Soft Computing (2024) 28:6937–6951
https://doi.org/10.1007/s00500-023-09491-0

NEURAL NETWORKS

On the use of AI for metamodeling: a case study of a 3D bar structure

Larissa Driemeier[1] · Eduardo Lobo Lustosa Cabral[2] · Gabriel Lopes Rodrigues[1] · Marcos Tsuzuki[1] · Marcilio Alves[1] · Lucas Pires da Costa[1] · Rafael Traldi Moura[1]

Accepted: 21 November 2023 / Published online: 27 December 2023

Abstract
In scenarios where complex analyses are routinely conducted on similar structures, such as in a redesign process to meet performance requirements or when input parameters require frequent adjustments within a specified domain, a practical approach involves the use of metamodels calibrated using machine learning methodologies. In our investigation, we introduce a metamodel that utilizes an artificial neural network to analyze 3D nonlinear structures undergoing plastic deformations and large strains. Snap-through and snapback behaviors are addressed through network training, which is based on 10,000 Force vs Displacement curves (target outputs) obtained from nonlinear finite element analyses. This interplay between finite element analysis and machine learning, as demonstrated here, exhibits promising potential as an effective technique. The results indicate that the proposed deep neural network can learn from the simulations of finite elements. The discussion explores scenarios where the utilization of AI in the analysis of nonlinear structures is justified.

Keywords Deep neural network · Metamodel · Finite element method · Truss analysis

1 Introduction

Differential equations govern the response of a continuum-mechanics-based mathematical model. However, exact analytical solutions that satisfy all boundary and initial conditions are only possible for simple models, and therefore, numerical approximations must be employed. The Finite Element Method (FEM) emerges as the primary numerical tool employed for solving partial differential or integral equations. In current applications, nonlinear Finite Element Analyses (FEAs) are widely employed for modeling complex engineering structures, providing accurate predictions across diverse domains, spanning scientific research to industrial applications.

In parallel, Artificial Intelligence (AI) is a branch of computer science that attempts to understand intelligent entities (Russell and Norvig 2009). In a more restrictive definition, AI involves the creation of machines capable of executing tasks traditionally linked to intelligence, particularly in replicating intelligent human behavior. As a subset within the vast field of artificial intelligence, Machine Learning (ML) focuses on the technologies and algorithms that empower systems to attain intelligence. Artificial intelligence, in the form of machine learning and data science, is becoming ubiquitous in practically every branch of knowledge—business, medicine, education, engineering, economics (Ma 2021; Gao et al. 2018; Esteva et al. 2017; Liao et al. 2021; Young et al. 2018; Shao and Liu 2020; Sharma et al. 2021; Zhao 2021; Zeng et al. 2021; Lukin et al. 2020).

These authors have contributed equally to this work.

✉ Larissa Driemeier
driemeie@usp.br

Eduardo Lobo Lustosa Cabral
elobocabral@gmail.com

Gabriel Lopes Rodrigues
gabriellopes@usp.br

Marcos Tsuzuki
mtsuzuki@usp.br

Marcilio Alves
maralves@usp.br

Lucas Pires da Costa
lucas.pcosta@usp.br

Rafael Traldi Moura
moura.gmsie@usp.br

[1] Department of Mechatronics Engineering, University of São Paulo, Avenida Prof. Luciano Gualberto, 380, São Paulo 05508-010, SP, Brazil

[2] Nuclear and Energy Research Institute, Avenida Lineu Prestes, São Paulo 05508-000, SP, Brazil

Among the ML techniques, this work makes use of Artificial Neural Networks (ANNs), or, more particularly, Deep Neural Networks (DNNs). DNNs, where *deep* refers to the number of layers in a neural network, and Convolutional Neural Networks (CNNs), which are neural networks named for their utilization of convolution operations instead of dense multiplication. This approach significantly reduces the size of the problem, allowing the incorporation of images as input to the network. Although ANN is not a new idea, the recent popularity of DNNs is due to cheaper and more powerful computational processors, together with the increasing amount and variety of available data. According to Abiodun et al. (2018), ANNs exhibit outstanding properties such as self-learning, adaptability, fault tolerance, and a nonlinear approach in mapping inputs to outputs.

Particularly in structural analysis, pioneering works in the use of neural networks date from 90s, as in Vanluchene and Sun (1990). The authors developed three different networks for three well known structural problems. For example, in a simply supported beam with concentrated loads, the authors trained an ANN to accurately determine the load position needed to generate the given moment profile within the beam. This example provided a significant insight into the application of AI in structural analysis, particularly in addressing inverse problems. Indeed, as described in Lee et al. (2018), the limited exploration of ANNs in traditional structural analysis stems from the suboptimal performance and the extensive computational time required for training. They suggest the use of DNNs to overcome performance problems.

One of the first use of CNNs in structural engineering was conducted by Abdeljaber et al. (2017). The authors introduced a methodology for detecting structural damage through vibrations employing one-dimensional CNNs. In fact, the convolutional neural network is a successful method of monitoring structural health in real-time (Teng et al. 2020). Abdeljaber et al. (2017) used a combination of FEA and experimental data due to the difficulty in obtaining enough damaged structures in practical engineering.

Moreover, obtaining optimal solutions requires numerous iterations involving nonlinear analyses.

Recently, a few attempts to apply ANN to obtain optimal solutions can be found in the literature (Torky and Aburawwash 2018; Gu et al. 2018; Bilal et al. 2020; Abueidda et al. 2020; Zhou et al. 2021), since the process requires numerous iterations involving nonlinear analyzes. Always, the authors proposed to solve a specific problem, without generalization for different structural problems. The authors change, at most, the boundary conditions and load. Torky and Aburawwash (2018) applied the deep learning approach to optimize the design of prestressed beams. Gu et al. (2018) applied machine learning to search for the optimal design of composites. Differential evolution algorithms for solving complex optimization problems are reviewed in Bilal et al. (2020). Abueidda et al. (2020) suggested a pipeline to train a neural network model to solve a classical optimization problem.

Finally, complex structural problems, involving material, geometry and/or contact nonlinearities, are often solved by computationally expensive FE analysis (Koller et al. 2021; Liang et al. 2018; Lukin et al. 2020). Liang et al. (2018) estimated the stress distributions of the aorta using DNN instead of complex structural FEA and computational fluid dynamics. Arnold and King (2021) suggested a state-space model formulation utilizing PINNs (Physics-Informed Neural Networks). The approach is suitable for real-time applications of complex dynamic system models, based on partial differential equations.

Complex models, such as those mentioned before, can be simplified by a metamodel, or surface response, that is, a (simplified) model created from a (complex) model. One motivation for metamodeling is when the original complex model is frequently redesigned with modifications in geometry and/or loading and/or boundary conditions delimited by instantaneous operational aspects. An example is the design of a blowout preventer addressed in Lukin et al. (2020). The authors found a metamodel based on loading conditions and misalignment of the riser and rams. In that case, only a limited number of characteristics varied, facilitating the use of a simplified model that generates a correct answer at significantly lower computational and financial costs, given that the use of a sophisticated finite element software is only necessary to generate the dataset for network training.

Since metamodeling is performed by exploring relationships among inputs and outputs, the response surface can be encapsulated by the various layers of a DNN.

In the present work, a metamodel for a 3D star-like bar structure is proposed. Its nonlinear behavior is analyzed via the traditional finite element method, and the results are target outputs for the DNN training. Basically, instead of using constitutive laws, compatibility, and equilibrium equations, a neural network model is trained with approximately 10,000 input data that define the structure geometry and the respective response in terms of displacement–force. Then, the performance of the proposed DNN to run a nonlinear structural problem is evaluated and the benefit of doing so is discussed.

The work is presented as follows: Sect. 2 presents the geometry and material characteristics of the structural problem to be explored in the next sections; Sect. 3 details a flowchart that defines the implementation from data generation up to result analyses. Finally, in Sect. 4, the results are discussed in terms of performance, restrictions, processing speed, scalability, and convergence. The procedure developed here can be applied to other nonlinear mechanical problems.

2 3D star-like bar structure

Fig. 1 shows the geometry and loading of a 3D dome truss composed by 24 bars with areas randomly varying from $25 to 85$ mm^2. In Fig. 1, different colors for the nodes represent different z coordinates, as shown in the side view. The displacement is imposed in the highest unique green node ($z = 72.16\ mm$) and blue nodes ($z = 0\ mm$) have restricted displacement in all directions, intermediary turquoise nodes ($z = 42.16\ mm$) are unrestrained nodes. The dimensions are based on Driemeier et al. (2005). The bars have a homogeneous cross-section within the element and are made of isotropic elastoplastic material, Table 1. The different cross-section areas for each bar are detailed in the next section.

The selection of 3D bar structures for this study is grounded in their inherently complex nonlinear mechanical behavior resulting from its kinematics and constitutive material model with plasticity. In fact, there is an extensive literature dedicated to the examination of nonlinearities (Alves 2020; Rezaiee-Pajand et al. 2020; Hencky 2020), design (Arora et al. 2018), and optimization (Li et al. 2019; Shakya et al. 2018; Ozbasaran and Eryilmaz Yildirim 2020) analyses in bar structures.

