Structures 59 (2024) 105712

Contents lists available at ScienceDirect

Structures

journal homepage: www.elsevier.com/locate/structures

On the use of mechanics-informed models to structural engineering systems: Application of graph neural networks for structural analysis

Fabio Parisi [a,b,*], Sergio Ruggieri [b], Ruggiero Lovreglio [c], Maria Pia Fanti [a], Giuseppina Uva [b]

[a] DEI Department, Polytechnic University of Bari, Via Orabona 4, Bari, Italy
[b] DICATECH Department, Polytechnic University of Bari, Via Orabona 4, Bari, Italy
[c] School of Built Environment, Massey University, Auckland, New Zealand

ARTICLE INFO

Keywords:
Mechanics-Informed Model
Graph Neural Network
Structural Engineering

ABSTRACT

This paper investigates the application of mechanics-informed artificial intelligence to civil structural systems. Structural analysis is a traditional practice that involves engineers to solve different real-life problems. Several approaches can be used for this task, going from "by hand" computation to the recent advanced finite element method. However, when structures become complex, the success of the analysis can be complicated, often requiring high computational efforts and time. To tackle this challenge, traditional high-demanding methods can be supported by new technologies, such as machine-learning tools. This new paradigm aims to solve structural problems by defining the desired output after directly elaborating input data. One of the current limitations is that often the physics behind the problem is ignored. To solve this issue, resolution models can combine empirical data and available mechanics prior knowledge to improve the predictive performance involving physical mechanisms. In this paper, a method to develop a Mechanics-Informed Surrogate Model (MISM) on structural systems is proposed, for which input structured data are used to enrich the informative content of mechanics systems. Then, Graph Neural Networks (GNNs) are explored, as a method capable of properly representing and embedding knowledge about a structural system, such as truss structures. The main advantage of the proposed approach is to provide an alternative way to the usual black-box machine-learning-based models. In fact, in the proposed MISM, the mechanics of the structural system plays the key role in the surrogate model definition, in order to obtain physically based outputs for the investigated problem. For the case at hand, MISMs are developed and employed to learn the deformations map of the system, starting from the knowledge of the structural features. The proposed approach is applied to bi-dimensional and tri-dimensional truss structures and the results indicate that the proposed solution performs better than standard surrogate models.

1. Introduction

Over the last years, the rise to prominence of new technologies related to Machine-Learning (ML), Deep Learning (DL) and, more in general, Artificial Intelligence (AI) have influenced several traditional disciplines, including structural engineering [1,2]. In particular, these new techniques were applied to the fields of design and optimization of new structures [3–5], assessment of the health state of the existing ones [6–15], the effects of human-induced and natural hazards on new and existing structures [6,16–18]. Different ML solutions were proposed for dealing with these specific structural engineering problems, such as neural networks [19], response surface models [20] and kriging models [21]. These ML solutions are usually used as meta-models fitting structural data to simulate the desired response and represent the so-called Surrogate Models (SMs) in the literature. Many examples of this methodological approach are available. In [22], authors employed a ML-based surrogate model to optimize truss structure considering its nonlinear behaviour. In [23], the authors dealt with the seismic assessment of buildings and proposed an ML-based surrogate model framework considering both structural and seismic uncertainties to predict structural outputs for unseen earthquakes. In [24], authors investigated an adaptive risk-based life-cycle management for large-scale structures, using deep neural networks to configure surrogate modelling for structural risk assessment. In [25], a surrogate finite element approach based on a neural network was also proposed to estimate the time-varying response of a one-dimensional beam. In [26],

* Corresponding author at: DEI Department, Polytechnic University of Bari, Via Orabona 4, Bari, Italy.
E-mail address: fabio.parisi@poliba.it (F. Parisi).

https://doi.org/10.1016/j.istruc.2023.105712
Received 31 July 2023; Received in revised form 3 November 2023; Accepted 4 December 2023
Available online 14 December 2023

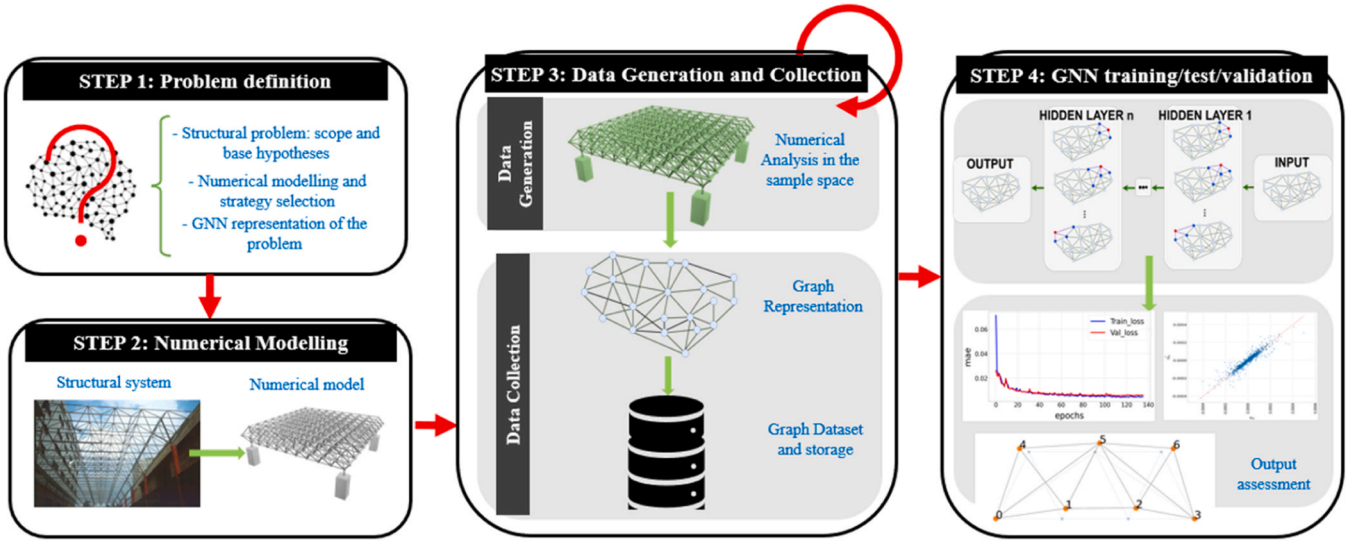

Fig. 1. Flow chart of the proposed methodology.

