Mello (2012) in order to comprehensively develop an understanding of the parameters which affect the structural behaviour of perforated beams. Akrami and Erfani (2016), in a comparative analysis of the design methodologies for perforated beams, found that the works of Chung et al (2003) and Tsavdaridis and D'Mello (2012) were less restrictive as compared to other design methods (ASCE 23-97, SCI-P100, SCI-P355) and produce the lowest errors. In 2011, Lawson and Hicks published the SCI-P355 (Lawson and Hicks 2011) design guidelines, an update to SCI-P068 (Lawson 1987) and SCI-P100 (Ward 1990) which proposed that the Vierendeel bending resistance is dependent on the classification of the web of the T-beams. This approach produced

DOI: http://doi.org/10.31224/osf.io/wg7hd

acceptable approximations for openings of specific dimensions where the best results were found with an error of 25-30%. It is worth noting that Chung et al. (2003), Verweij (2010) and Morkhade and Gupta (2015) have reported that the current guidelines, specifically SCI-P100 (Ward 1990) and SCI-P355 (Lawson and Hicks 2011), are inadequate, complicated and conservative when it comes to the design of perforated steel beams.

Artificial Neural Networks (ANN) have become a popular method to predict the response of structures. Gholizadeh et al. (2011) presented a study relating the use of ANN in the evaluation of the load carrying capacity of the web-post of castellated steam beams based on 140 FE models. The computational technique generated predictions with great accuracy when compared to other methods. Sharifi and Tohidi (2014) also illustrated the application of ANN to accurately estimate the elastic buckling load capacity of steel bridge girders that have rectangular openings at the bottom zone in the web. This is considered as the worse possible location to place an opening to resist lateral torsional buckling. The ANN formula was derived from 21 FE models which managed to accurately predict the elastic buckling load. In 2014, Tohidi and Sharifi demonstrated the versatility of ANN by studying the buckling capacity of steel beams with rectangular web openings that has experienced corrosion in the web. In addition, Tohidi and Sharifi (2015) developed an ANN model to estimate the bearing capacity of steel girders with corrosion at the bearing region. The ANN empirical formulas obtained were reported to be accurate in predicting the residual capacity of deteriorated steel beams.

The current study was motivated by the lack of rational (simple, efficient and accurate) design procedures relating to the buckling response of cellular beams. This paper proposes an

DOI: http://doi.org/10.31224/osf.io/wg7hd

ANN-based formula to estimate the critical elastic buckling load of simply supported cellular beams under uniformly distributed vertical loads, as function of eight independent geometrical parameters. This research is the first step of an ongoing investigation that aims to propose a novel and simple analytical design method to accurately compute the inelastic resistance of cellular steel beams. A FE-based dataset comprising 3645 points was generated for this study, in order to allow the ANN model to have a significant generalization ability and be considered as a powerful tool for structural engineers and scientists to (i) estimate the elastic buckling load of cellular steel beams, and (ii) efficiently perform sensitivity analyses to further assess the behaviour of those members.

2. Data Generation

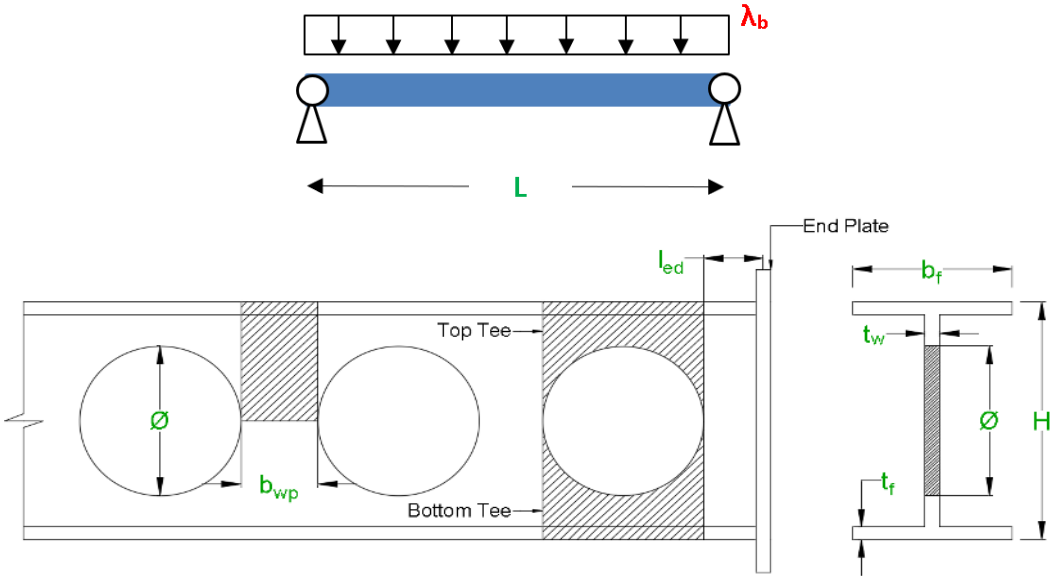
2.1 FE Modelling

Three-dimensional FE models were developed using ABAQUS (Dassault Systèmes Simulia Corp, 2017), which were then parametrised to generated 3645 simulations. Typical values for the modulus of elasticity and Poisson's ratio were adopted (E = 210 GPa, ν=0.3). All models are simply supported where one end allows in-plane rotations but not translations and the other admits translations along the beam axis, beyond in-plane rotations. End twisting rotations were prevented by restraining both the top and bottom flange tips against out-of-plane displacements at the supports. A unitary load was applied to the top flange as a uniformly distributed pressure (then converted to a line load for ANN simulation purposes – see Tab. 1). The FE mesh adopted was quad-dominated using shell elements of type SR8, which was tested against experimental

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

work conducted by Tsavdaridis and D'Mello (2011), and Surtees and Liu (1995), providing accurate and reliable results (Rajana 2018). The mesh sizes recommended by El-Sawy et al. (2014) for web and flanges were adopted. Fig. 1(a) illustrates the various parameters considered in the parametric analysis, whereas Fig. 1(b) illustrates one application of these type of structural members.



(a)

(b)

Fig. 1. Steel Cellular Members: **(a)** input (parametric) variables, and **(b)** application in an office building floor system in Lisbon, Portugal.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

2.2 Parametric Analysis

The parametric models were submitted to the ABAQUS Lanczos Eigensolver using Python scripts. Tab. 1 presents the possible values taken for each independent (parametric) variable (see Fig. 1(a)) considered in the FEA. The 'first' web opening was placed at the centre of the beam whereas the remaining ones were offset from the former until (for a fixed beam's length, opening diameter, and web-post width) no more circular openings could fit within member's length. This approach resulted in 135 different distances from the end opening edge to the centre-line of the endplate (distance named 'opening-support end distance' in Tab. 1). Thus, combining all values of variables 1 and 3 to 8, presented in Tab. 1, one has (3^6) x 5 lengths = 3645 distinct steel beams for FEA (also called data points or examples in this manuscript). The 3645-point dataset considered in ANN simulations is available in Developer (2018a).

Tab. 1. Variables, values and units employed in the parametric FEA and the ANN model.

Inputs Variables - Fig. 1(a)		ANN Node No.	Possible Values				
Beam's length	L (**m**)	1	4	5	6	7	8
Opening-support end distance	l_{ed} (**mm**)	2	135 values in [12, 718]				
Opening diameter	Φ (**mm**)	3	H/1.25	H/1.5	H/1.7	-	-
Web-post width	b_{wp} (**mm**)	4	Φ/10	Φ/3.45	Φ/2.04	-	-
Section height	H (**mm**)	5	700	560	420	-	-
Web thickness	t_w (**mm**)	6	15	12	9	-	-
Flange width	b_f (**mm**)	7	270	216	162	-	-
Flange thickness	t_f (**mm**)	8	25	20	15	-	-
Target / Output Variable							
Elastic Buckling Load		λ_b (**KN/m**)					

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

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). When there is abundant data (also called examples or patterns) explaining a certain phenomenon, but its theory richness is poor, machine learning can be a perfect tool. 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. 2.

DOI: http://doi.org/10.31224/osf.io/wg7hd

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 (see Fig. 5). 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. 2. ANN's computing power makes them suitable to efficiently solve small to large-scale complex problems, which can be attributed to their (i) massively parallel distributed structure and (ii) ability to learn and generalize, i.e., produce reasonably accurate outputs for inputs not used during the learning (also called training) phase.