Table 1 Generic elastoplastic properties for the 3D dome bars

Property	Value
Mass Density [ton/mm^3]	7.8e−9
Young Modulus [MPa]	210000
Poisson's Ratio [–]	0.3
Yield Stress [MPa]	660

The classical elastoplastic behavior, with positive piecewise linear hardening for trusses is summarized in Box 1. $N(\alpha)$ is the approximating piecewise linear isotropic hardening function for the current value of the work hardening parameter α and is specified in tabular form in Table 2. Utiliz-

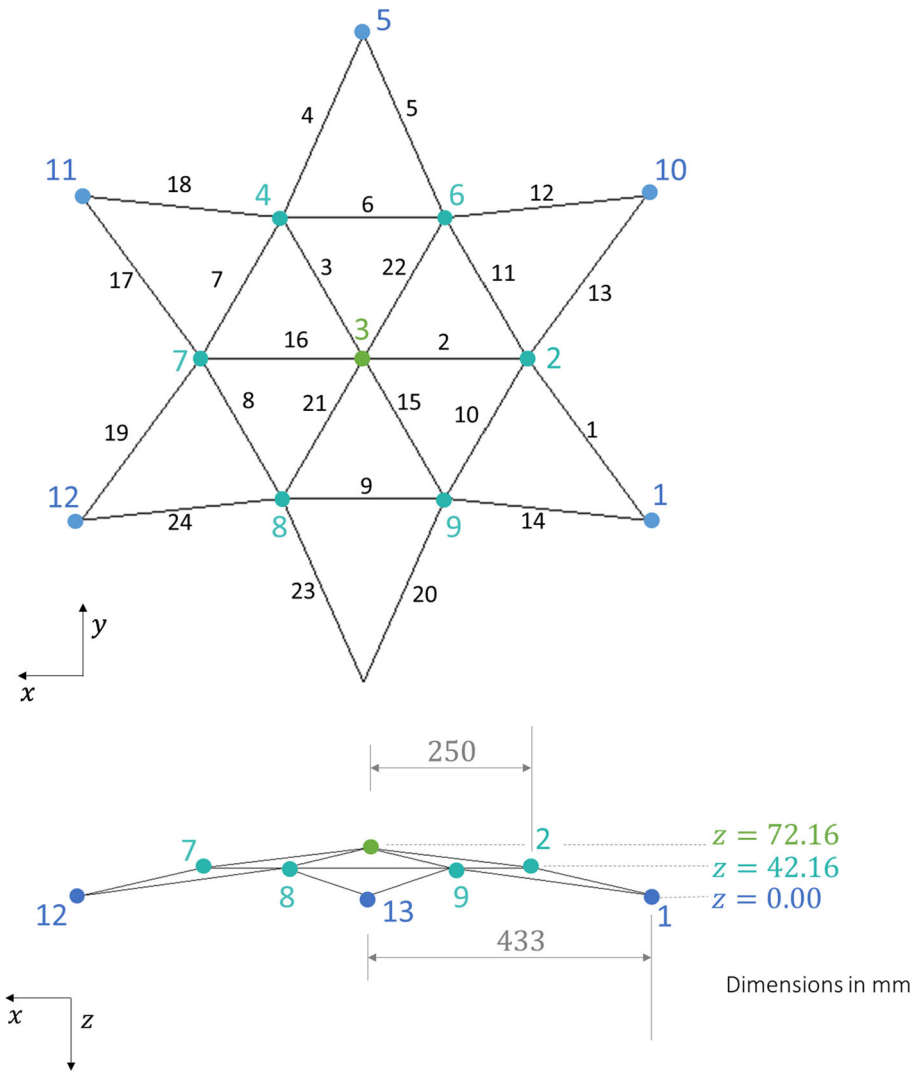

Fig. 1 Top and side view of the 3D model for 24-bar dome-like structure

ing piecewise plasticity in this context is not only frequently realistic, but also computationally efficient.

Stress strain relationship for the elastic state:

$$\sigma = E\left(\epsilon - \epsilon^p\right).$$

Isotropic hardening evolution law:

$$\dot{\epsilon}^p = \gamma \frac{\sigma}{\|\sigma\|}$$
$$\dot{\alpha} = \gamma.$$

Yield surface equation:

$$\phi(\sigma, \alpha) = \|\sigma\| - N(\alpha) \leq 0.$$

Complementary Kuhn–Tucker conditions:

$$\gamma \leq 0 \qquad \phi(\sigma, \alpha) \leq 0 \qquad \gamma\phi(\sigma, \alpha) = 0.$$

Consistency condition:

$$\gamma\dot{\phi}(\sigma, \alpha) = 0.$$

Box 1: One-dimensional elastoplastic model

The Abaqus software employs path-following techniques to provide a numerical solution for the structure with various area combinations. The curves depicting the reaction force versus the displacement imposed on the central node are illustrated in Figs. 2 and 3.

As shown in Fig. 2, the nonlinear structure proposed here is prone to *snap-through* type instability. The trajectory of loading process is stable until it reaches the critical state at the point B, leading to a snap through along the loading direction. If the process was force controlled, the response jumps to the next equilibrium configuration (E), causing the system to rapidly assume a tensile configuration. In displacement control, positive reaction forces occur on segments ABC and EF, while a negative force is observed on segment CDE.

The mechanical behavior can vary significantly based on the assigned cross-sectional areas for each bar. In the case of a *snap-through* type instability, most responses can be divided into three groups, as illustrated in Fig. 3. If the mechanical responses of the bar are mostly elastic, the behavior is as

Table 2 True stress vs equivalent plastic strain for the bars

True Stress $[MPa]$	† α $[-]$
660	0.0000
706	0.0247
754	0.0488
834	0.0953
897	0.1398
940	0.1823

† Plasticity was modeled as piecewise linear

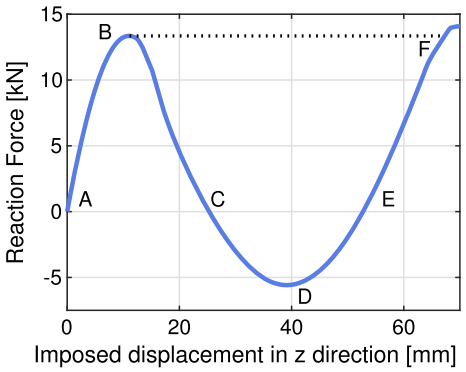

Fig. 2 Snap-through instability

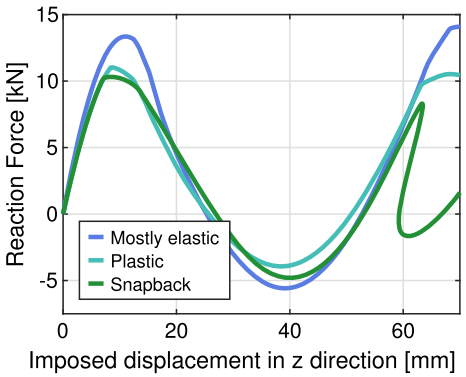

Fig. 3 Different behaviors for different area combinations

explained before in Fig. 2. If there is considerable plastic deformation, the behavior can be exemplified by lower displacement values at the B point, with a typical yield starting point that separates the elastic and elastoplastic parts, such as in a strain–stress curve. Depending on the combinations of areas, the structure shows snapback behavior. Differences in the evolution of the displacement profile during simulation, for plastic behavior with and without snapback, are presented in Fig. 4.

3 Methodology

Figure 5 elucidates the adopted methodology. The starting point is the basic finite element model of the elastoplastic 3D structure, as defined previously. An Abaqus script automatically runs the following loop: generates 24 area combinations; sets them to the bars; runs the FEA; records the force reactions vs imposed displacement for the structure. The repetition of this loop generates the data used to train the DNN. The Machine Learning step presents the neural network architecture and hyperparameters. Finally, the results obtained via AI are compared with those from FEA.

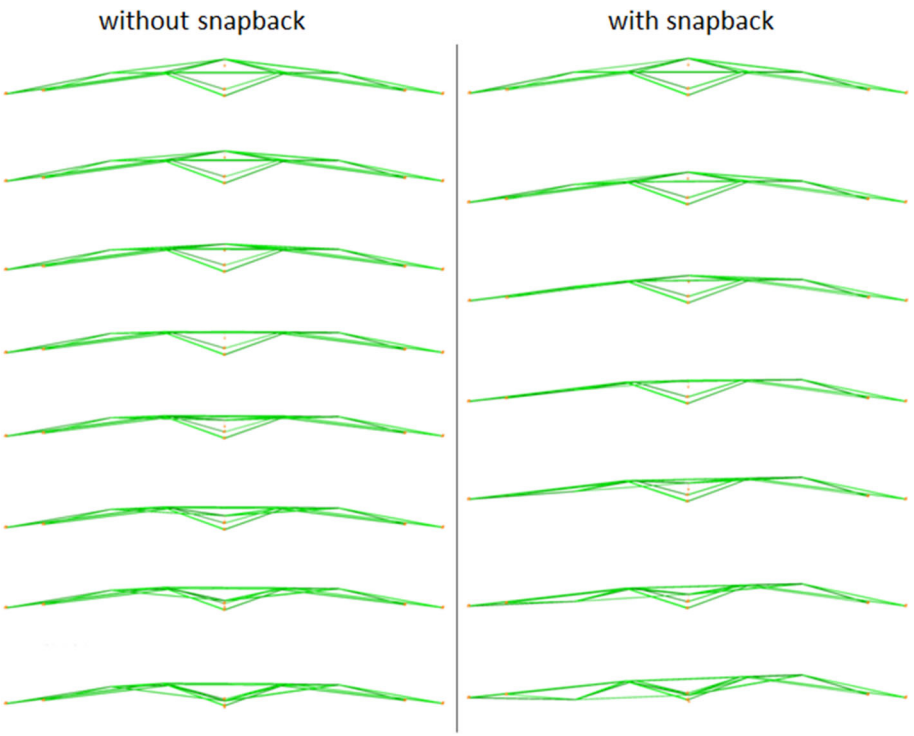

Fig. 4 Side view of the displacement evolution in two different structures. In the left snapback does not occur, while in the right the structure presents snapback in the global force–displacement response