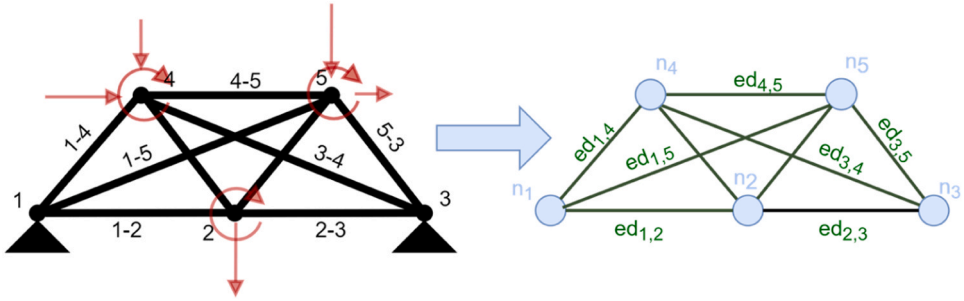

Fig. 2. Example of the representation of a simple truss structure as a graph: the load conditions are in red, the starting structural model is in black, while nodes and edges in the graph representation are in blue and green respectively.

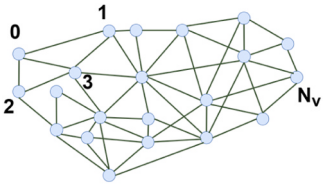

Fig. 3. Random GNN scheme. Nodes 1, 2 and 3 are neighbours for node 0.

Monte Carlo simulations and a machine learning-based SM were adopted to calibrate the reliability of slab-column joints response in a real engineering example. Moreover, the SM-based approaches are widely used in the seismic assessment of existing structures with fragility analysis, in which most of different methodological phases are interpreted with ML to reduce computational burden [27–29].

SMs represent one of the most promising ML applications in structural engineering as they show better performance than traditional numerical methods in terms of computational efficiency. However, when these models are used without a mechanics-based approach, they generate issues of unidentifiability. In fact, ignoring the theoretical and physical foundation of structural problems could generate a loss of contact with the ground truth [30]. Using SMs to solve this type of problem can considerably reduce the computational effort (e.g., in complex structures, the number of equations to solve, also with a Finite Element Model, FEM, could consistently increase). On the other hand, the results could be far from the exact ones. The current trend is trying to assess the real sense of the SM by explainability metrics [14,31–33].

A possible improvement can be achieved by adding prior knowledge of the mechanics system into analysis methodologies, and, thus, by defining the Mechanics-Informed Surrogate Models (MISM). To this scope, the concept of Physics-Informed Neural Networks [34,35] can be exploited, as a novel neural network approach used to solve physical problems traditionally far from the field of AI (e.g., solid mechanics and fluid-dynamics [36]). A possibility in this direction is to develop ML models based on graph-structured data, representing physical information about the system to study (e.g., [37,38]). In this context, the solution is represented by the Graph Neural Networks (GNNs), which belong to the family of artificial neural networks and are capable of processing data as graphs [40]. This latter structure is a non-Euclidean representation (i.e., data are defined through their semantics and not through numbers, vectors, or matrix) of data in nodes and edges specifically featured by explicating relationships. The representative power of GNNs is related to the capability to describe relations among data and to provide additional informative content and features hardly implementable in standard DL and ML approaches [38]. The use of GNNs is a simple idea to apply to some structural systems, for example by associating a graph with nodes and edges with frames or trusses. In the authors' knowledge, very few examples are available in the engineering-related scientific literature (see for instance Song et al. [39]), while inspiring applications concerning physical problems are presented in [37,38], where the prediction of the displacements of mass-springs systems is used as an example of the GNNs potential.

This work aims to evaluate the efficiency of GNNs in a structural engineering problem. With this goal in mind, the paper presents a guided approach to create a new Mechanics-Informed Surrogate Model

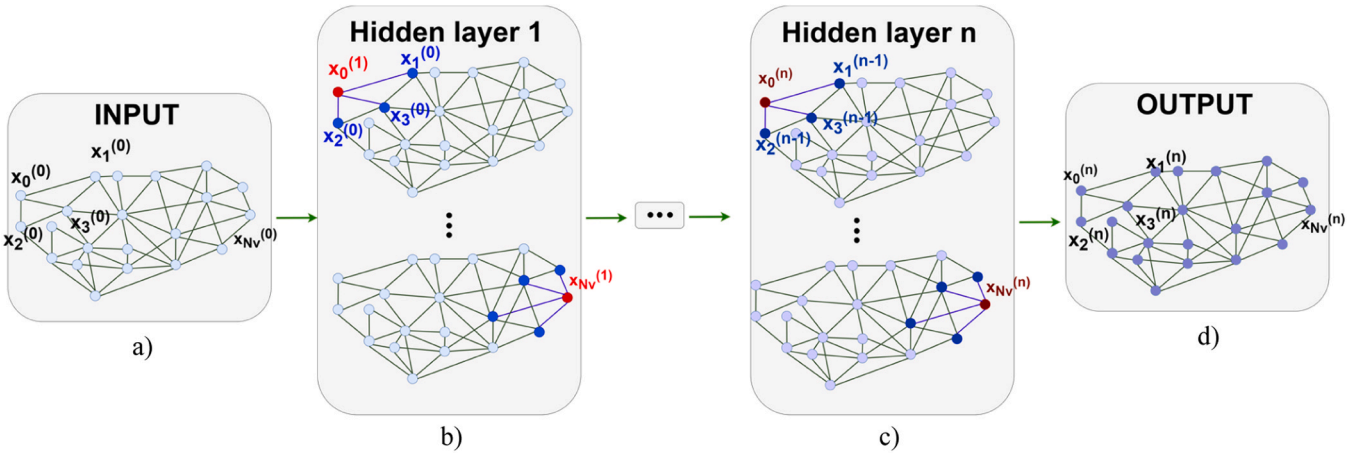

Fig. 4. General updating process of the components in GNNs.

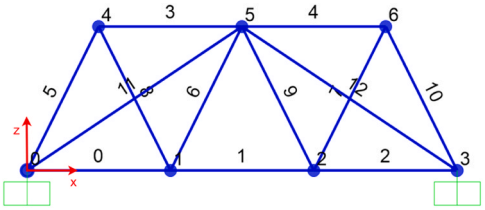

Fig. 5. Structural example model in the FE environment: 2D steel truss.

(MISM), as well as a SM developed exploiting the knowledge of the mechanics of the structural system. In detail, the paper reports all the steps to elaborate the MISM, starting from the data generation, going through the training of the new SM, up to the test and validation with the solution obtained by a traditional SM. The proposed approach was developed and applied to perform a linear analysis of two types of elastic truss structures (bi-dimensional, 2D, and tri-dimensional, 3D, cases), showing the performance of the MISM and the related potentialities. From the experience reported in the paper, new perspectives in the field of structural engineering could be open, first by supporting traditional analytical and numerical approaches (e.g., FEM) and after by improving the current practice in the resolution of complex problems through ML-based approaches.