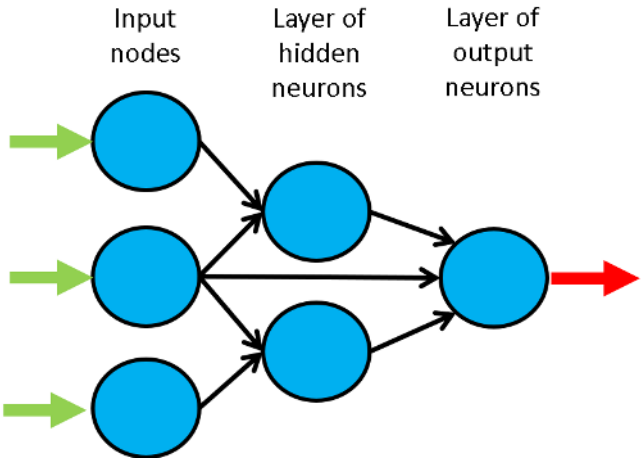


Fig. 2. Example of a feedforward neural network.

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

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

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), 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).

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3.3 Implemented ANN features

The 'behavior' of any ANN depends on many 'features', having been considered 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

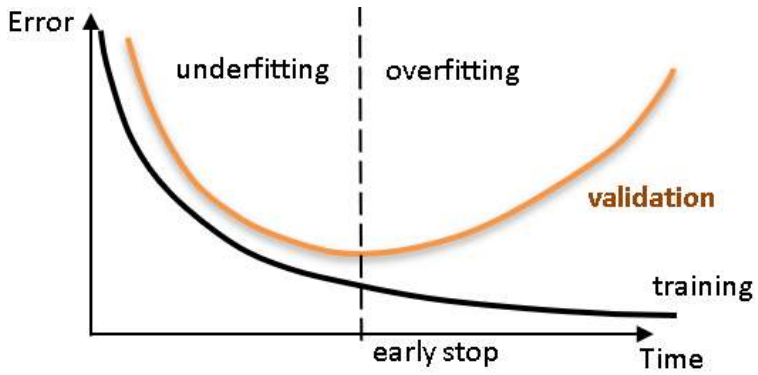


Fig. 3. Cross-validation - assessing network's generalization ability.

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 implemented methods, selected from state of art literature on ANNs (including both traditional and promising modern techniques), is presented next – Tabs. 2-4 present all features and methods per feature. 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

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

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.

It is worth highlighting 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).

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

3.3.1 Qualitative Variable Representation (feature 1)

A qualitative variable taking n distinct 'values' (usually called classes) can be represented in any of the following formats: one variable taking n equally spaced values in]0,1], or 1-of-n encoding (boolean vectors – e.g., n=3: [1 0 0] represents class 1, [0 1 0] represents class 2, and [0 0 1] represents class 3). After transformation, qualitative variables are placed at the end of the corresponding (input or output) dataset, in the same original order.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3.3.2 Dimensional Analysis (feature 2)

The most widely used form of dimensional analysis is the Buckingham's π-theorem, which was implemented in this work as described in Bhaskar and Nigam (1990).

3.3.3 Input Dimensionality Reduction (feature 3)

When designing any ANN, it is crucial for its accuracy that the input variables are independent and relevant to the problem (Gholizadeh et al. 2011, Kasun et al. 2016). There are two types of dimensionality reduction, namely (i) feature selection (a subset of the original set of input variables is used), and (ii) feature extraction (transformation of initial variables into a smaller set). In this work, dimensionality reduction is never performed when the number of input variables is less than six. The implemented methods are described next.

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	-	-	-	-

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

Linear Correlation

In this feature selection method, all possible pairs of input variables are assessed with respect to their linear dependence, by means of the Pearson correlation coefficient R_{XY}, where X and Y denote any two distinct input variables. For a set of n data points (x_i, y_i), the Pearson correlation reads

$$R_{XY} = \frac{\sum_{i=1}^{n}\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i-\bar{x}\right)^2\sum_{i=1}^{n}\left(y_i-\bar{y}\right)^2}} = \frac{Cov(X,Y)}{\sqrt{Var(X)\,Var(Y)}} \quad , \quad (1)$$

where (i) $Var(X)$ and $Cov(X, Y)$ are the variance of X and covariance of X and Y, respectively, and (ii) $\bar{x}$ and $\bar{y}$ are the mean values of each variable. In this work, cases where $|R_{XY}| \geq 0.99$ indicate that one of the variables in the pair must be removed from the ANN modelling. The one to be removed is the one appearing less in the remaining pairs (X, Y) where $|R_{XY}| \geq 0.99$. Once a variable is selected for removal, all pairs (X, Y) involving it must be disregarded in the subsequent steps for variable removal.

Auto-Encoder

This feature extraction technique uses itself a 3-layer feedforward ANN called auto-encoder (AE). After training, the hidden layer output (y_{2p}) for the presentation of each problem's input pattern (y_{1p}) is a compressed vector (Q_2 x 1) that can be used to replace the original input layer by a (much) smaller one, thus reducing the size of the ANN model. In this work, Q_2=*round*(Q_1/2) was adopted, being *round* a function that rounds the argument to the nearest integer. The implemented

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

AE was trained using the 'trainAutoencoder(…)' function from MATLAB's neural net toolbox. In order to select the best AE, 40 AEs were simulated, and their performance compared by means of the performance variable defined in sub-section 3.4. Each AE considered distinct (random) initialization parameters, half of the models used the 'logsig' hidden transfer functions, and the other half used the 'satlin' counterpart, being the identity function the common option for the output activation. In each AE, the maximum number of epochs – number of times the whole training dataset is presented to the network during learning, was defined (regardless the amount of data) by

$$\max epochs = \begin{cases} 3000, Q_1 > 8 \\ 1500, Q_1 \leq 8 \end{cases} . \quad (2)$$

Concerning the learning algorithm used for all AEs, no L_2 weight regularization was employed, which was the only default specification not adopted in 'trainAutoencoder(…)'.

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	NNC	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	-	-	-	-

DOI: http://doi.org/10.31224/osf.io/wg7hd

Orthogonal and Sparse Random Projections

This is another feature extraction technique aiming to reduce the dimension of input data Y_1 (Q_1 x P) while retaining the Euclidean distance between data points in the new feature space. This is attained by projecting all data along the (i) orthogonal or (ii) sparse random matrix A (Q_1 x Q_2, $Q_2 < Q_1$), as described by Kasun et al. (2016)

3.3.4 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 – see 3.3.1):

1) For each variable q (row) in the complete input dataset, compute its minimum and maximum values.
2) 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 'does not reach' p_t, one should add the missing amount, providing those patterns are the ones having more variables taking extreme (minimum or maximum) values.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3) 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} is not equal to the one imposed *a priori* (before step 1), which is due to the minimum required training patterns specified in step 2.

3.3.5 Input Normalization (feature 5)

The progress of training can be impaired if training data defines a region that is relatively narrow in some dimensions and elongated in others, which can be alleviated by normalizing each input variable across all data patterns. The implemented techniques are the following:

Linear Max Abs

Lachtermacher and Fuller (1995) proposed a simple normalization technique given by

$$\{Y_1\}_n(i,:) = \frac{Y_1(i,:)}{\max\left\{\left|Y_1(i,:)\right|\right\}} \quad , \quad (3)$$

where $\{Y_1\}_n(i,:)$ and $Y_1(i,:)$ are the normalized and non-normalized values of the i^{th} input variable for all learning patterns, respectively. Notation ':' in the column index, indicate the selection of all columns (learning patterns).

Linear [0, 1] and [-1, 1]

A linear transformation for each input variable (i), mapping values in Y_1(i,:) from [a*, b*]=[min(Y_1(i,:)), max(Y_1(i,:))] to a generic range [a, b], is obtained from

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

$$\{Y_1\}_n(i,:) = a + \frac{\left(Y_1(i,:) - a^*\right)}{\left(b^* - a^*\right)}(b-a) \quad . \quad (4)$$

Ranges $[a, b]=[0, 1]$ and $[a, b]=[-1, 1]$ were considered.

Nonlinear

Proposed by Pu and Mesbahi (2006), although in the context of output normalization, the only nonlinear normalization method implemented for input data reads

$$\{Y_1\}_n(i,j) = sign\left(Y_1(i,j)\right)\sqrt{\frac{\left|Y_1(i,j)\right|}{10^t}} + C(i) \quad , \quad (5)$$

where (i) $Y_1(i, j)$ is the non-normalized value of input variable i for pattern j, (ii) t is the number of digits in the integer part of $Y_1(i, j)$, (iii) $sign(\ldots)$ yields the sign of the argument, and (iv) C(i) is the average of two values concerning variable i, C_1(i) and C_2(i), where the former leads to a minimum normalized value of 0.2 for all patterns, and the latter leads to a maximum normalized value of 0.8 for all patterns.