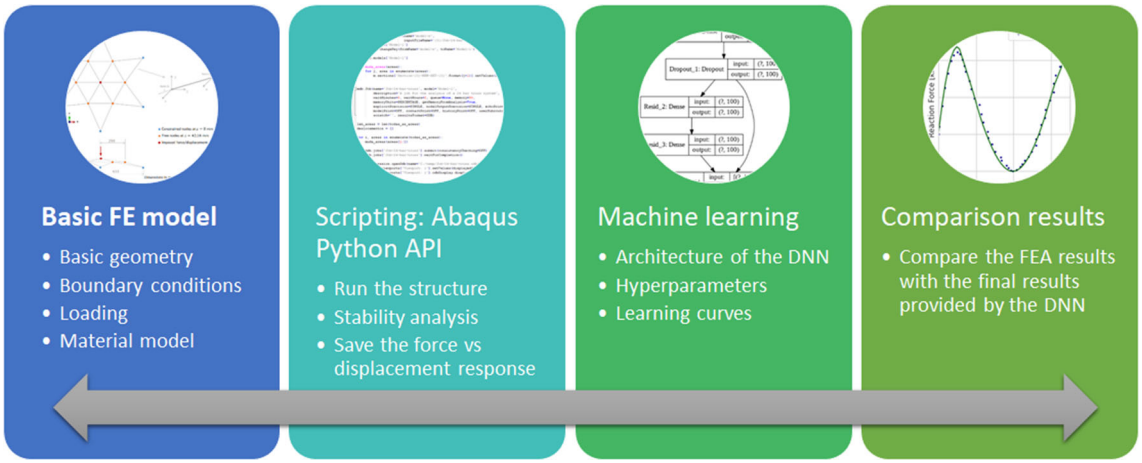

Fig. 5 Proposed methodology to join AI and structural analysis. The main four steps embrace: the finite element model and analysis, the machine learning technique via DNN, and comparison of the results to validate the approach

3.1 FE model

As suggested in Lee et al. (2018), the length and material model of the structure are maintained and the cross-section areas for the 24 bars are randomly defined for each model. Geometry is defined by a random set of 24 bars of element type TRUSS with different cross-sectional areas varying from $25-85\ \text{mm}^2$. A total of 10,000 combinations were generated. The area distributions for three different bars are illustrated in Fig. 6. The figures show that the number of times each bar has a specific cross-sectional area is approximately equally distributed within the adopted range. Table 3 shows the statistics with respect to the geometry, for three different analyzed structures.

Figure 7 illustrates the geometric difference among the models with and without snapback behavior. Snapback is present when the plastic state of the thinner bars is less expressive than the elastic unloading at the thicker bars in the global response.

In Fig. 8 the structure IDs 100 (without snapback) and 156 (with snapback) show localized plasticity in the thinner bars. In structure with snapback the level of plasticity in the thinner central bar 16, with 25.378 mm^2, is sufficient to generate instability. In this case, most of the bars unload, since they

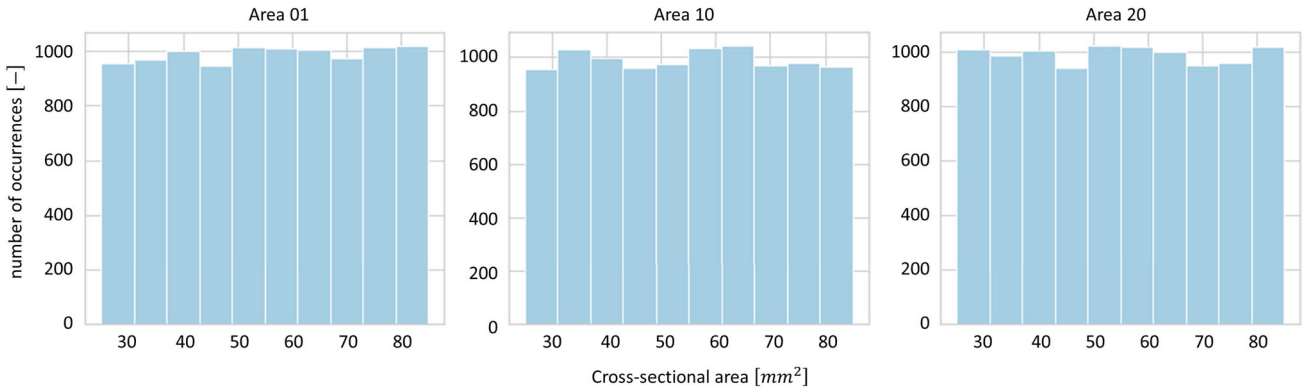

Fig. 6 The uniform distribution of the random cross-sectional area is illustrated, from left to right, for bars 1, 10, and 20

Table 3 Statistics of three different area combinations, in [mm^2]

Structure n.	Max.	Min.	Mean	Std Dev
21	84.47	25.28	55.73	20.16
26	74.77	29.57	56.97	13.00
1150	81.00	26.73	53.42	16.28

are still in elastic regime, and the global response presents snapback behavior. The loss of symmetry is detailed in Fig. 9.

There is also a rare fourth structural mechanical response, herein called *negative* behavior. In this negative response, the structure cannot reach a displacement of 70 mm at the central point, being thrown back on the other direction. This behavior can be illustrated by the structure $ID = 220$, in Fig. 10.

Due to instabilities in some *ad hoc* area combinations, 80 simulations could not be finished. The 10 simulations with the *negative* behavior were also not considered for the next steps. This resulted in 9910 analyzes, with 2169 observed snapback behavior (21.89% of total).

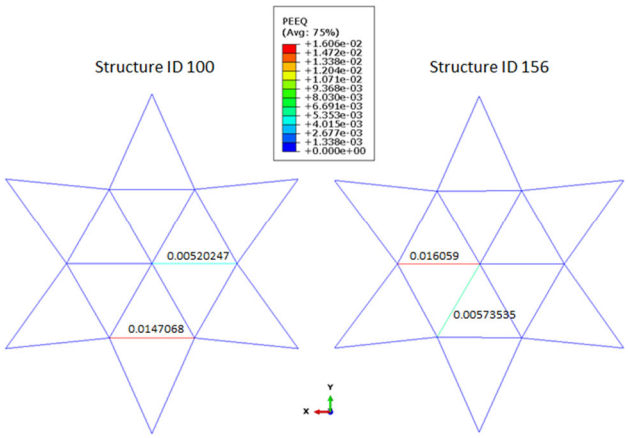

Fig. 8 Equivalent plastic strain at the end of the analysis for structure IDs 100 and 156

3.2 Scripting

Figure 11 shows the script proposed to generate the dataset to train the DNN, once the random combination of areas

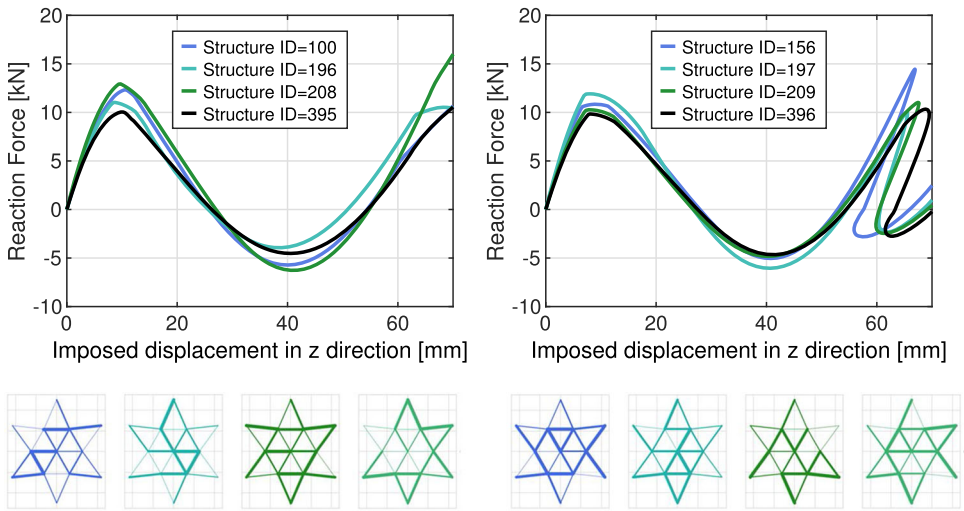

Fig. 7 Force vs displacement curves and area distribution in the structures: the larger the cross-sectional area, the thicker the bar representation. The area distribution is random and some configurations induce snapback behavior