2. Material and methods

Fig. 1 shows the proposed methodology described in four main steps, extensively discussed in the next section with a focus on the technologies and the approaches involved. Step 1 consists of the problem definition, in which the scenarios of use of the MISM are defined, and the problem to address is specified in terms of generalization requirements, i.e., scope and base hypotheses. This phase is of paramount importance due

Table 1
Random sections assigned to frames (from Fig. 5) with relative dimensions from Fig. 6: t_3 represents the section dimension in the z direction of the local reference system, t_2 the section dimension in the y direction of the local reference system, tf is the thickness of the section and d is the distance between different components of the section in the case of double angle ones.

Frame	Shape	t3 (m)	t2 (m)	tf (m)	d (m)
0	Pipe	0,11		0003	
1	Pipe	0,11		0003	
2	Tube	0,17	0,1	0003	
3	Tee	0,09	0,03	0002	
4	Tee	0,09	0,03	0002	
5	Angle	0,1	0,06	0003	
6	Angle	0,1	0,06	0003	
7	Channel	0,06	0,05	0005	
8	Channel	0,06	0,05	0005	
9	Double Angle	0,13	0,08	0003	0015
10	Double Angle	0,13	0,08	0003	0015
11	Double Angle	0,17	0,08	0004	0007
12	Double Angle	0,17	0,08	0004	0007

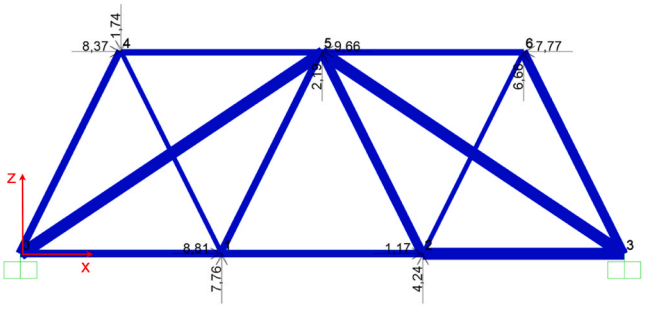

Fig. 7. Random loads assigned to joints (in kN).

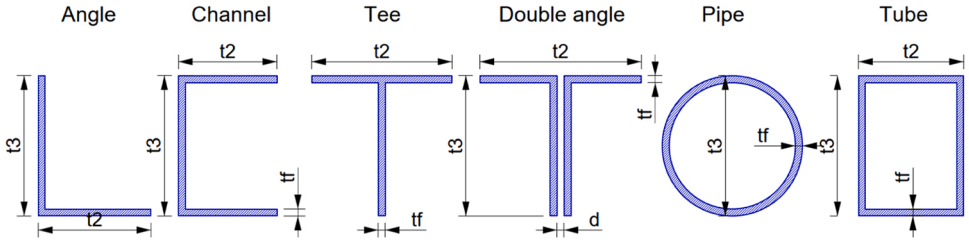

Fig. 6. Random sections for frame elements. The dimensions identified with the black arrows represent respectively: t_3 the section dimension in the z direction of the local reference system, t_2 the section dimension in the y direction of the local reference system, tf is the thickness of the section and d is the distance between different components of the section in the case of double angle ones.

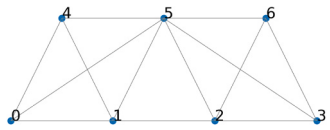

Fig. 8. Graph representation of the model in Fig. 7. The nodal coordinates of the structural system are used to localize the graph nodes.

Table 2
Hyperparameters for GNNs training.

Hyperparameters	Specification
Epochs	300
Batch size	32
Optimizer	Adam
Starting Learning rate	0.001
Loss Function	Mean Absolute Error

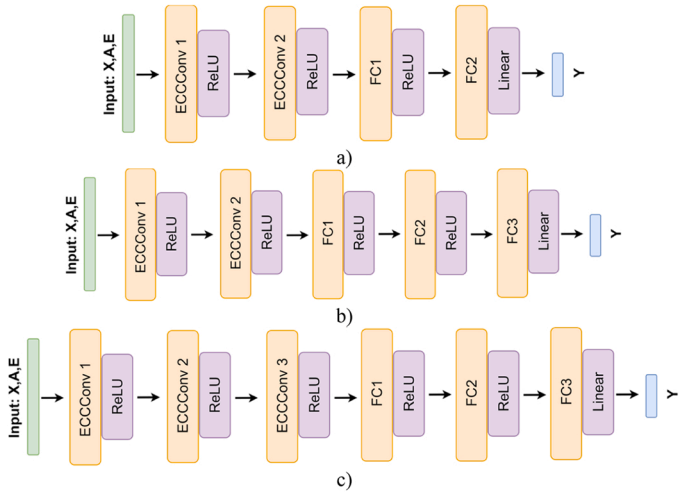

Fig. 9. GNN architectures investigated: a) typology 1, b) typology 2, c) typology 3.

Table 3
Specifications of the models trained: typology of architecture from Fig. 9, layers dimension [29], trainable parameters, training time on a NVIDIA V100 Tensor Core GPU.

Model	Architecture		Parameters
	Typology	Layers dimension	
GNN1	1	ECCConv1: 32, ECCConv2: 32, FC: 16	5379
GNN2	1	ECCConv1: 64, ECCConv2: 64, FC: 32	19971
GNN3	1	ECCConv1: 128, ECCConv2: 128, FC: 64	76803
GNN4	2	ECCConv1: 128, ECCConv2: 128, FC1: 64, FC2: 32	78787
GNN5	3	ECCConv1: 128, ECCConv2, ECCConv3: 128, FC1: 64, FC2: 32,	144451
GNN6	1	ECCConv1: 512, ECCConv2: 256, FC: 128	568579

to the data-driven nature of this approach since it drives and identifies the best strategy and the characteristics of the data to generate, i.e., the numerical modelling approach. Step 2 focuses on numerical modelling, in which the most suitable approach to represent the structural system (e.g., FEM model) and to obtain the numerical solution to the problem stated is defined. This step allows authors to obtain the Reference Model (RM), which represents the ground truth of the structural analysis. Step 3 regards the data synthesis, and it is structured in two main phases: (a) data generation, enabled by automated structural analyses in the numerical model environment to generate the data for training and testing the MISM; (b) data collection as a graph dataset, consisting of, for each structural analysis in the numerical modelling environment, the representation of data in the form of a graph according to the requirements of the MISM. Finally, Step 4 focuses on the MISM training/test/validation, in which the model is trained and tested on the base of the data generated/collected and the problem set.