Linear Mean Std

Tohidi and Sharifi (2014) proposed the following technique

$$\{Y_1\}_n(i,:) = \frac{Y_1(i,:) - \mu_{Y_1(i,:)}}{\sigma_{Y_1(i,:)}} \quad , \quad (6)$$

where $\mu_{Y_{1(i,:)}}$ and $\sigma_{Y_1(i,:)}$ are the mean and standard deviation of all non-normalized values (all patterns) stored by variable i.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3.3.6 Output Transfer Functions (feature 6)

Logistic

The most usual form of transfer functions is called Sigmoid. An example is the logistic function given by

$$\varphi(s) = \frac{1}{1+e^{-s}} \quad . \quad (7)$$

Hyperbolic Tang

The Hyperbolic Tangent function is also of sigmoid type, being defined as

$$\varphi(s) = \frac{e^{s} - e^{-s}}{e^{s} + e^{-s}} \quad . \quad (8)$$

Bilinear

The implemented Bilinear function is defined as

$$\varphi(s) = \begin{cases} s, & s \geq 0 \\ 0, & s < 0 \end{cases} \quad . \quad (9)$$

Identity

The Identity activation is often employed in output neurons, reading

$$\varphi(s) = s \quad . \quad (10)$$

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3.3.7 Output Normalization (feature 7)

Normalization can also be applied to the output variables so that, for instance, the amplitude of the solution surface at each variable is the same. Otherwise, training may tend to focus (at least in the earlier stages) on the solution surface with the greatest amplitude (Flood and Kartam 1994a). Normalization ranges not including the zero value might be a useful alternative since convergence issues may arise due to the presence of many small (close to zero) target values (Mukherjee et al. 1996). Four normalization methods were implemented. The first three follow eq. (4), where (i) [a, b] = 70% [φ_{min}, φ_{max}], (ii) [a, b] = 60% [φ_{min}, φ_{max}], and (iii) [a, b] = 50% [φ_{min}, φ_{max}], being [φ_{min}, φ_{max}] the output transfer function range, and [a, b] determined to be centered within [φ_{min}, φ_{max}] and to span the specified % (e.g., (b-a) = 0.7 (φ_{max} - φ_{min})). Whenever the output transfer functions are unbounded (Bilinear and Identity), it was considered [a, b] = [0, 1] and [a, b] = [-1, 1], respectively. The fourth normalization method implemented is the one described by eq. (6).

3.3.8 Network Architecture (feature 8)

Multi-Layer Perceptron Network (MLPN)

This is a feedforward ANN exhibiting at least one hidden layer. Fig. 2 depicts a 3-2-1 MLPN (3 input nodes, 2 hidden neurons and 1 output neuron), where units in each layer link only to some nodes located ahead. At this moment, it is appropriate to define the concept of partially- (PC) and fully-connected (FC) ANNs. In this work a FC feedforward network is characterized by having each node connected to every node in a different layer placed forward – any other type of network is said to be PC (e.g., the one in Fig. 2). According to Wilamowski (2009), PC

DOI: http://doi.org/10.31224/osf.io/wg7hd

MLPNs are less powerful than MLPN where connections across layers are allowed, which usually lead to smaller networks (less neurons).

Fig. 4 represents a generic MLFN composed of L layers, where l (l = 1,…, L) is a generic layer and '*ql'* a generic node, being $q = 1,..., Q_l$ its position in layer l (1 is reserved to the top node). Fig. 5 represents the model of a generic neuron (l = 2,…, L), where (i) p represents the data pattern presented to the network, (ii) subscripts m = 1,…, Q_n and n = 1,…, l-1 are summation indexes representing all possible nodes connecting to neuron 'ql' (recall Fig. 4), (iii) b_{ql} is neuron's bias, and (iv) w_{mnql} represents the synaptic weight connecting units '*mn'* and '*ql'*. Neuron's net input for the presentation of pattern p (S_{qlp}) is defined as

$$S_{qlp} = y_{mnp}\, w_{mnql} + b_{ql}, \qquad y_{mnp}\, w_{mnql} \equiv \sum_{m=1}^{Q_n} \sum_{n=1}^{l-1} y_{mnp}\, w_{mnql} \quad , \quad (11)$$

where y_{m1p} is the value of the m^{th} network input concerning example p. The output of a generic neuron can then be written as (l = 2,…, L)

$$y_{qlp} = \varphi_l(S_{qlp}) \quad , \quad (12)$$

where φ_l is the transfer function used for all neurons in layer l.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

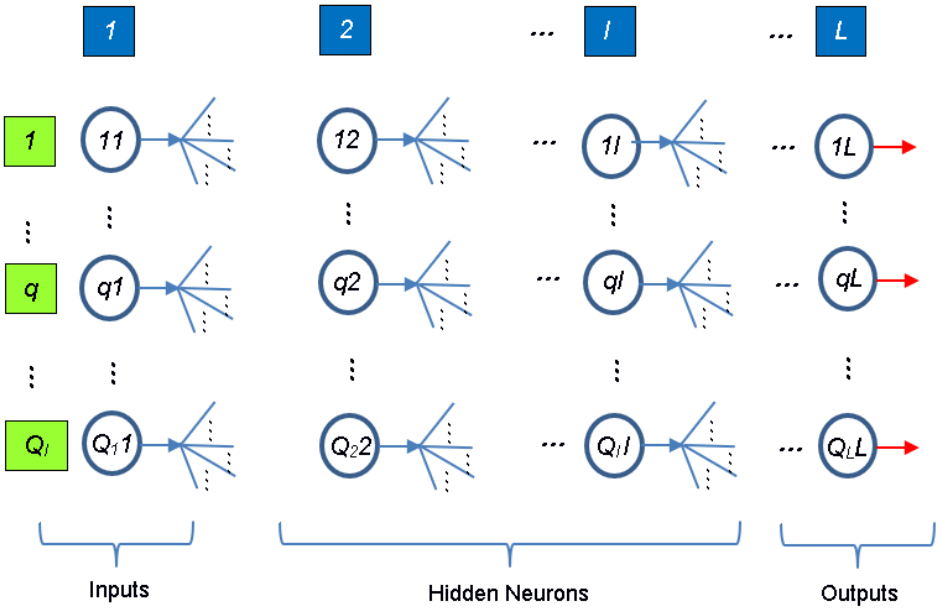


Fig. 4. Generic multi-layer feedforward network.

Radial-Basis Function Network (RBFN)

Although having similar topologies, RBFN and MLPN behave very differently due to distinct hidden neuron models – unlike the MLPN, RBFN have hidden neurons behaving differently than output neurons. According to Xie et al. (2011), RBFN (i) are specially

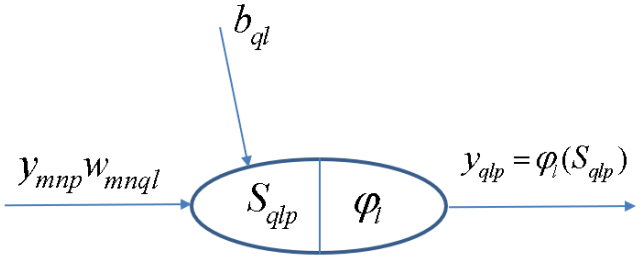


Fig. 5. Generic neuron placed anywhere in the MLPN of Fig. 4 ($l = 2,\ldots, L$).