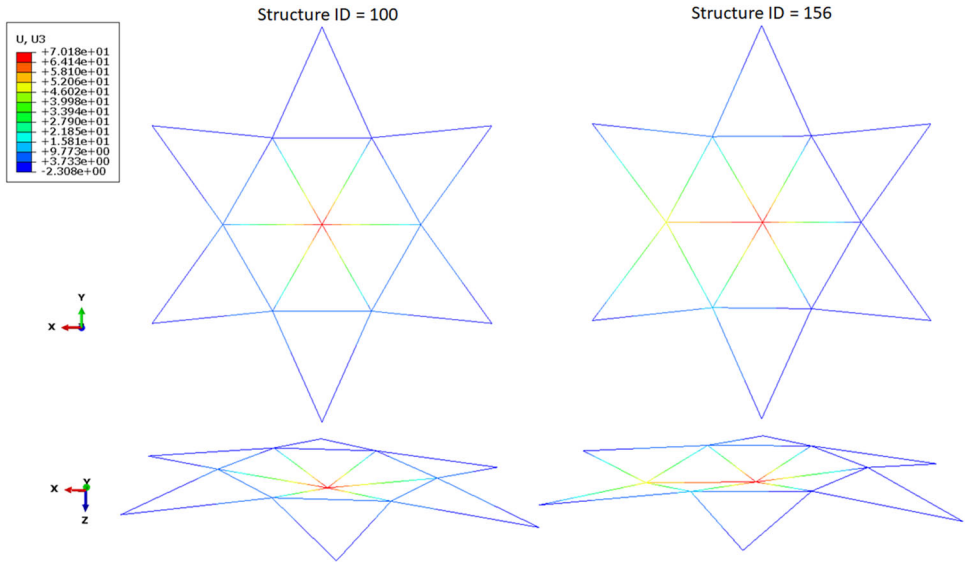

Fig. 9 Equivalent plastic strain at the end of the analysis for structure IDs 100 and 156

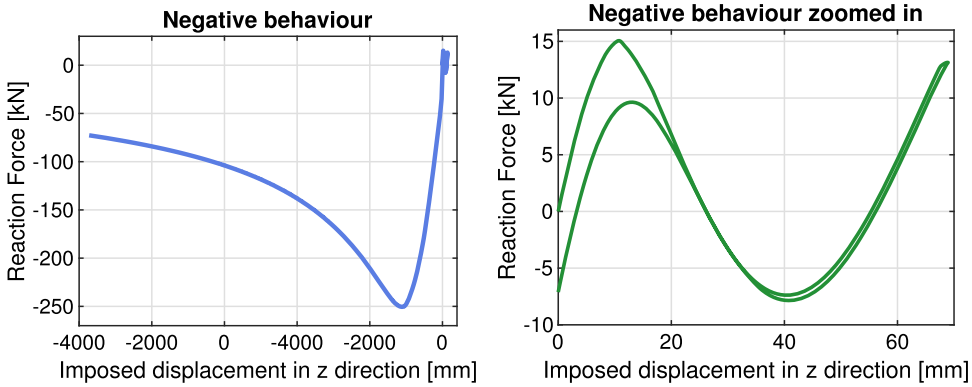

Fig. 10 Rare structural mechanical response, not considered for DNN training

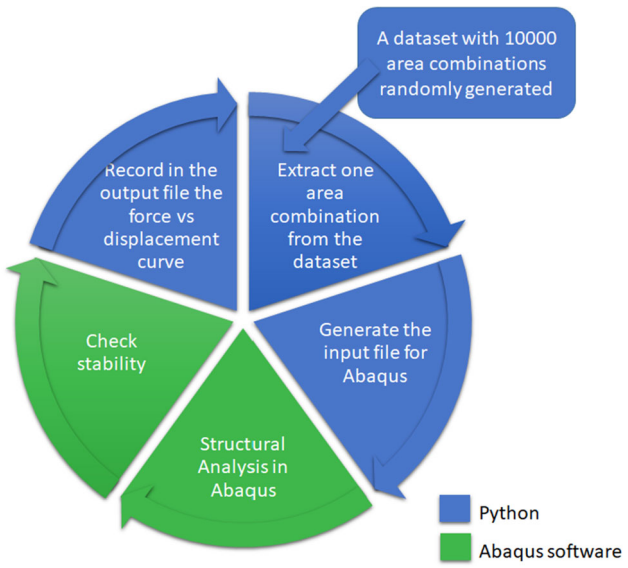

Fig. 11 Script for automatic generation of the output dataset

has already been generated and written in a `.csv` input file. A code in Python extracts a combination of areas from the input file and generates the file necessary to run Abaqus. After the analysis is finished, the forces and displacements are extracted from Abaqus and written in a `.csv` output file. The code runs automatically until the structure is analyzed with all area combinations defined in the input file. The script is composed of three codes available in Github.

Abaqus automatically determined the time-step based on the convergence criterion. Although a consistent displacement range of $w = 0 - 70$ mm in the z direction was maintained for all simulations, the number of points on the force–displacement curve varies for each dataset. To ensure uniformity, pre-processing of the input data is essential. Consequently, an interpolation process is employed to define points for all curves.Fig. 12 shows some examples of the processed output data. The structural response scenario highlights the inherent complexity of the problem, presenting a significant challenge for any AI process.

Table 4 shows three metrics for the first peak force—maximum, mean, and standard deviation of the original data and the post-processed data (50 points). The first peak force is considered, which is around $w = 10$ mm, when the snap-through initiates. The negligible difference between the

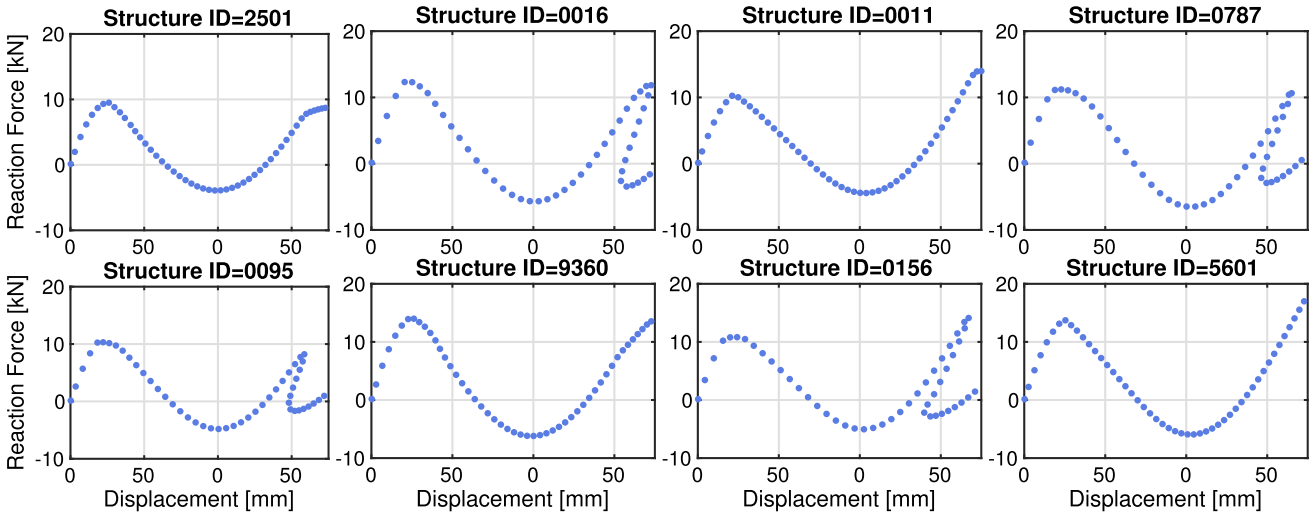

Fig. 12 Random examples of the output data for train the DNN

Table 4 Difference between original and preprocessed results

Metric	FEA Original	† 50 Points	Error (%)
Maximum	17.194[1]	17.7186[1]	0.0045
Mean	11.4947	11.4392	0.4837
Standard deviation	1.43013	1.41372	1.1474

[1] Maximum peak force found in the structure ID 6331 † values in kN

original and post-processed data is explained by the significant decrease in the number of points used in defining the curve. While some curves of the original data reach more than 3000 points, the processed data have, for all structures, 50 points properly interpolated. The force unit was changed to kN, to avoid a considerable difference between displacement and force magnitude for the output. This difference in output magnitude could negatively impact network performance.

3.3 Machine learning: deep neural network

As illustrated in Fig. 13, a DNN is composed of one input layer (blue), several hidden layers (yellow), and one output layer (green). Each layer has a defined number of artificial neurons, represented in the figure by the circles. Arrows show how a *dense* deep neural network is connected, that is, how all neurons are interconnected and how the input data flows from the input layer all the way to the output layer. Figure 13 shows also the perceptron, the very basic ingredient of any artificial neural network. Each ith neuron of the lth hidden layer receives a weighted sum of the neurons from previous layer,

$$a_i^{[l]} = \sum_{k=1}^{n} w_{ik}^{[l]} x_k + b_i^{(l)} = \mathbf{W}^{[l]}\mathbf{x} + \mathbf{b}^{[l]}, \tag{1}$$

where $w_{ik}^{[l]}$ are the weights, $b_i^{[l]}$ is the bias and n is the number of neurons of the previous layer.