2.1. Step 1: problem definition

In this phase, the scenario to build the MISM is defined. Considering the field of application (i.e., structural engineering), the MISM could assume different roles, ranging from the replacement of the numerical models for the purpose of existing structures assessment, to the support of the optimization task during the design phase of new structures. Since the MISM will be trained on data generated in the numerical modelling environment, the importance of numerical modelling (e.g., FEM, which represents the most popular approach to simulate the behaviour of real systems and solve engineering problems in controlled software environments) can be easily assessed. In structural engineering, numerical modelling aims to implement physical laws governing a physical phenomenon, allowing generalized solutions over potentially infinite geometries and conditions. As such, this generalization capability comes with several limitations, especially when facing complex structures: high computational effort, time-consuming repeated activities, need for expert analysts, lifelong dependence on economically expensive software, difficulties in the customization of out-of-the-box procedures (e.g., optimization task, damage detection and damage severity assessment). Moreover, numerical models can reach different levels of representativeness about to their purpose: more specific problems require more detailed and complex simulations. These features can drive users to define the SMs and, in the case at hand, the proposed MISM by defining a trade-off between generalization capability and computational complexity [41].

The problem definition phase in the case of a GNN-based MIMS implies the identification of the MISM requirements in terms of generalization capability, and the selection of the most recommended numerical modelling strategy for data generation. Moreover, since the GNN is used as the technique to specifically replace the RM in a specific task, this phase also determines the graph representation of the problem. This graph representation step requires the definition of the elements as nodes or edges, the noticeable quantities characterizing both nodes and edges, the type of relationship between system input and output prediction, and then the task to perform, such as regression or classification. In summary, the problem definition phase identifies the following aspects:

1. Role of the MISM: the scope of the proposed model must be initially set by identifying the inputs and the desired outputs of the model.
2. Generalization capability: it describes the wideness and the type of input variability (i.e., distributions) that the model can handle with sufficient accuracy in output computation. Examples of input variability strongly depend on the specific application and can vary among the load condition distribution to frame section typologies and dimensions, the external and internal constraint variations, and the degrees of internal connections between structural elements. This generalization-driven variability can be named generalization space.
3. Numerical modelling environment: it identifies the most suitable numerical modelling strategy, accounting for both complexity and the required level of detail of the simulation (i.e., data to generate).
4. Graph structure of the problem: it defines the structure to attribute to the inputs of the model, e.g., the definition of the elements represented by nodes, the ones represented by edges, and the relations among nodes and edges.

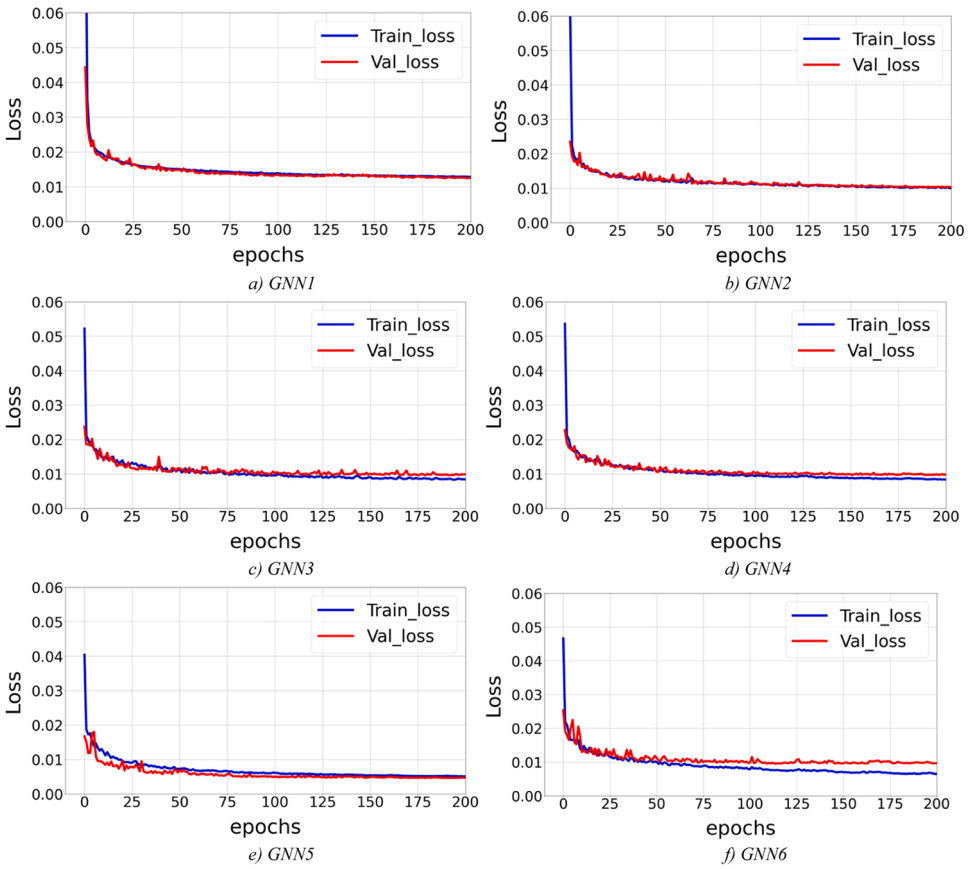

Fig. 10. GNNs training and validation performance: training loss (Train_loss) in blue, validation loss (Val_loss) in red.

Table 4
GNN5 dimensionality increments.

Model	Architecture		Parameters	Training Time
	Typology	Layers dimension		
GNN5–2	3	ECCConv1: 256, ECCConv2: 128, ECCConv3: 128, FC1: 64, FC2: 32	212675	13 min, 53 s
GNN5–3	3	ECCConv1: 256, ECCConv2: 256, ECCConv3: 128, FC1: 64, FC2: 32	409411	19 min, 37 s

2.2. Step 2: numerical modelling

The numerical modelling phase is preliminary to data generation for the training and testing of the MISM. In structural engineering, the solution to different problems may require different levels of detail, considering that the modelling strategies depend on several aspects such as the type and the complexity of the structure, the quantity and quality of data, and the aim of the investigation. As such, the numerical modelling approach needs to be chosen accurately. Among possible strategies, some options can be represented by FEM (e.g., [42]), Boundary Element Method, BEM (e.g., [43]), Discrete Element Method, DEM (e.g., [44]) and other ones provided by the scientific literature. Regardless of the numerical approach, traditional structural analysis methods allow for investigating structures through different analysis typologies, such as linear or nonlinear analyses. In addition, the definition of the structural system is also essential. The system must be modelled in several aspects reported in the problem definition (see Section 2.1) and necessary to solve the numerical model, e.g., geometry, constitutive materials, constraints and loads. The output of the numerical modelling phase is a proper RM, i.e., a numerical model capable of solving the structural problem and providing the desired outputs according to specific input parameters, for example, some numerical simulation results. Once the RM is ready, it can be used to generate data to collect and elaborate, as shown in the next step.