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

recommended in functional approximation problems when the function surface exhibits regular peaks and valleys, and (ii) perform more robustly than MLPN when dealing with noisy input data. Although traditional RBFN have 3 layers, a generic multi-hidden layer (see Fig. 4) RBFN is allowed in this work, being the generic hidden neuron's model concerning node '$l_1 l_2$' (l_1 = 1,…,Q_{l2}, l_2 = 2,…, L-1) presented in Fig. 6. In this model, (i) $v_{l_1 l_2 p}$ and $\xi_{l_1 l_2}$ (called RBF center) are vectors of the same size ($\xi_{z l_1 l_2}$ denotes de z component of vector $\xi_{l_1 l_2}$, and it is a network unknown), being the former associated to the presentation of data pattern p, (ii) $\sigma_{l_1 l_2}$ is called RBF width (a positive scalar) and also belongs, along with synaptic weights and RBF centers, to the set of network unknowns to be determined through learning, (iii) φ_{l_2} is the user-defined radial basis (transfer) function (RBF), described in eqs. (20)-(23), and (iv) $y_{l_1 l_2 p}$ is neuron's output when pattern p is presented to the network. In ANNs not involving learning algorithms 1-3 in Tab. 4, vectors $v_{l_1 l_2 p}$ and $\xi_{l_1 l_2}$ are defined as (two versions of $v_{l_1 l_2 p}$ where implemented and the one yielding the best results was selected)

$$v_{l_1 l_2 p} = \left[y_{1(l_2-1)p}\, w_{1(l_2-1) l_1 l_2} \quad \cdots \quad y_{z(l_2-1)p}\, w_{z(l_2-1) l_1 l_2} \quad \cdots \quad y_{Q_{l_2-1}(l_2-1)p}\, w_{Q_{l_2-1}(l_2-1) l_1 l_2} \right]$$

or

$$v_{l_1 l_2 p} = \left[y_{1(l_2-1)p} \quad \cdots \quad y_{z(l_2-1)p} \quad \cdots \quad y_{Q_{l_2-1}(l_2-1)p} \right] \qquad , (13)$$

and

$$\xi_{l_1 l_2} = \left[\xi_{1 l_1 l_2} \quad \cdots \quad \xi_{z l_1 l_2} \quad \cdots \quad \xi_{Q_{l_2-1} l_1 l_2} \right]$$

whereas the RBFNs implemented through MATLAB neural net toolbox (involving learning algorithms 1-3 in Tab. 4) are based on the following definitions

DOI: http://doi.org/10.31224/osf.io/wg7hd

$$v_{l_1 l_2 p} = \left[y_{1(l_2-1)p} \quad \cdots \quad y_{z(l_2-1)p} \quad \cdots \quad y_{Q_{l_2-1}(l_2-1)p} \right]$$
$$\xi_{l_1 l_2} = \left[w_{1(l_2-1)l_1 l_2} \quad \cdots \quad w_{z(l_2-1)l_1 l_2} \quad \cdots \quad w_{Q_{l_2-1}(l_2-1)l_1 l_2} \right] \quad . \quad (14)$$

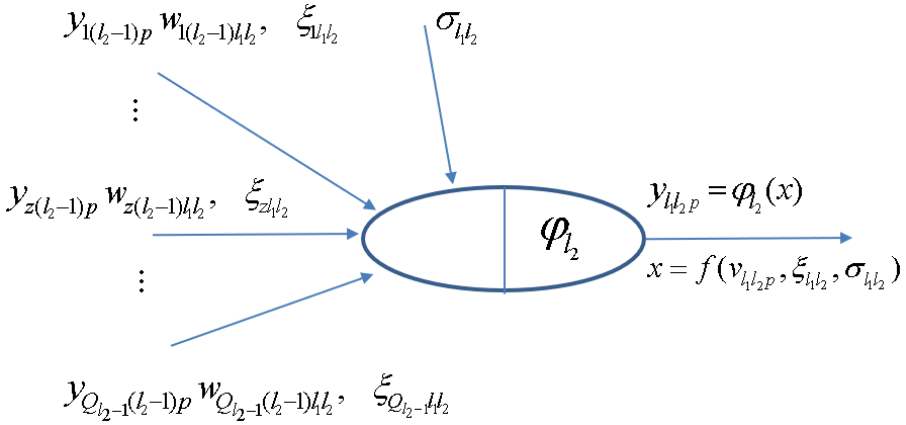


Fig. 6. Generic hidden neuron $l_1 l_2$ placed anywhere in the RBFN of Fig. 4 ($l_2 = 2,\ldots, L\text{-}1$).

Lastly, according to the implementation carried out for initialization purposes (described in 3.3.12), (i) RBF center vectors per hidden layer (one per hidden neuron) are initialized as integrated in a matrix (termed RBF center matrix) having the same size of a weight matrix linking the previous layer to that specific hidden layer, and (ii) RBF widths (one per hidden neuron) are initialized as integrated in a vector (called RBF width vector) with the same size of a hypothetic bias vector.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3.3.9 Hidden Nodes (feature 9)

Inspired by several heuristics found in the literature for the determination of a suitable number of hidden neurons in a single hidden layer net (Aymerich and Serra 1998, Rafiq et al. 2001, Xu and Chen 2008), each value in *hntest*, defined in eq. (15), was tested in this work as the total number of hidden nodes in the model, ie the sum of nodes in all hidden layers (initially defined with the same number of neurons). The number yielding the smallest performance measure for all patterns (as defined in 3.4, with outputs and targets not postprocessed), is adopted as the best solution. The aforementioned *hntest* is defined by

$$\begin{aligned} &incr = [4,\,4,\,4,\,10,\,10,\,10,\,10] \\ &minimum = [1,\,1,\,1,\,10,\,10,\,10,\,10] \\ &max_1 = \min\left(\text{round}\left(\max\left(2Q_1+Q_L,\,4Q_1,\,\sqrt{\frac{P}{Q_1\ln(P)}}\right)\right),\,1500\right) \\ &max_2 = \max\left(\min\left(\text{round}\left(0.1P\right),1500\right),300\right) \\ &maximum = [max_1,\,max_1,\,max_1,\,max_2,\,max_2,\,max_2,\,max_2] \\ &hntest = minimum(\mathrm{F}_{13}) : incr(\mathrm{F}_{13}) : maximum(\mathrm{F}_{13}) \end{aligned} \quad , \quad (15)$$

where (i) Q_1 and Q_L are the number of input and output nodes, respectively, (ii) P and P_t are the number of learning and training patterns, respectively, and (iii) F_{13} is the number of feature 13's method (see Tab. 4).

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3.3.10 Connectivity (feature 10)

For this ANN feature, three methods were implemented, namely (i) adjacent layers – only connections between adjacent layers are made possible, (ii) adjacent layers + input-output – only connections between (ii_1) adjacent and (ii_2) input and output layers are allowed, and (iii) fully-connected (all possible feedforward connections).

3.3.11 Hidden Transfer Functions (feature 11)

Besides functions (i) Logistic – eq. (7), (ii) Hyperbolic Tangent – eq. (8), and (iii) Bilinear – eq. (9), defined in 3.3.6, the ones defined next were also implemented as hidden transfer functions. During software validation it was observed that some hidden node outputs could be infinite or NaN (not-a-number in MATLAB – e.g., 0/0=Inf/Inf=NaN), due to numerical issues concerning some hidden transfer functions and/or their calculated input. In those cases, it was decided to convert infinite to unitary values and NaNs to zero (the only exception was the bipolar sigmoid function, where NaNs were converted to -1). Other implemented trick was to convert possible Gaussian function's NaN inputs to zero.

Identity-Logistic

In Gunaratnam and Gero (1994), issues associated with flat spots at the extremes of a sigmoid function were eliminated by adding a linear function to the latter, reading

$$\varphi(s) = \frac{1}{1+e^{-s}} + s \quad . \quad (16)$$

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

Bipolar

The so-called bipolar sigmoid activation function mentioned in Lefik and Schrefler (2003), ranging in [-1, 1], reads

$$\varphi(s) = \frac{1-e^{-s}}{1+e^{-s}} \quad . \quad (17)$$

Positive Saturating Linear

In MATLAB neural net toolbox, the so-called Positive Saturating Linear transfer function, ranging in [0, 1], is defined as

$$\varphi(s) = \begin{cases} 1, & s \geq 1 \\ s, & 0 < s < 1 \\ 0, & s \leq 0 \end{cases} \quad . \quad (18)$$

Sinusoid

Concerning less popular transfer functions, reference is made in Bai et al. (2014) to the sinusoid, which in this work was implemented as

$$\varphi(s) = \sin\left(\frac{\pi}{2}s\right) \quad . \quad (19)$$

Radial Basis Functions (RBF)

Although Gaussian activation often exhibits desirable properties as a RBF, several authors (e.g., Schwenker et al. 2001) have suggested several alternatives. Following nomenclature used in 3.3.8, (i) the Thin-Plate Spline function is defined by

DOI: http://doi.org/10.31224/osf.io/wg7hd

$$\varphi_{l_2}\left(s\right)=s\ln\left(\sqrt{s}\right),\quad s=\left\|v_{l_1l_2p}-\xi_{l_1l_2}\right\|^2 \quad , \quad (20)$$

(ii) the next function is employed as Gaussian-type function when learning algorithms 4-7 are used (see Tab. 4)

$$\varphi_{l_2}\left(s\right)=e^{-0.5s},\quad s=\left\|v_{l_1l_2p}-\xi_{l_1l_2}\right\|^2\Big/\sigma_{l_1l_2}^2 \quad , \quad (21)$$

(iii) the Multiquadratic function is given by

$$\varphi_{l_2}\left(s\right)=\sqrt{s},\quad s=\left\|v_{l_1l_2p}-\xi_{l_1l_2}\right\|^2+\sigma_{l_1l_2}^2 \quad , \quad (22)$$

and (iv) the Gaussian-type function (called 'radbas' in MATLAB toolbox) used by RBFNs trained with learning algorithms 1-3 (see Tab. 4), is defined by

$$\varphi_{l_2}\left(s\right)=e^{-s^2},\quad s=\left\|v_{l_1l_2p}-\xi_{l_1l_2}\right\|\sigma_{l_1l_2} \quad , \quad (23)$$

where || … || denotes the Euclidean distance in all functions.