Moreover, the activation function is applied to the neuron value, determining whether this neuron should be activated or not. The hyperbolic activation function ($tanh$) is used here, and it is defined as,

$$z_i^{[l]} = tanh\left(a_i^{[l]}\right) = \frac{e^{a_i^l} - e^{-a_i^{[l]}}}{e^{a_i^{[l]}} + e^{-a_i^{[l]}}}. \tag{2}$$

A large number of hidden layers define a DNN. Although, in general, deeper neural networks can result in models with better performance, the definition of the number of layers is a trade off between overfitting and underfitting of the model. To analyze whether the model is ideally fitted, the data set is divided into train and validation sets, 80% train, and 20% validation is a common rate. While the training set is the collection of data points the model learns from, the validation set is used to analyze how well the model generalizes the problem, which means the model's ability to give correct outputs to input sets it has never seen before. Complex problems with small NNs can show underfitting, that is, the model poorly represents the problem for the train or validation data. However, deeper neural networks are prone to overfitting because of the millions of parameters they enclose. Overfitting is the case where the overall cost is significantly small, but the generalization of the model is poor. This is due to the model intensively learning from the training data set and memorizing them, instead of learning.

Figure 13 presents a simple sequential architecture for a DNN. The architecture of the proposed DNN is far more

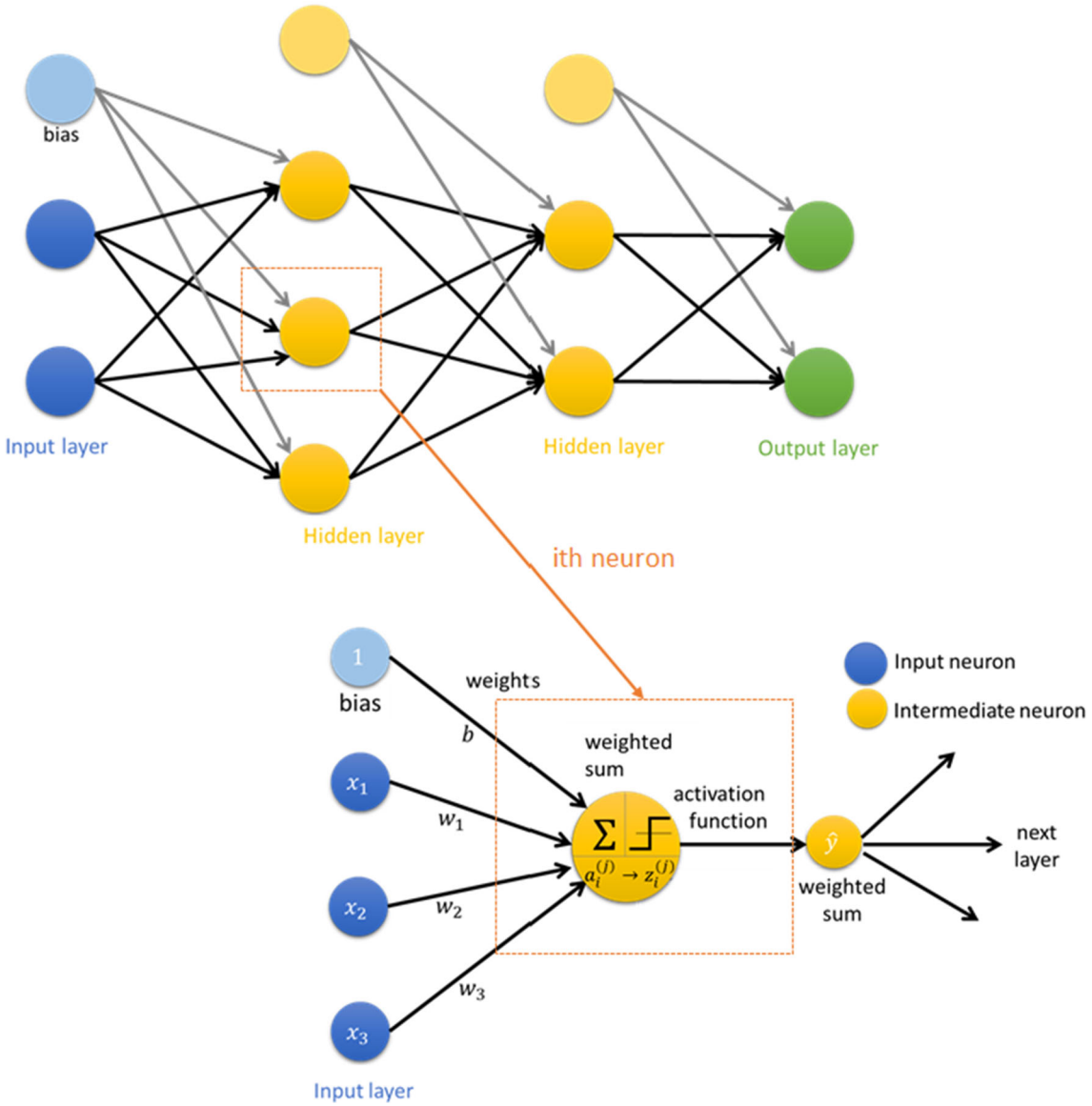

Fig. 13 The Artificial Neural Network (ANN) is composed by input, hidden, and output layers. Each hidden layer has a defined number of neurons. The neuron receives weighted values from the neurons in the previous layer, applies the activation function and sends the result to the neurons in the next layer

complex. Although different models were tested, only the architecture with the best performance is discussed here.

Figure 14 presents three basic blocks. The first block, called *Sequential*, contains four dense layers, each one with batch normalization and hyperbolic tangent as activation function. Batch normalization layers are used in training deep neural networks to normalize each batch of data during each epoch.

To reduce overfitting in DNNs, He et al. (2016) proposed a *residual learning framework*, well known as *ResNet*. The technique eases the training of convolutional networks that are substantially deep and is applied here to the proposed deep dense network. The *Identity* and *Dense* blocks in Fig. 14 present the *ResNet* framework, with a shortcut between two non sequential layers of the network similar to those suggested by Chen et al. (2020) for convolutional neural networks. Shortcut connections can just copy the input, as in the Identity block, or they can contain a dense layer, as in the Dense block. At the end, the shortcut output is added to the output of the stacked layers. The *Mixed* block presented in Fig. 15 combines the basic blocks from Fig. 14 to compound the final DNN.

Various techniques and hyperparameters, including dropout, were tested for addressing overfitting. The final proposed architecture incorporates ResNet with L2 regular-

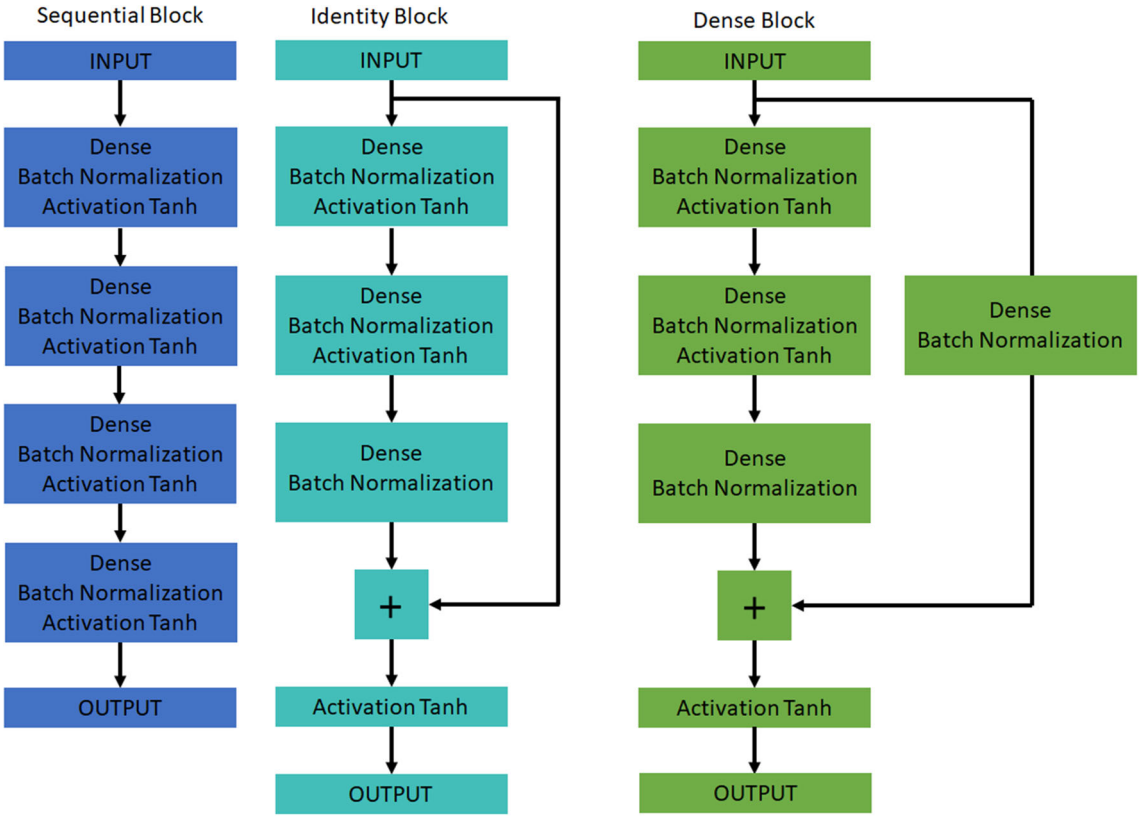

Fig. 14 The primary blocks inserted in the final architecture of the proposed DNN

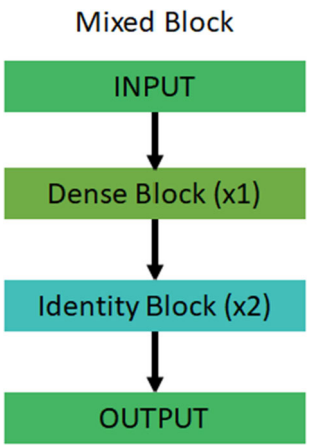

Fig. 15 Mixed block inserted in the final architecture of the proposed DNN

ization and batch normalization, which has been shown to be more efficient.