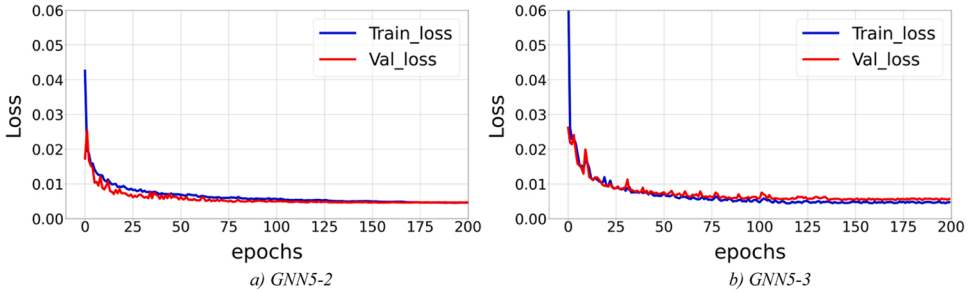

Fig. 11. Training and validation performance of the models obtained by increasing the dimensionality of GNN5.

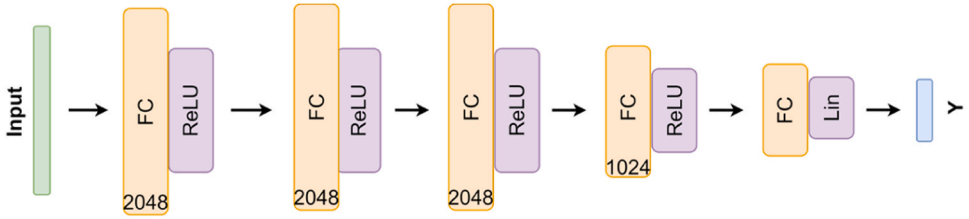

Fig. 12. FFNN architecture to compare with the GNN.

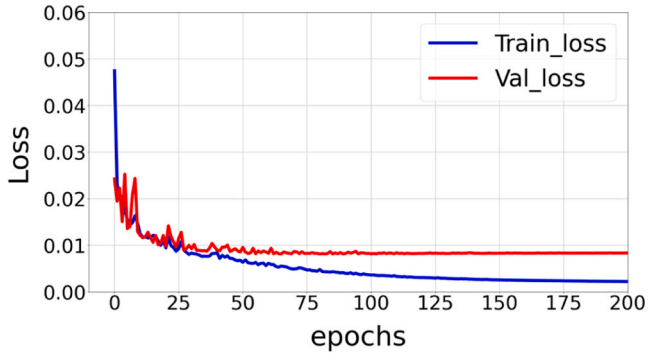

Fig. 13. FFNN training phase; training time 6 min and 24 s.

2.3. Step 3: data generation and collection

The collection of training/test/validation data in the form of a graph is performed in different sub-steps during the data generation and collection phase, and implies:

(i) the definition of the RM for the analysis according to the input parameters sampled from the generalization space of the problem definition.
(ii) the numerical analysis to obtain the output from the RM.
(iii) the representation of all the noticeable quantities and features of the RM and its solution in terms of graphs, and storage of the graph dataset conformed.

These three sub-steps generate a single instance for the MISM; then, the procedure is looped as long as a representative number of instances is generated for the training/test/validation phases, as graphically indicated by the red recursive arrow in Fig. 1.

2.3.1. Numerical model specification and analysis

In sub-step (i), a fully automated procedure was selected to introduce input parameter variables in RM according to the problem definition phase: a custom script drives the numerical modelling environment and features the RM with the input parameters sampled from the generalization space. Depending on the level of detail required according to Step 1, both commercial and custom software environments can be used. Several tools can be employed to vary the parameters in the RM, such as specific codes that exchange information with the structural analysis software (e.g., [45,46]). Alternatively, the generation can be manually processed, by developing and implementing a custom solver environment for the considered problem. Once the RM is featured with all the input parameters for the problem solution, the analysis can be run, and the solutions to the problem can be obtained (sub-step (ii)).

2.3.2. Graph representation and dataset

The sub-step (iii) deals with representation of the problem in terms of graph built both on the input and output parameters of the RM (i.e., the results of the analysis) defined in the previous sub-step.

In general, a graph is a mathematical abstraction of a data structure, which properly represents the entities involved in a problem and the relations among these entities [40]. Hence, the indirect graph is described by a couple $G = (\boldsymbol{V}, \boldsymbol{C})$ where $\boldsymbol{V}$ and $\boldsymbol{C}$ represent a finite set of nodes and a finite set of edges, respectively. The set of nodes $\boldsymbol{V}$ is defined as follows in Eq. 1:

$$\boldsymbol{V} = \{n_0, n_1, \ldots, n_i, \ldots, n_{N_v-1}\} \tag{1}$$

where N_v is the number of nodes of the graph. The set of edges is instead defined as in Eq. 2:

$$\boldsymbol{C} = \{ed_{ij} | n_i, n_j \in \boldsymbol{V}\} \tag{2}$$

where $ed_{ij} \in \boldsymbol{C}$ if there exists, an edge connecting the nodes n_i and n_j. Thus, we denote the set of neighbours of each node n_i as follows:

$$\boldsymbol{N}_i = \{n_j \in \boldsymbol{V} | ed_{ij} \in \boldsymbol{C}\}. \tag{3}$$