3.3.12 Parameter Initialization (feature 12)

The initialization of (i) weight matrices (Q_a x Q_b, being Q_a and Q_b node numbers in layers a and b being connected, respectively), (ii) bias vectors (Q_b x 1), (iii) RBF center matrices (Q_{c-1} x Q_c, being c the hidden layer that matrix refers to), and (iv) RBF width vectors (Q_c x 1), are independent and in most cases randomly generated. For each ANN design carried out in the context of each parametric analysis combo, and whenever the parameter initialization method is not the 'Mini-Batch SVD', ten

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

distinct simulations varying (due to their random nature) initialization values are carried out, in order to find the best solution. The implemented initialization methods are described next.

Midpoint, Rands, Randnc, Randnr, Randsmall

These are all MATLAB built-in functions. *Midpoint* is used to initialize weight and RBF center matrices only (not vectors). All columns of the initialized matrix are equal, being each entry equal to the midpoint of the (training) output range leaving the corresponding initial layer node – recall that in weight matrices, columns represent each node in the final layer being connected, whereas rows represent each node in the initial layer counterpart. *Rands* generates random numbers with uniform distribution in [-1, 1]. *Randnc* (only used to initialize matrices) generates random numbers with uniform distribution in [-1, 1], and normalizes each array column to 1 (unitary Euclidean norm). *Randnr* (only used to initialize matrices) generates random numbers with uniform distribution in [-1, 1], and normalizes each array row to 1 (unitary Euclidean norm). *Randsmall* generates random numbers with uniform distribution in [-0.1, 0.1].

Rand [-lim, lim]

This function is based on the proposal in Waszczyszyn (1999), and generates random numbers with uniform distribution in [-*lim*, *lim*], being *lim* layer-dependent and defined by

$$lim = \begin{cases} Q_b^{\,1/Q_a} & , b < L \\ 0.5 & , b = L \end{cases} \quad , \quad (24)$$

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

where *a* and *b* refer to the initial and final layers integrating the matrix being initialized, and *L* is the total number of layers in the network. In the case of a bias or RBF width vector, *lim* is always taken as 0.5.

SVD

Although Deng et al. (2016) proposed this method for a 3-layer network, it was implemented in this work regardless the number of hidden layers.

Mini-Batch SVD

Based on Deng et al. (2016), this scheme is an alternative version of the former SVD. Now, training data is split into $\min\{Q_b, P_t\}$ chunks (or subsets) of equal size $P_{ti} = \max\{floor(P_t / Q_b), 1\}$ – *floor* rounds the argument to the previous integer (whenever it is decimal) or yields the argument itself, being each chunk aimed to derive $Q_{bi} = 1$ hidden node.

3.3.13 Learning Algorithm (feature 13)

The most popular learning algorithm is called error back-propagation (BP), a first-order gradient method. Second-order gradient methods are known to have higher training speed and accuracy (Wilamowski 2011). The most employed is called Levenberg-Marquardt (LM). All these traditional schemes were implemented using MATLAB toolbox (The Mathworks, Inc 2017).

DOI: http://doi.org/10.31224/osf.io/wg7hd

Back-Propagation (BP, BPA), Levenberg-Marquardt (LM)

Two types of BP schemes were implemented, one with constant learning rate (BP) – 'traingd' in MATLAB, and another with iteration-dependent rate, named BP with adaptive learning rate (BPA) – 'traingda' in MATLAB. The learning parameters set different than their default values are:

(i) Learning Rate = 0.01 / $cs^{0.5}$, being cs the chunk size, as defined in 3.3.15.

(ii) Minimum performance gradient = 0.

Concerning the LM scheme – 'trainlm' in MATLAB, the only learning parameter set different than its default value was the abovementioned (ii).

Extreme Learning Machine (ELM, mb ELM, I-ELM, CI-ELM)

Besides these traditional learning schemes, iterative and time-consuming by nature, four versions of a recent, powerful and non-iterative learning algorithm, called Extreme Learning Machine (ELM), were implemented (unlike initially proposed by the authors of ELM, connections across layers were allowed in this work), namely: (batch) ELM (Huang et al. 2006a), Mini-Batch ELM (mb ELM) (Liang et al. 2006), Incremental ELM (I-ELM) (Huang et al. 2006b), Convex Incremental ELM (CI-ELM) (Huang and Chen 2007).

3.3.14 Performance Improvement (feature 14)

A simple and recursive approach aiming to improve ANN accuracy is called Neural Network Composite (NNC), as described in Beyer et al. (2006). In this work, a maximum of

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

10 extra ANNs were added to the original one, until maximum error was not improved between successive NNC solutions. Later in this manuscript, a solution given by a single neural net might be denoted as ANN, whereas the other possible solution is called NNC.

3.3.15 Training Mode (feature 15)

Depending on the relative amount of training patterns, with respect to the whole training dataset, that is presented to the network in each iteration of the learning process, several types of training modes can be used, namely (i) batch or (ii) mini-batch. Whereas in the batch mode all training patterns are presented (called an epoch) to the network in each iteration, in the mini-batch counterpart the training dataset is split into several data chunks (or subsets) and in each iteration a single and new chunk is presented to the network, until (eventually) all chunks have been presented. Learning involving iterative schemes (e.g., BP- or LM-based) might require many epochs until an 'optimum' design is found. The particular case of having a mini-batch mode where all chunks are composed by a single (distinct) training pattern (number of data chunks = P_t , chunk size = 1), is called online or sequential mode. Wilson and Martinez (2003) suggested that if one wants to use mini-batch training with the same stability as online training, a rough estimate of the suitable learning rate to be used in learning algorithms such as the BP, is $\eta_{\text{online}}/\sqrt{cs}$, where cs is the chunk size and η_{online} is the online learning rate – their proposal was adopted in this work. Based on the proposal of Liang et al. (2006), the constant chunk size (cs) adopted for all chunks in mini-batch mode reads $cs = \min\{mean(hn) + 50, P_t\}$, being hn a vector

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

storing the number of hidden nodes in each hidden layer in the beginning of training, and *mean*(*hn*) the average of all values in *hn*.

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,

$$e_{qp} = 100 \left| \frac{d_{qp} - y_{qLp}}{d_{qp}} \right| \quad , \quad (25)$$

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. (25) is replaced by 1 whenever $|d_{qp}| < 0.05 - d_{qp}$ in the nominator keeps its real value. This exception to eq. (25) 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).

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3.4.1 Maximum Error

This variable measures the maximum relative error, as defined by eq. (25), 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. (25), 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. (25), 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 (2018b).

DOI: http://doi.org/10.31224/osf.io/wg7hd

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 first SA, (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.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

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 microprocessors used in this work have the following features: OS: Win10Home 64bits, RAMs: 48/128 GB, Local Disk Memory: 1 TB, CPUs: Intel® Core™ i7 8700K @ 3.70-4.70 GHz / i9 7960X @ 2.80-4.20 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	1	1	1	1	1	1	3	2	3	1	3
2	1	2	6	2	1	7	1	1	1	1	3	2	5	1	3
3	1	2	6	2	1	1	1	1	1	1	3	2	3	1	3
4	1	2	6	2	1	1	2	1	1	1	3	2	3	1	3
5	1	2	6	3	1	1	3	1	1	1	3	2	3	1	3
6	1	2	6	2	1	7	4	1	1	1	3	2	3	1	3
7	1	2	6	3	1	7	5	1	1	1	3	2	3	1	3
8	1	2	6	3	1	7	5	1	1	1	1	5	3	1	3
9	1	2	6	3	1	7	5	1	3	3	1	5	3	1	3

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, 1,

DOI: http://doi.org/10.31224/osf.io/wg7hd

3, 1, 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 8-11-11-11-1. Concerning connectivity, the network is fully-connected, and the hidden and output transfer functions are all Logistic (eq. (7)) and Identity (eq. (10)), respectively. The network was trained using the LM algorithm (1500 epochs). After design, the average network computing time concerning the presentation of a single example (including data pre/postprocessing) is 6.93E-05 s – Fig. 7 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 (2018a).