The final DNN structure is illustrated in Fig. 16 and presents a total of 8, 338, 300 parameters. The DNN contains a *Sequential* block, followed by 3 groups of 7 *Mixed* blocks each and 2 groups of 6 *Mixed* blocks each. It is important to point out that the DNN has external bypasses, similar to those found in the *Identity* blocks. Hyperparameters are set before training. Some hyperparameter values determine the network structure, for example, the number of hidden layers or neurons per layer, and activation functions; and others define the training procedure, for example, learning rate or number of epochs. These hyperparameters dictate how the DNN learns the parameters (weights and bias) from the data, during the training process.

In the sequential block, the four layers contain, respectively, $600 - 450 - 300 - 300$ neurons, while the Identity and Dense block layers are composed of 150 neurons. The dimension of the output of the DNN is 100, 50 displacement values and the 50 corresponding force values. Due to the high nonlinear response behavior, activation ReLU (Rectified Linear Unit) or Leaky ReLU, more frequently adopted activations, did not work well. Since the values can be negative, the hyperbolic tangent is more suitable than the sigmoid. Therefore, hyperbolic tangent was chosen as the activation function for all layers. The learning rate, necessary to the neural network to perform the gradient descent algorithm, is defined as $\alpha = 0.05$, with 15000 epochs and batch size 1024. Adam optimizer (short for ADAptive Moment estimation) was adopted (Kingma and Ba 2014).

The mean squared error, i.e., the average squared difference between the estimated values from DNN and true values from FEA, was chosen as loss function,

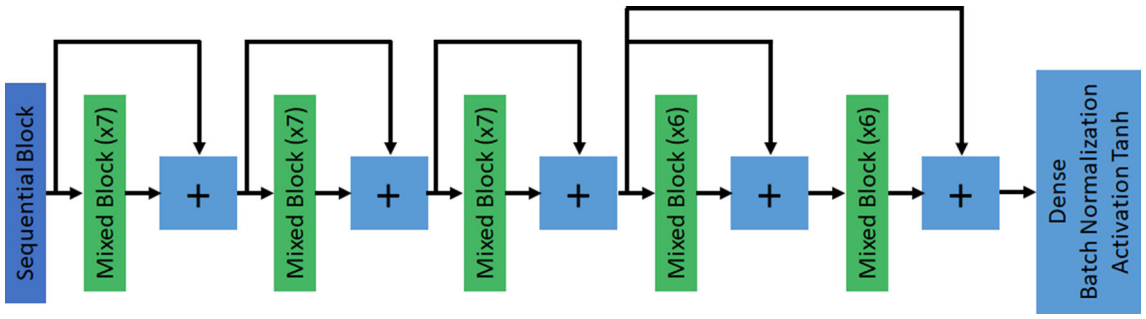

Fig. 16 The proposed architecture to the DNN to solve the 24-bar dome

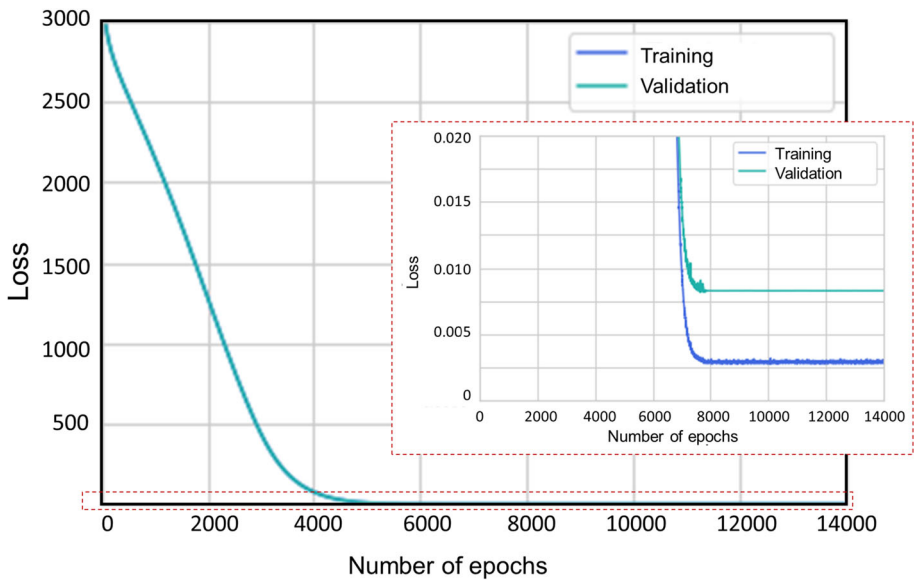

Fig. 17 The learning curve obtained during training the DNN

$$L\left(\mathbf{W}, \mathbf{b}, \mathbf{x}_j, \mathbf{y}_j\right) = \|\mathbf{y}_j - \hat{\mathbf{y}}_j\| = \|\mathbf{y}_j - \left(\mathbf{W}^{[L]}\mathbf{x}_j + \mathbf{b}^{[L]}\right)\|, \tag{3}$$

where the subscript j refers to the j-th instance in the dataset; the superscript $[L]$ is the last layer; $\hat{\mathbf{y}}_j$ is the vector of predicted values (output in the last layer); $\mathbf{y}_j$ is the vector of target values. Therefore, the regularized cost function is defined as

$$J\left(\mathbf{W}, \mathbf{b}\right) = \frac{1}{N}\sum_{j=1}^{N} L\left(\mathbf{W}, \mathbf{b}, \mathbf{x}_i, \mathbf{y}_i\right) + \frac{\lambda}{2N}\sum_{l=1}^{L}\|\mathbf{W}^{[l]}\|_F^2, \tag{4}$$

where N is the dataset length, Frobenius norm is denoted by the subscript F, λ is the regularization hyperparameter tuned as $\lambda = 0.10$ to minimize overfitting.

During DNN training, the parameters $\mathbf{W}^{[l]}$ $\mathbf{b}^{[l]}$, $l = 1, ..., L$, are chosen to fit the dataset in a manner that minimizes the cost function defined in Eq. 4.

4 Results

The results obtained for the 3D star-like bar structure using the neural network presented in the previous section are detailed here.

4.1 Learning curve

The learning curve in Fig. 17 the evolution of learning performance, as indicated by the loss value over time with respect to the number of epochs. These curves, generated for the train and validation datasets, serve to identify under- or overfitting in the model.

Fig. 17 demonstrates that the validation loss closely follows the training loss, with a slight increase. Although overfitting is observed in the zoom detailed in Fig. 17, small absolute loss values show that the train and validation datasets are representative of the problem domain.

Once the Deep Neural Network (DNN) was trained, predictions were generated using both the test and validation