In the GNN literature, each node or edge of the graph is characterized by several features representing noticeable quantities. Regarding nodes, a vector of attributes $\boldsymbol{x}_i$ of each node $n_i \in \boldsymbol{V}$ is defined, and the matrix $\boldsymbol{X}$ collecting the features from all the nodes is defined as follows:

$$\boldsymbol{x_i} = [x_{i,0}, x_{i,1}, \ldots, x_{i,N_{xf}-1}]^{\boldsymbol{T}} \tag{4}$$

$$\boldsymbol{X} = [\boldsymbol{x_0}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_i}, \ldots, \boldsymbol{x}_{N_v-1}] \tag{5}$$

where N_{xf} is the number of features of each node. Regarding edges, the vector of attributes $\boldsymbol{e}_{ij}$ of each edge $ed_{ij} \in \boldsymbol{C}$ is defined as follows:

$$\boldsymbol{e_{ij}} = [e_{ij,0}, e_{ij,1}, \ldots, e_{ij,N_{ef}-1}]^{\boldsymbol{T}} \tag{6}$$

with N_{ef} representing the number of features of each edge. Moreover, the following matrix $\boldsymbol{E}$ collects the vectors of edge features:

$$\boldsymbol{E} = [\boldsymbol{e_{ij}}] \tag{7}$$

To represent the topology of the graph, describing the connection among nodes, the adjacency matrix $\boldsymbol{A}$ is defined where $a_{ij} = 1$if $ed_{ij} \in \boldsymbol{C}$, otherwise $a_{ij} = 0$. It is worth noting that the problem to solve is the one that fully characterizes the graph configuration, which is automatically derived from the RM: n_i and ed_{ij} are described in terms of features, i.e., $\boldsymbol{x}_i$ and $\boldsymbol{e}_{ij}$, which in turn are configured by considering modelling input parameters, such as coordinates, structural materials, elastic moduli, constraints, restraints and loads. Fig. 2 shows an example of the representation of a simple supported truss structure by a defined graph. As shown, given the input parameters for nodes and edges (each node is characterized by global coordinates, loads, constraints, and restraints; each edge is characterized by section area of truss, and Young Modulus), the MISM is set to compute the output parameters, e.g., nodal displacements.

In the case of truss structures, the match between structural components and the graph's components is quite straightforward and in line with the literature employing GNN in physical systems: the structural nodes are represented as the graph nodes n_i while trusses and frame elements are represented by the edges ed_{ij}. Obviously, there are also possible cases in which different modelling strategies could be required so that frames can be represented as nodes. The automated analysis

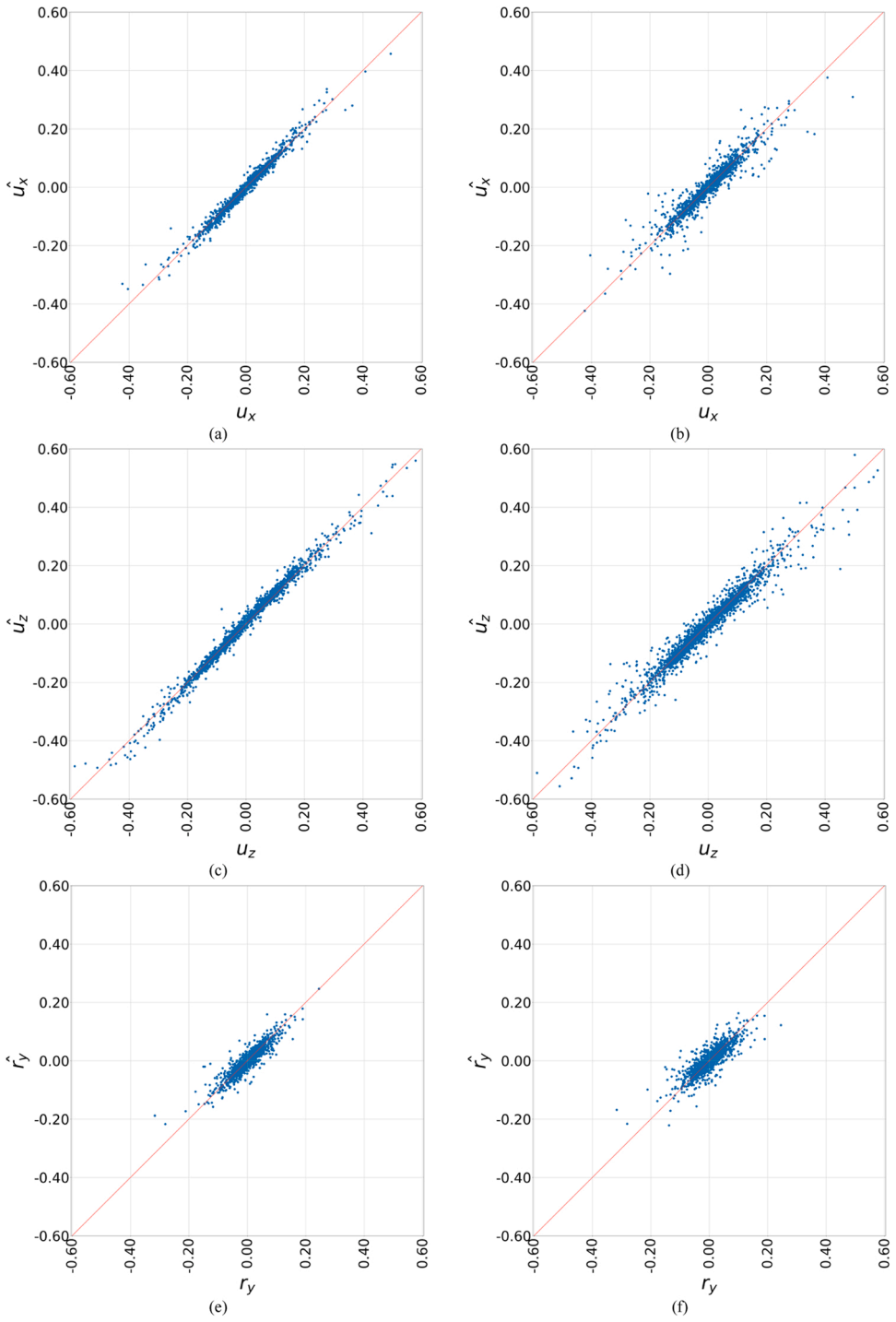

Fig. 14. Regression performance on test set: a), c) and e) indicate u_x, u_z in mm and r_x in rad/1000 predicted by GNN5–3, b), d) and f) indicate u_x, u_z in mm and r_x in rad/1000 predicted by FFNN.