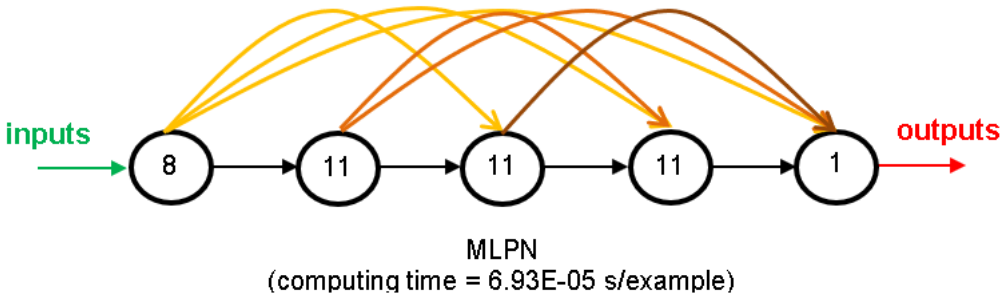


Fig. 7. Proposed 8-11-11-11-1 fully-connected MLPN – simplified scheme.

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).

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

Tab. 6. Performance results for the best design from each parametric sub-analysis: (a) ANN, (b) NNC.

SA	ANN				
	Max Error (%)	Performance All Data (%)	Errors > 3% (%)	Total Hidden Nodes	Running Time / Data Point (s)
1	7.5	0.7	1.8	32	7.33E-05
2	201.0	5.7	54.9	365	7.06E-05
3	8.2	0.7	1.2	32	8.49E-05
4	8.5	0.7	1.3	32	7.67E-05
5	6.8	0.7	1.8	32	6.86E-05
6	7.2	0.7	1.6	32	7.57E-05
7	28.9	1.5	13.2	29	6.82E-05
8	8.2	0.9	3.4	29	6.70E-05
9	3.7	0.4	0.1	33	6.93E-05

(a)

SA	NNC				
	Max Error (%)	Performance All Data (%)	Errors > 3% (%)	Total Hidden Nodes	Running Time / Data Point (s)
1	-	-	-	-	-
2	180.0	5.2	51.4	365	8.16E-05
3	-	-	-	-	-
4	8.6	0.6	0.9	32	7.83E-05
5	-	-	-	-	-
6	2.6	0.3	0.0	32	8.34E-05
7	9.5	0.9	3.9	29	6.93E-05
8	7.0	0.7	1.9	29	6.80E-05
9	-	-	-	-	-

(b)

3.7.1 Input Data Preprocessing

For future use of the proposed ANN to simulate new data $Y_{1,sim}$ (8 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 1 – 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.

DOI: http://doi.org/10.31224/osf.io/wg7hd

Dimensional Analysis and Dimensionality Reduction

Since dimensional analysis (*d.a.*) and dimensionality reduction (*d.r.*) were not carried out, one has

$$\left\{Y_{1,sim}\right\}_{d.r.}^{after} = \left\{Y_{1,sim}\right\}_{d.a.}^{after} = Y_{1,sim} \quad . \quad (26)$$

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

$$\left\{Y_{1,sim}\right\}_{n}^{after} = \text{INP .x} \left\{Y_{1,sim}\right\}_{d.r}^{after}$$

$$\text{INP} = \begin{bmatrix} 0.125 \\ 0.00139275766016713 \\ 0.00178571428571429 \\ 0.00364431486880467 \\ 0.00142857142857143 \\ 0.0666666666666667 \\ 0.0037037037037037 \\ 0.04 \end{bmatrix} \quad , \quad (29)$$

where one recalls that operator '.x' multiplies component *i* in vector INP by all components in row *i* of $\{Y_{1,\text{sim}}\}_{\text{d.r}}^{after}$.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

3.7.2 ANN-Based Analytical Model

Once determined the preprocessed input dataset $\{Y_{1,sim}\}_n^{after}$(8 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'.

$$\begin{aligned}
Y_2 &= \varphi_2\left(W_{1-2}^T\left\{Y_{1,sim}\right\}_n^{after} + b_2\right)\\
Y_3 &= \varphi_3\left(W_{1-3}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-3}^T Y_2 + b_3\right)\\
Y_4 &= \varphi_4\left(W_{1-4}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-4}^T Y_2 + W_{3-4}^T Y_3 + b_4\right)\\
\left\{Y_{5,sim}\right\}_n^{after} &= \varphi_5\left(W_{1-5}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-5}^T Y_2 + W_{3-5}^T Y_3 + W_{4-5}^T Y_4 + b_5\right)
\end{aligned}, \quad (30)$$

where

$$\begin{aligned}
\varphi_2 &= \varphi_3 = \varphi_4 = \varphi(s) = \frac{1}{1+e^{-s}}\\
\varphi_5 &= \varphi_5(s) = s
\end{aligned}. \quad (31)$$

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

Arrays $W_{j\text{-}s}$ and b_s are stored online in Developer (2018b), 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.

Once obtained $\{Y_{5,sim}\}_n^{after}$, the following relations hold for its relation to its non-normalized ($\{Y_{5,sim}\}_{d.a.}^{after}$) and original ($Y_{5,sim}$) formats (just after the dimensional analysis stage, and free of any pre-processing effects, respectively), reading

$$Y_{5,sim} = \left\{Y_{5,sim}\right\}_{d.a.}^{after} = \left\{Y_{5,sim}\right\}_{n}^{after} \quad , \quad (32)$$

since no output normalization nor dimensional analysis were carried out.

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. 8) 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), as defined in eq. (1); (ii) a performance plot (Fig. 9), where performance

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

(average error) values are displayed for several learning datasets; and (iii) an error plot (Fig. 10), 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.

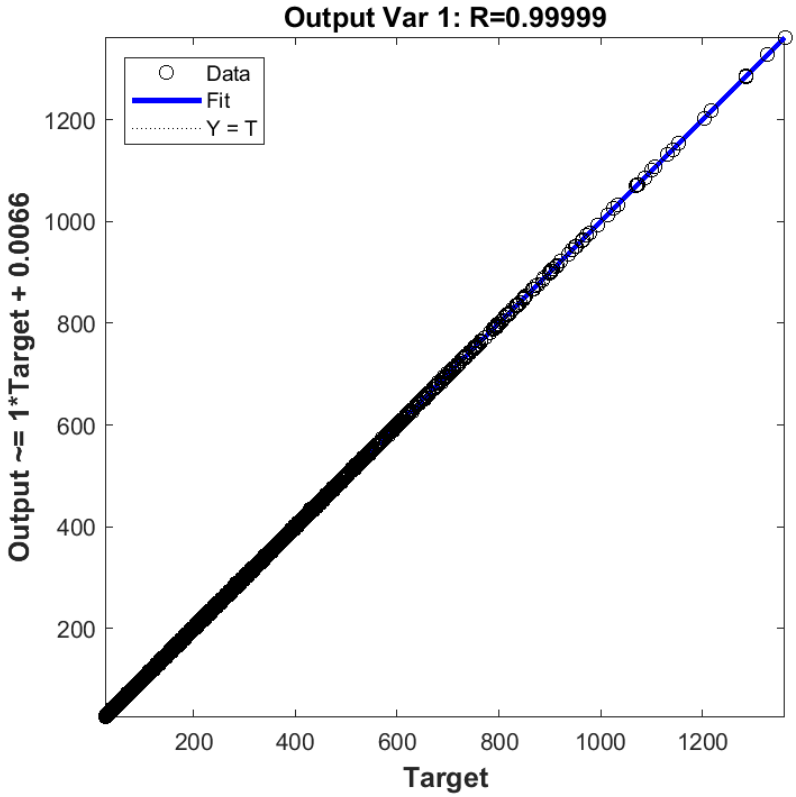


Fig. 8. Regression plot for the proposed ANN.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

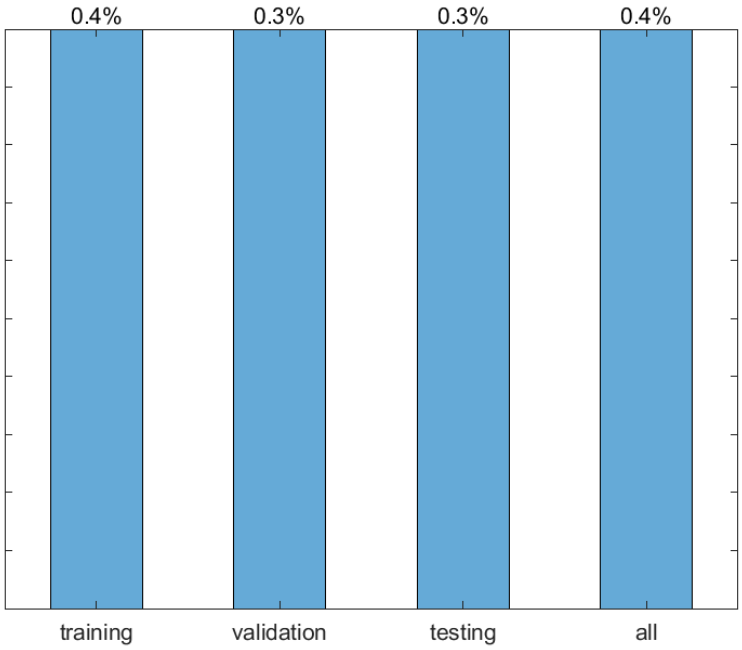


Fig. 9. Performance plot (mean errors) for the proposed ANN.