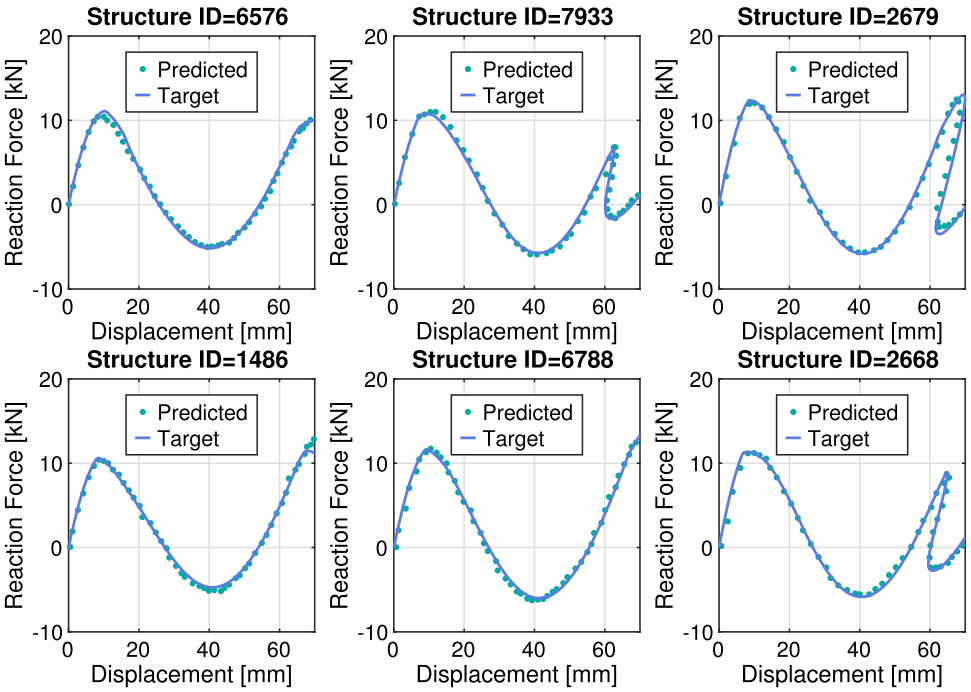

Fig. 18 Train dataset performance of DNN for some structures

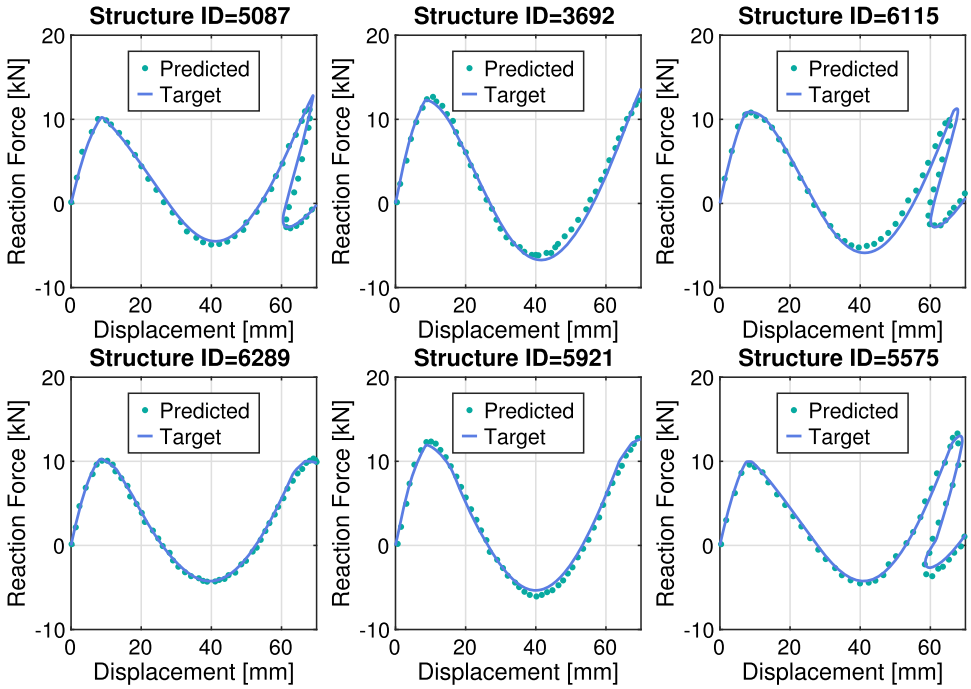

Fig. 19 Validation dataset performance of DNN for some structures

sets. These predictions were then compared with the values derived from structural analyzes performed with Abaqus A graphical comparison is shown in Figs. 18 and 19. It can be seen that the predicted and real curves are closely related. As expected, the error is smaller for the train set (Fig. 18) compared to the validation set (Fig. 19).

The metrics—maximum value, maximum error, mean, and standard deviation—are summarized in Table 5 considering the peak force value. The mean and standard deviation

Table 5 Metrics about the differences in the first peak force values

Metrics	FEA	DNN	
		Training	Validation
Maximum peak force† $[kN]$	17.7186	16.7861	15.5423
Mean $[kN]$	11.4392	11.3977	11.3468
Standard deviation $[kN]$	1.41372	1.32526	1.21376
Maximum error $[kN]$	–	2.09917[1]	2.88492[2]

[1] Structure ID 5702 [2] Structure ID 1473, see Fig. 20

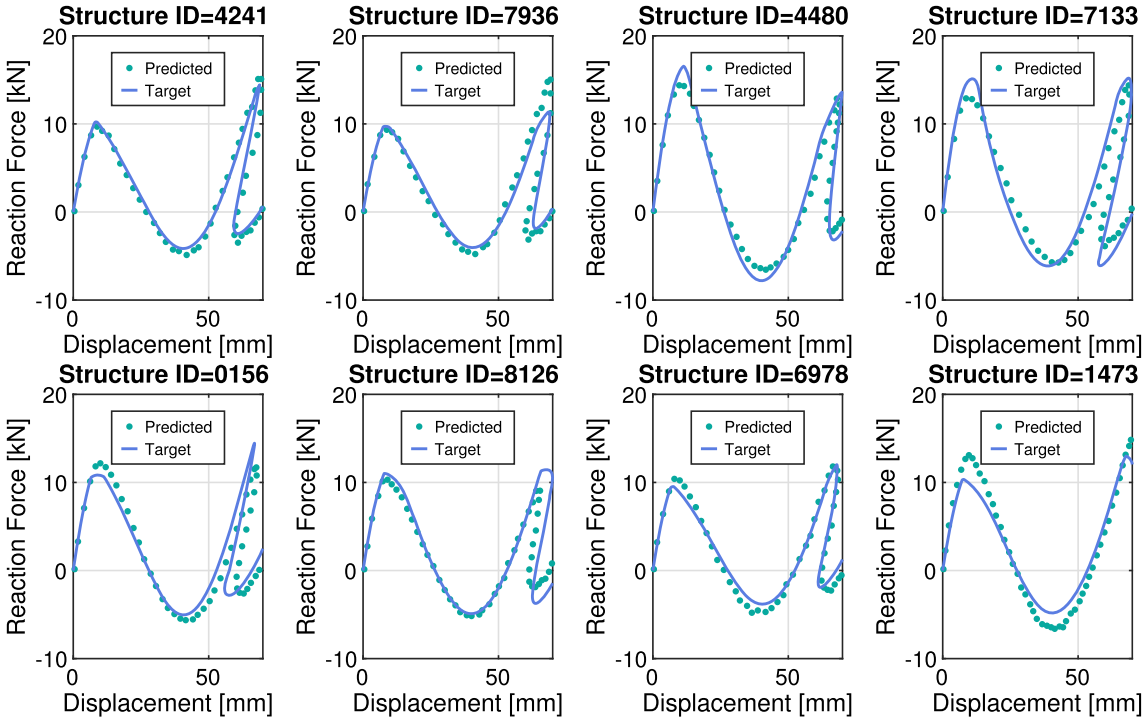

Fig. 20 Curves with higher deviation than average, obtained from train (upper curves) and validation (lower curves) sets

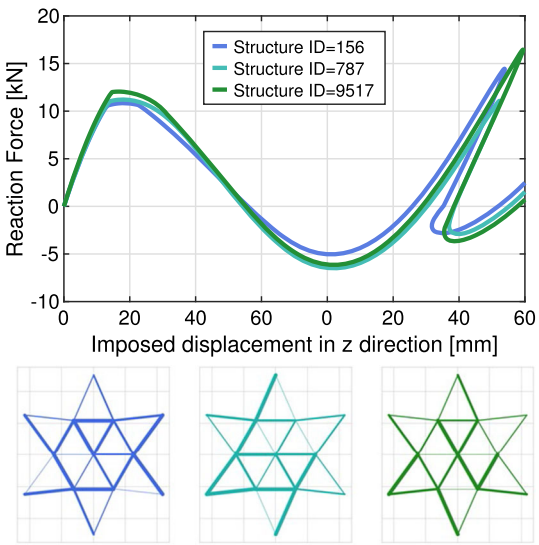

Fig. 21 Curves with higher deviation than average, obtained from train (upper curves) and validation (lower curves) sets

indicate a close relationship between the majority of results and the target curve. The maximum error is the largest difference between the maximum force predicted by the model and the target values from the FEA. Notably, the error in the training dataset is smaller than in the validation set. The structure related to the maximum error, $ID1473$, is illustrated in Fig. 20. Despite the challenging nature of learning the snapback phenomenon, the most deficient performance in terms of load peak occurs when it significantly deviates below the average. This is attributed to the premature onset of the plastic regime within the structure. It happens when the structure contains a substantial number of bars with low cross-sectional area, as exemplified by $ID395$ in Fig. 7.

With snapback occurring in only 21.89% of cases, we expect the Neural Network (NN) to perform less effectively in predicting this phenomenon compared to cases exclusively involving snap-through. The instances where the Deep Neural Network (DNN) shows suboptimal performance, as illustrated in Fig. 20, are primarily associated with snapback. Notably, in situations characterized by pronounced snapback, as shown in Fig. 21, the DNN model exhibits satisfactory responses, as exemplified in Fig. 22.

5 Conclusions

A total of 10,000 combinations of 24 cross-sectional areas randomly selected were generated and each combination was individually imputed to a FE dome-shaped truss model with fixed material parameters and dimensions. Nonlinear numerical analyses of all these geometries were performed using a commercial software, with the load displacement curves recorded as output. Despite the truss nature of the so-formed dome shape, the generated responses were seen to be a very complex regression problem.