Table 5
Comparison between GNN-5 and FFNN.

metric	GNN5-3			FFNN		
	u_x (m)	u_z (m)	r_y (rad)	u_x (m)	u_z (m)	r_y (rad)
R^2	0.98	0.99	0.83	0.88	0.93	0.69
MAE	4.91e-06	7.89e-06	6.78e-06	8.24e-06	1.24e-05	8.89e-06
SAE	1.72e-02	2.76e-02	2.37e-02	2.88 e-02	4.34e-02	3.14e-02

process allows running the simulations as long as the number of instances required for the MISM training/test is reached. Hence, the graph representation of each analysis is stored so that the graph dataset is generated. In the end, it is worth pointing out the reason for which GNNs were selected as the optimal solution to develop the MISM. First, the principles at the base of GNN and the related graphical structure allow users to simplify the definition of the parameters characterizing the structural system. In fact, as for a generic truss structure (or for a moment-frame structure), nodes and edges constituting GNNs can be directly related to nodes and frames of the physical system, which makes the proposed approach simple and logical. In addition, the mechanics behind the investigated structural system is accounted for through the

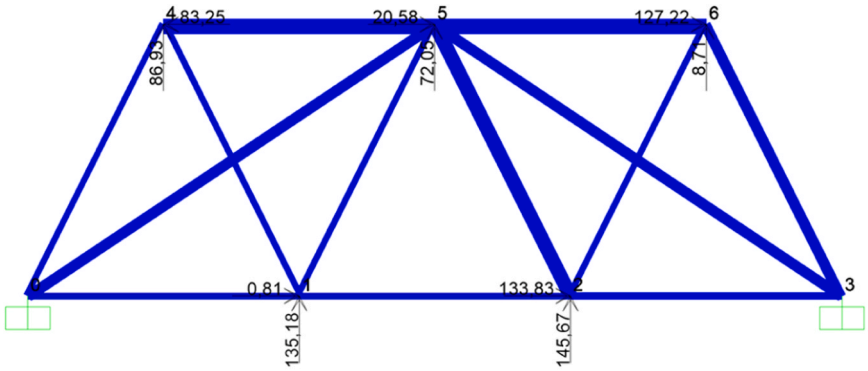

Fig. 15. Test instance representation in the FE software.

Table 6
Sections assigned at random to frames (from Fig. 5) with relative dimensions from Fig. 6.

Frame	Shape	t3 (m)	t2 (m)	tf (m)	d (m)
0	Pipe	0,07			0003
1	Pipe	0,06			0003
2	Tube	0,08	0,04	0002	0002
3	Tee	0,19	0,07	0003	0003
4	Tee	0,17	0,07	0003	0003
5	Angle	0,06	0,08	0004	0004
6	Angle	0,05	0,08	0004	0004
7	Angle	0,06	0,08	0004	0004
8	Angle	0,08	0,08	0004	0004
9	Channel	0,15	0,07	0003	0003
10	Double Angle	0,11	0,09	0003	0003
11	Double Angle	0,12	0,04	0005	0005
12	Double Angle	0,10	0,04	0005	0005

adjacency matrix $\boldsymbol{A}$, which provides the physical informative content of the structural system under investigation. This concept can be easily extended to other structural engineering problems, for which the adjacency matrix $\boldsymbol{A}$ can be set according to the desired elaboration, driving the training of the MISM (as shown in the next Section).

2.4. Definition of the MISM

The graph dataset conformed in the data generation and collection phase is used to train, test, and validate the proposed GNN-based MISM.

2.4.1. Graph neural networks generalities

GNNs are models enabling DL on graph-structured data and are defined as an "*optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances)*" [40]. Their main peculiarity is the capability of updating information stored in each component (nodes, edges, global context) by using information from other components on the base of the defined topology. This updating process creates updated representations of the components on the base of their features, and these representations are intended like embeddings, as provided in the GNN related literature.

Fig. 4 shows the general updating process for the GNN in Fig. 3, in which a random GNN with N_v nodes is considered. The GNN is obtained by stacking n hidden layers, where l_L with $L = 1, \ldots, n$ identifies each layer. According to Section 2.3.2, we denote by $\boldsymbol{x}_i^{(L)}$ the feature representation of each node $n_i \in \boldsymbol{V}$ in the hidden layer L, obtained with one of the available updating procedure. For instance, Fig. 4 shows the first four nodes and the last one of the graphs, and the description of the process is only conducted for these nodes (but it is applied to all the nodes). In the first hidden layer l_1, the embedding $\boldsymbol{x}_0^{(1)}$ of the node 0 (red point), is trained considering its neighbours input embeddings (blue points), that is, $\boldsymbol{x}_1^{(0)}, \boldsymbol{x}_2^{(0)}, \boldsymbol{x}_3^{(0)}$. The same procedure is applied for all nodes until the last one, $\boldsymbol{x}_{N_v}^{(1)}$. All the embeddings are updated until the last hidden layers (l_n), so that $\boldsymbol{x}_0^{(n)}$ (dark red point) represents the last embedding of node 0 performed considering its neighbours input embeddings (dark blue points), that is, $\boldsymbol{x}_1^{(n-1)}, \boldsymbol{x}_2^{(n-1)}, \boldsymbol{x}_3^{(n-1)}$. The procedure, as previously explained, is applied for each node, considering the related embeddings.

Note that the different colours are used in the layers to highlight the information diffusion through the network: in l_1 all nodes are updated only considering the original neighbours, while in l_n all embeddings store information processed after different steps. It is straightforward to notice that more steps imply the influence of farer nodes on each embedding. The output of this generic GNN is represented by the new node embeddings after n hidden layers.

The GNNs variability is mainly based on the formulations to obtain these embeddings, with the standard convolution operation [47] generalized on graph-structured data gathering progressively more attention for this purpose. According to [48], two main approaches to convolution on graphs can be identified: (a) spatial convolution, mainly focused on data aggregation from neighbour components; (b) spectral convolution, which uses the graph Fourier transform to process and update features of components, considering fewer local characteristics.

In this work, we propose the application of a GNN allowing the embedding computation considering also the edge features (i.e., mechanical characteristics of the frames) due to their importance in structural systems. This implies that an edge-conditioned GNN on the model was employed, according to Simonovsky & Komodakis [49]. This architecture is obtained by stacking many edge-conditioned convolutional layers and computing the nodes embedding by considering the edge features:

$$\boldsymbol{x}_i^{(L)} = \boldsymbol{x}_i^{(L-1)} \boldsymbol{W}_{root} + \sum_{j \in N_i} \boldsymbol{x}_j^{(L-1)} MLP(\boldsymbol{e}_{ij}) + \boldsymbol{b} \tag{6}$$

where $\boldsymbol{x}_i^{(L)}$ is the embedding of the node i in the hidden layer l_L, $\boldsymbol{x}_i^{(L-1)}$ is the embedding of the same node i in the previous hidden layer, $\boldsymbol{N}_i$ is the set of neighbours for the node i, $\boldsymbol{x}_j^{(L-1)}$ is the embedding of the j neighbours of the node i, $\boldsymbol{W}_{root}$ is a trainable weight matrix multiplying $\boldsymbol{x}_i^{(L-1)}$, $MLP(\boldsymbol{e}_{ij})$ is a trainable Multi-Layer Perceptron [50] that outputs an edge-specific weight as a function of edge attributes $\boldsymbol{e}_{ij}$, and $\boldsymbol{b}$ represents the trainable biases.