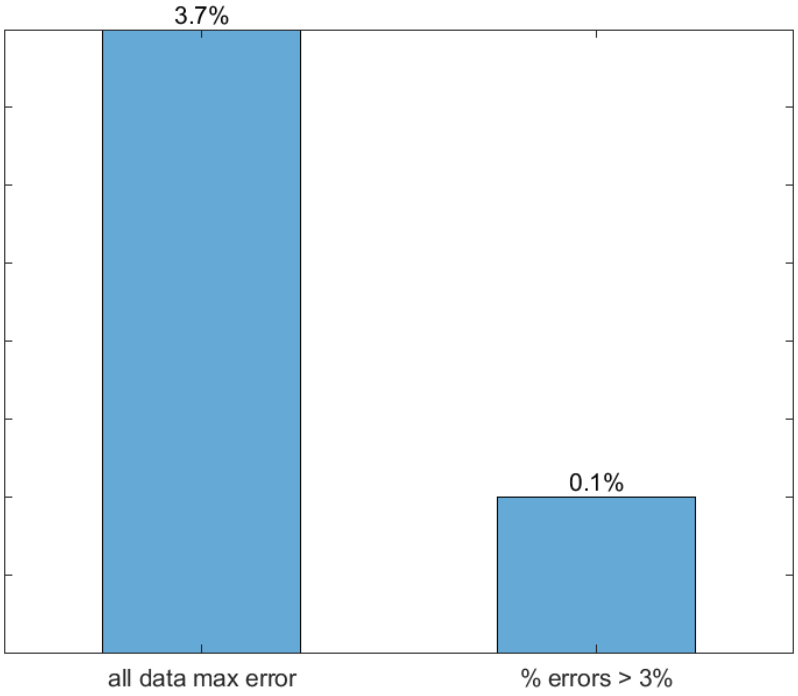


Fig. 10. Error plot for the proposed ANN.

DOI: http://doi.org/10.31224/osf.io/wg7hd

4. Design Considerations

The bar chart in Fi. 11, where beams numbers in the horizontal axis are referenced to the data point ID in the ANN dataset (Developer 2018a), presents (i) the design loads given by SCI-P355 (Lawson and Hicks 2011), and (ii) the FEA-based elastic buckling loads obtained from this work. As expected, SCI-P355 yielded significantly lower loads and it is worth noting that the differences between the two approaches do not produce the same percentage variance in load estimates for the 8 randomly selected beams. In particular, for slender web-posts (i.e., closely spaced web openings) such as in beams 82, 136, 163, and 217, buckling of the web-post will always govern the design in SCI-P355, hence the reduced design load. As for the FEAs, parameter led (end web-post distance to the support – see Figure 1(a)) governs the design for widely spaced web openings. SCI-P355 does not consider the distance led and recommends it to be greater than 50% of the opening diameter. In this work, the opening diameter and web-post width were taken within the recommended design limits ($1.25<H/\Phi<1.75$ and $1.08<(bwp+\Phi)/\Phi<1.5$, respectively).

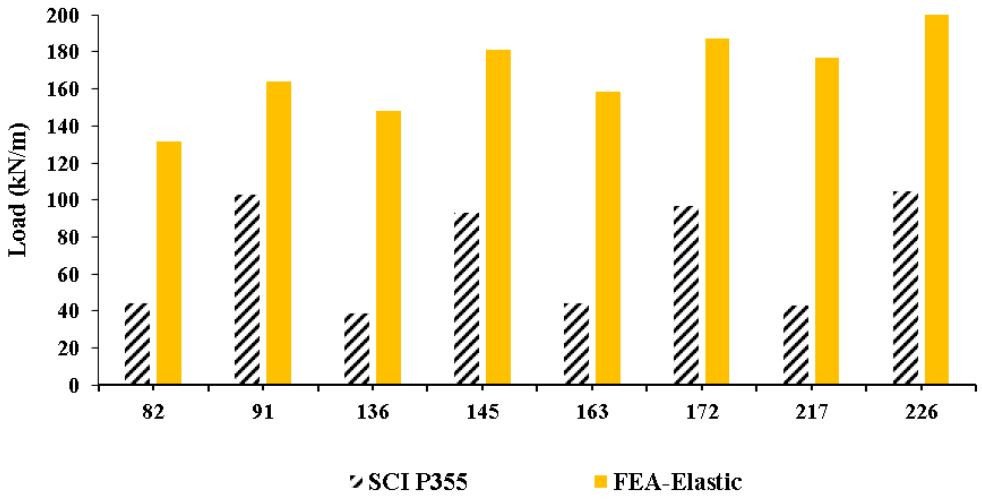


Fig. 11. SCI P355 design strength vs. FE-based elastic buckling prediction.

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

5. Discussion

In future publications it will be guaranteed that the validation and testing data subsets will be composed only by points where at least one variable (does not have to be the same for all) takes a value not taken in the training subset by that same variable. Based on very recent empirical conclusions by Abambres, the author believes it will lead to more robust ANN-based analytical models concerning their generalization ability (i.e. prediction accuracy for any data point within the variable ranges of the design data).

6. Concluding Remarks

An ANN-based analytical model is proposed to effectively predict the elastic buckling load of simply supported cellular steel beams subjected to a uniformly distributed load. Finite element solutions from 3645 distinct beams were used for ANN design (training, validation, and testing). The independent variables adopted as ANN inputs are the following: beam's length, opening diameter, web-post width, cross-section height, web thickness, flange width, flange thickness, and the distance between the last opening edge and the end support. The maximum and average relative errors yielded by the proposed ANN among the 3645 data points were 3.7% and 0.4%, respectively. Moreover, that model is able to compute the buckling load of a single beam in less than a millisecond, for any current personal computer. These facts make the proposed model a potential tool for structural engineers and researchers who aim to accurately estimate the elastic buckling load of cellular steel beams (i) within the ranges of the

Abambres M, Rajana K, Tsavdaridis K, Ribeiro T (2018). Neural Network-based formula for the buckling load Prediction of I-section cellular steel beams. **engrXiv** (December), 1-50, doi: http://doi.org/10.31224/osf.io/wg7hd

DOI: http://doi.org/10.31224/osf.io/wg7hd

input variables adopted in this study (see Tab. 1), and (ii) without the burden of the costly resources associated to FEA.

This research is the first step of an ongoing investigation that aims to propose a novel and simple analytical design method to accurately compute the inelastic resistance of cellular steel beams.

Author Contributions

Section 3, 5: MA; Section 1: KR, KDT, TPR; Sections 2 and 4: KR, KDT; All other contributions equally distributed among all authors.

References

Akrami V, Erfani S (2016). Review and assessment of design methodologies for perforated steel beams, Journal of Structural Engineering, 142(2), 1-14.

Anderson D, Hines EL, Arthur SJ, Eiap EL (1997). Application of Artificial Neural Networks to the Prediction of Minor Axis Steel Connections, Computers & Structures, 63(4), 685-692.

Aymerich F, Serra M (1998). Prediction of fatigue strength of composite laminates by means of neural networks, Key Eng. Materials, 144(September), 231–240.

Bai Z, Huang G, Wang D, Wang H, Westover M (2014). Sparse extreme learning machine for classification, IEEE Transactions on Cybernetics, 44(10), 1858–70.

Beyer W, Liebscher M, Beer M, Graf W (2006). Neural Network Based Response Surface Methods - A Comparative Study, 5th German LS-DYNA Forum, October 2006, 29-38, Ulm.

Bhaskar R, Nigam A (1990). Qualitative physics using dimensional analysis, Artificial Intelligence, 45(1-2), 111–73.

Chung KF, Liu CH, Ko ACH (2003). Steel beams with large web openings of various shapes and sizes: an empirical design method using a generalized moment-shear interaction curve. Journal of Construction Steel Research, 59(9), 1177-1200.