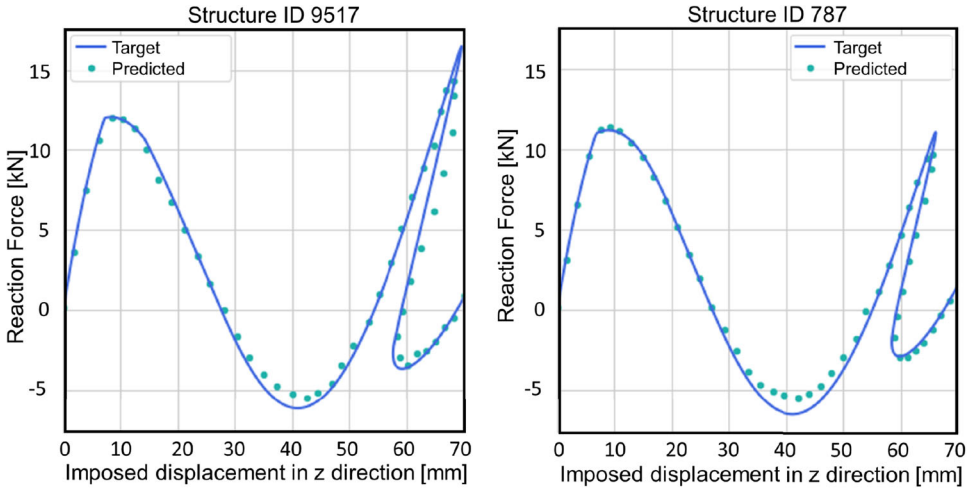

Fig. 22 Example of DNN model response for severe snapback cases

A DNN was trained, with the geometry as input and 50 points of the curve vs force–displacement as output. To enhance the performance of the proposed DNN, several steps were taken, including architectural refinement, hyperparameter tuning, and expansion of the database. The residual strategy, commonly employed in convolutional neural networks, has been successfully implemented in the proposed dense neural network to address the challenges associated with a complex regression problem. The final DNN model's results closely match those from numerical analyses. Notably, the global response of the neural network during snapback emphasizes the importance of the dimension and balance of the training dataset. Ensuring that the training data closely represent the actual data is crucial for the neural network's robust performance. The work presented here is of a deterministic nature, and an extensive dataset was necessary to encapsulate material nonlinearity and complex geometric effects. As expected, the results are not easily applicable to other structures, and their effectiveness heavily depends on data and results obtained from traditional finite element methods. The deep learning problem explored in this context highlights that the neural network can learn to estimate the force–displacement curve of a dome-shaped structure, albeit at a considerable initial training The results suggest that using DNN as a substitute in Finite Element Analyses (FEAs) can be justified in certain situations. For instance, in inverse problems where experimental results with inherent data dispersion guide the design. In such cases, conventional structural problem models may not precisely match the data, making NN a promising area that merits further exploration. Additionally, for optimization problems of high computational cost per iteration, DNNs can be a good alternative, as often recommended in the literature. Large-scale numerical simulations are increasingly used to replicate complex engineering problems in various domains, from mechanical to biomedical. Particularly in domains like robot stability and safety, real-time predictions are essential. Despite the intensive computations required during the initial stages of data collection and model training, machine learning methods exhibit exceptional speed once adequately trained.

The technique developed here emerges as a promising approach, seamlessly integrating artificial intelligence and Finite Element methods for the analysis of complex structures.

Software and scripts

Abaqus/CAE™ 2019 Student Edition was adopted to obtain FE results. The Jupyter notebook with Python code for DNN training and results analysis is available in Github.

Acknowledgements We thank prof. Thiago Martins, from Complex Systems Laboratory of University of São Paulo, for providing computer support.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data availability: The authors confirm that the data supporting the findings of this study are available within the article and its supplementary materials.

Declarations

Conflict of interest Author Larissa Driemeier declares that she has no conflict of interest. Author Eduardo Lobo Lustosa Cabral declares that he has no conflict of interest. Author Gabriel Lopes Rodrigues declares that he has no conflict of interest. Author Marcos Tsuzuki declares that he has no conflict of interest. Author Marcilio Alves declares that he has no conflict of interest. Author Lucas Pires da Costa declares that he has no conflict of interest. Author Rafael Traldi Moura declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abdeljaber O, Avc O, Kiranyaz S et al (2017) Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. J Sound Vib 388:154–170

Abiodun OI, Jantan A, Omolara AE et al (2018) State-of-the-art in artificial neural network applications: a survey. Heliyon 4(11):e00,938

Abueidda DW, Koric S, Sobh NA (2020) Topology optimization of 2d structures with nonlinearities using deep learning. Comput Struct 237(106):283

Alves M (2020) Impact engineering: fundamentals, experiments and nonlinear finite elements, 1st edn. https://doi.org/10.4322/978-85-455210-0-6

Arnold F, King R (2021) State-space modeling for control based on physics-informed neural networks. Eng Appl Artif Intell 101(104):195

Arora R, Jacobson A, Langlois TR, et al (2018) Designing volumetric truss structures. arXiv:1810.00706v3

Bilal PM, Zaheer H et al (2020) Differential evolution: a review of more than two decades of research. Eng Appl Artif Intell 90(103):479

Chen D, Hu F, Nian G, et al (2020) Deep residual learning for nonlinear regression. Entropy 22(2)

Driemeier L, Proença SPB, Alves M (2005) A contribution to the numerical nonlinear analysis of three-dimensional truss systems considering large strains, damage and plasticity. Commun Nonlinear Sci Numer Simul 10:515–535

Esteva A, Kuprel B, Novoa RA et al (2017) Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639):115–118

Gao T, Huang M, Wang Q et al (2018) A systematic model of stable multilateral automated negotiation in e-market environment. Eng Appl Artif Intell 74:134–145

Gu GX, Chen CT, Buehler MJ (2018) De novo composite design based on machine learning algorithm. Extreme Mechanics Letters 18:19–28

He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 770–778, https://doi.org/10.1109/CVPR.2016.90

Hencky H (2020) On the theory of plastic deformations and the residual stresses caused by them in the material. Journal of Applied Mathematics and Mechanics 100(3):e202002,019. https://doi.org/10.1002/zamm.202002019

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. CoRR arXiv:1412.6980 [cs.LG]

Koller L, Witteveen W, Pichler F et al (2021) A general hyper-reduction strategy for finite element structures with nonlinear surface loads based on the calculus of variations and stress modes. Comput Methods Appl Mech Eng 379(113):744

Lee S, Ha J, Zokhirova M et al (2018) Background information of deep learning for structural engineering. Archives of Computational Methods in Engineering 25(1):121–129

Li Y, Wang S, Han M (2019) Truss structure optimization based on improved chicken swarm optimization algorithm. Advances in Civil Engineering 2019

Liang L, Minliang L, Caitlin M, et al (2018) A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. J R Soc Interface 1520170844(15)

Liao X, Zheng X, He J et al (2021) Computer-aided decision-making system for endometrial atypical hyperplasia based on multi-modal and multi-instance deep convolution neural networks. Soft Comput. https://doi.org/10.1007/s00500-021-06576-6

Lukin N, Moura RT, Alves M et al (2020) Analysis of api s-135 steel drill pipe cutting process by blowout preventer. J Petrol Sci Eng 195(107):819. https://doi.org/10.1016/j.petrol.2020.107819

Ma N (2021) Analysis of industry convergence based on improved neural network. Soft Comput. https://doi.org/10.1007/s00500-021-06439-0

Ozbasaran H, Eryilmaz Yildirim M (2020) Truss-sizing optimization attempts with csa: a detailed evaluation. Soft Computing 24(22):16,775–16,801. https://doi.org/10.1007/s00500-020-04972-y

Rezaiee-Pajand M, Momenipour M, Hozhabrossadati SM (2020) A novel grey prediction evolution algorithm for multimodal multiobjective optimization. Engineering with Computers. https://doi.org/10.1007/s00366-020-01209-2

Russell SJ, Norvig P (2009) Artificial Intelligence: a modern approach, 3rd edn. Pearson

Shakya A, Nanakorn P, Petprakob W (2018) A ground-structure-based representation with an element-removal algorithm for truss topology optimization. Structural and Multidisciplinary Optimization 58

Shao Y, Liu CL (2020) Teaching machines to write like humans using l-attributed grammar. Eng Appl Artif Intell 90(103):489

Sharma O, Sahoo N, Puhan N (2021) Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey. Eng Appl Artif Intell 101(104):211

Teng Z, Teng S, Zhang J et al (2020) Structural damage detection based on real-time vibration signal and convolutional neural network. Appl Sci 10:4720

Torky AA, Aburawwash AA (2018) A deep learning approach to automated structural engineering of prestressed members. International Journal of Structural and Civil Engineering 7(4):347–352

Vanluchene RD, Sun R (1990) Neural networks in structural engineering. Computer-Aided Civil and Infrastructure Engineering 5(3):207–215

Young T, Hazarika D, Poria S et al (2018) Recent trends in deep learning based natural language processing [review article]. IEEE Comput Intell Mag 13(3):55–75. https://doi.org/10.1109/MCI.2018.2840738

Zeng S, Hu Y, Xie X (2021) Q-rung orthopair fuzzy weighted induced logarithmic distance measures and their application in multiple attribute decision making. Eng Appl Artif Intell 100(104):167

Zhao H (2021) A reduced order model based on machine learning for numerical analysis: An application to geomechanics. Eng Appl Artif Intell 100(104):194

Zhou T, Hu Z, Zhou Q et al (2021) A novel grey prediction evolution algorithm for multimodal multiobjective optimization. Eng Appl Artif Intell 100(104):173