Chung KF, Lui TCH, Ko ACH (2001). Investigation on vierendeel mechanism in steel beams with circular web openings. Journal of Constructional Steel Research, 57(5), 467-490.

Darwin, D (1990). Steel and composite beams with web opening, Steel Design Guide Series 2, AISC, Chicago.

Dassault Systèmes (2011). ABAQUS 6.11, Abaqus/CAE user's manual, Dassault Systemes, USA.

Dassault Systèmes Simulia Corp (2017). ABAQUS CAE (2017). [Software].

Deng W-Y, Bai, Z., Huang, G.-B. and Zheng, Q.-H. (2016). A fast SVD-Hidden-nodes based extreme learning machine for large-scale data Analytics, Neural Networks, 77(May), 14–28.

DOI: http://doi.org/10.31224/osf.io/wg7hd

Developer (2018a). dataset ANN [Data set]. Zenodo, http://doi.org/10.5281/zenodo.1486181

Developer (2018b). W and b arrays [Data set]. Zenodo, http://doi.org/10.5281/zenodo.1486268

El-Sawhy KL, Sweedan AMI, Martini MI (2014). Moment gradient factor of cellular steel beams under inelastic flexure. Journal of Constructional Steel Research, 98(2014), 20-34.

Flood I (2008). Towards the next generation of artificial neural networks for civil engineering, Advanced Engineering Informatics, 228(1), 4-14.

Flood I, Kartam N (1994a). Neural Networks in Civil Engineering: I-Principals and Understanding, Journal of Computing in Civil Engineering, 8(2), 131-148.

Gholizadeh S, Pirmoz A, Attarnejad R (2011). Assessment of load carrying capacity of castellated steel beams by neural networks, Journal of Constructional Steel Research, 67(5), 770–779.

Gholizadeh S, Pirmoz A, Attarnejad R (2011). Assessment of the load carrying capacity of castellated steel beams by artificial neural networks. Journal of Constructional Steel Research, 67(2011), 770-779.

Gunaratnam DJ, Gero JS (1994). Effect of representation on the performance of neural networks in structural engineering applications, Computer-Aided Civil and Infrastructure Engineering, 9(2), 97–108.

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.

Huang G, Chen L, Siew C (2006b). Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE transactions on neural networks, 17(4), 879–92.

Huang G-B, Chen L (2007). Convex incremental extreme learning machine, Neurocomputing, 70(16–18), 3056–3062.

Huang G-B, Zhu Q-Y, Siew C-K (2006a). Extreme learning machine: Theory and applications, Neurocomputing, 70(1-3), 489-501.

Kasun LLC, Yang Y, Huang G-B, Zhang Z (2016). Dimension reduction with extreme learning machine, IEEE Transactions on Image Processing, 25(8), 3906–18.

Lachtermacher G, Fuller JD (1995). Backpropagation in time-series forecasting, Journal of Forecasting 14(4), 381–393.

Lawson RM (1987). Design for openings in the webs of composite beams SCI P068. Steel Construction Institute, Berkshire, UK.

Lawson RM, Hicks SJ (2011). Design of composite beams with large openings SCI P355. Steel Construction Institute, Berkshire, UK.

Lefik M, Schrefler BA (2003). Artificial neural network as an incremental non-linear constitutive model for a finite element code, Computer Methods in Applied Mechanics and Engineering, 192(28–30), 3265–3283.

Liang N, Huang G, Saratchandran P, Sundararajan N (2006). A fast and accurate online Sequential learning algorithm for Feedforward networks, IEEE Transactions on Neural Networks, 17(6), 1411–23.

Lucas WK, Darwin D (1990). Steel and composite beams with web openings. The American Iron and Steel Institute, Kansas.

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

Morkhade SG, Gupta LM (2015). An experimental and parametric study on steel beams with web openings. International Journal of Advanced Structural Engineering, 7(3), 249-260.

DOI: http://doi.org/10.31224/osf.io/wg7hd

Mukherjee A, Deshpande JM, Anmala J (1996), Prediction of buckling load of columns using artificial neural networks, Journal of Structural Engineering, 122(11), 1385–7.

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.

Pu Y, Mesbahi E (2006). Application of artificial neural networks to evaluation of ultimate strength of steel panels, Engineering Structures, 28(8), 1190–1196.

Rafiq M, Bugmann G, Easterbrook D (2001). Neural network design for engineering applications, Computers & Structures, 79(17), 1541–1552.

Rajana K (2018). Advanced computational parametric study of the linear elastic and non-linear post buckling behaviour of non-composite cellular steel beams, MSc (Eng). Thesis, University of Leeds, doi: 10.17605/osf.io/FHZJV

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

Schwenker F, Kestler H, Palm G (2001). Three learning phases for radial-basis-function networks, Neural networks, 14(4-5), 439–58.

SEI/ASCE (1998). Specifications for structural steel beams with openings, SEI/ASCE 23-97, ASCE, Reston VA.

Sharifi Y, Tohidi S (2014). Lateral-torsional buckling capacity assessment of web opening steel girders by artificial neural networks – elastic investigation. Frontiers of Structural and Civil Engineering, 8(2), 167-177.

Surtees JO, Lui Z (1995). Report of loading tests on cellform beams. Research Report, University of Leeds, Leeds.

Sweedan AMI, El-Sawhy K (2011). Elastic local buckling of perforated webs of steel cellular beam-column elements. Journal of Constructional Steel Research, 67(7), 1115-1127.

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

Tohidi S, Sharifi Y (2014). Inelastic lateral-torsional buckling capacity of corroded web opening steel beams using artificial neural networks, The IES Journal Part A: Civil & Structural Eng, 8(1), 24–40.

Tohidi S, Sharifi Y (2014). Inelastic lateral-torsional buckling capacity of corroded web opening steel beams using artificial neural networks. The IES Journal Part A: Civil and Structural Engineering, 8(1), 24-40.

Tohidi S, Sharifi Y (2016). Load-carrying capacity of locally corroded steel plate girder ends using artificial neural network. Thin-walled Structures, 100(2016), 48-61.

Tsavdaridis KD (2010). Structural performance of perforated steel beams with novel web openings and with partial concrete encasement, Ph.D. Thesis, City University London.

Tsavdaridis KD, D'Mello C (2009). Finite Element Investigation of Perforated Beams with Different Web Opening Configurations. The 6th International Conference on Advances is Steel Structures (ICASS 2009). 16-18 December 2009, Hong Kong, China, 213-220.

Tsavdaridis KD, D'Mello C (2011). Web buckling study of the behaviour and strength of perforated steel beams with different novel web opening shapes. Journal of Constructional Steel Research, 67 (10), 1605-20.

Tsavdaridis KD, D'Mello C (2012). Vierendeel bending study of perforated steel beams with various novel web opening shapes through non-linear finite element analyses. Journal of Structural Engineering, 138 (10), 1214-30.

Tsavdaridis KD, Kingman JJ, Toropov VV (2015). Application of Structural Topology Optimisation to Perforated Steel Beams. Computers and Structures, 158(October), 108-123.

Uenoya M, Redwood RG (1978). Buckling of webs with openings. Computers and Structures, 9(2), 191-199.

Verweij JG (2010). Cellular beam-columns in portal frame structures. MSc. Thesis, Delft University of Technology, Holland.

Ward JK (1990). Design of composite and non-composite cellular beams SCI P100. Steel Construction Institute, Berkshire, UK.

DOI: http://doi.org/10.31224/osf.io/wg7hd

Waszczyszyn Z (1999). Neural Networks in the Analysis and Design of Structures, CISM Courses and Lectures No. 404, Springer, Wien, New York.

Wilamowski BM (2009). Neural Network Architectures and Learning algorithms, IEEE Industrial Electronics Magazine, 3(4), 56-63.

Wilamowski BM (2011). How to not get frustrated with neural networks, 2011 IEEE International Conference on Industrial Technology (ICIT), 14-16 March 2011, IEEE (eds), Auburn University, Auburn, AL, USA.

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

Wilson DR, Martinez TR (2003). The general inefficiency of batch training for gradient descent learning, Neural Networks, 16(10), 1429–1451.

Xie T, Yu H, Wilamowski B (2011). Comparison between traditional neural networks and radial basis function networks, 2011 IEEE International Symposium on Industrial Electronics (ISIE), IEEE(eds), 27-30 June 2011, Gdansk University of Technology Gdansk, Poland, 1194–99.

Xu S, Chen L (2008). Novel approach for determining the optimal number of hidden layer neurons for FNN's and its application in data mining, In: International Conference on Information Technology and Applications (ICITA), Cairns (Australia), 23–26 June 2008, pp 683–686.