2.4.2. Graph neural networks training/test/validation

The dataset generated with the automated analyses can be used to train/test/validate the defined MISM. All the training-related quantities are specified in this phase: optimizer, learning rate schedule, batch size, epochs, loss function, and performance metrics. In particular, the loss function gains particular attention in this phase when dealing with

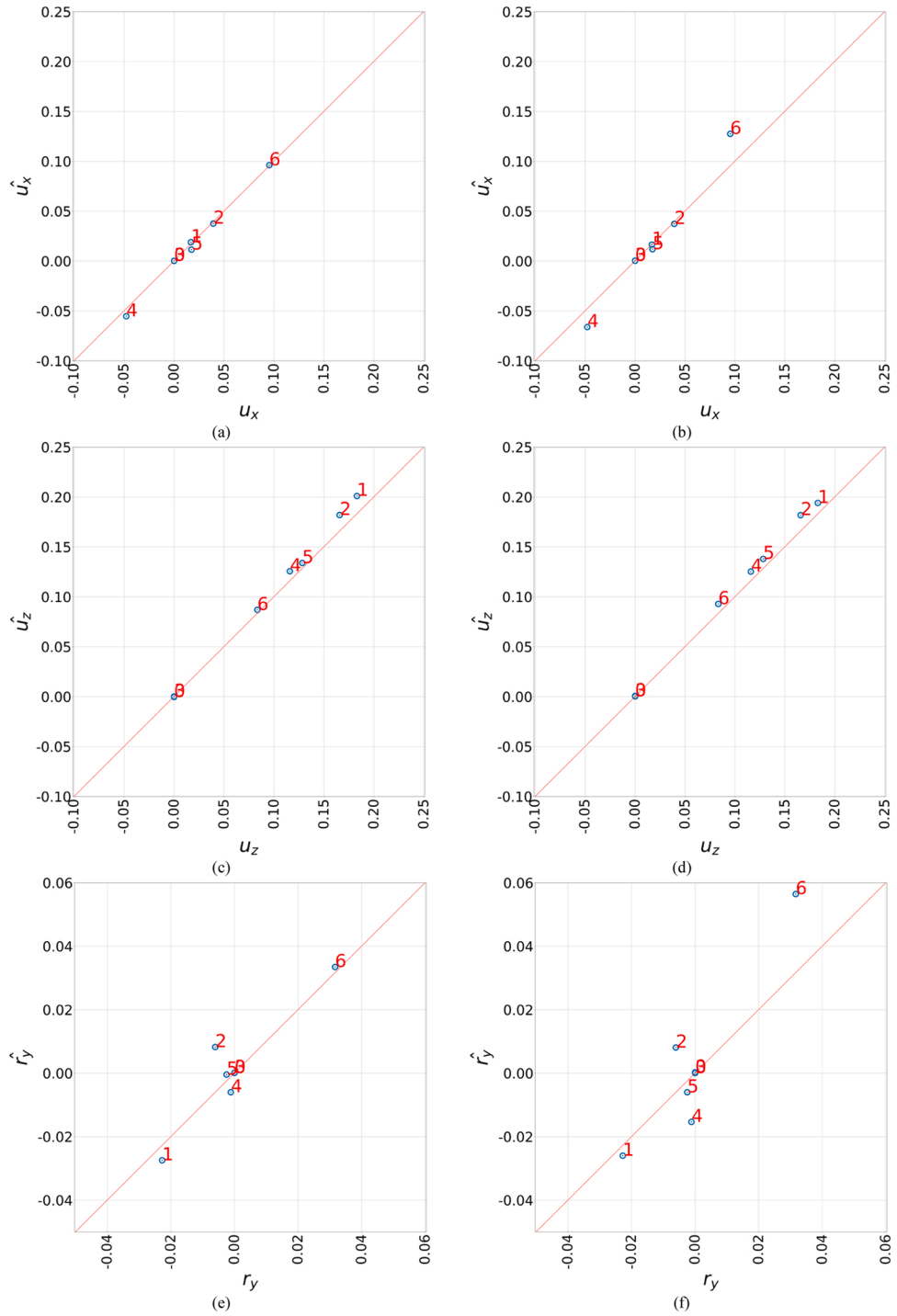

Fig. 16. Regression performance comparison on single test instance: a), c) and e) represent respectively the prediction for u_x, u_z in mm and r_x in rad/1000 in the case of GNN5–3; b), d) and f) represent respectively the prediction for u_x, u_z in mm and r_x in rad/1000 in the case of FFNN.

Table 7
Comparison metrics between GNN3–5 and FFNN for a single test prediction.

metric	GNN5-3			FFNN		
	u_x (m)	u_z (m)	r_y (rad)	u_x (m)	u_z (m)	r_y (rad)
R^2	0.99	0.96	0.84	0.86	0.96	0.35
MAE	2.69e-06	7.61e-06	3.98e-06	8.45e-05	8.06e-05	8.60e-05
SAE	1.88e-05	5.33e-05	2.74e-05	5.92e-04	5.64e-04	6.02e-04

physical systems. According to the literature review presented in the introduction, the right choice can influence massively the performance of the system and the custom loss function can inject further knowledge into the training process [34].

Many standard loss functions are generally available in deep learning applications. In classification tasks, noticeable loss functions are the cross-entropy-based ones, such as Binary Cross-entropy, categorical cross-entropy, sparse-categorical cross-entropy, or Poisson-based ones [54,55]. In regression tasks instead, frequently used loss functions are mean squared error, mean absolute error, mean absolute percentage error, mean squared logarithmic error and cosine similarity [54,55]. In

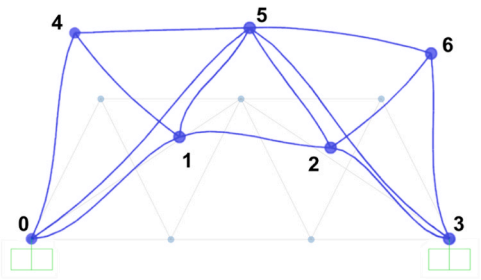

Fig. 17. Deformed shape computed by the RM.

the field of deep learning for physical systems, in which regression tasks are more widespread, two important aspects drive the choice of the loss function: (i) the possibility to encode in the loss function the knowledge of the system [34–36]; (ii) the interpretability of the resulting values [37,38]. When different typologies of independent variables are involved in the solution of the problem, the strategy (i) can be pursued. An example of this approach is reported in [36], where the authors specified the general PINN framework (i.e., the solution of partial differential equations via deep learning) for soil mechanics modelling. Since multiple displacements and stresses are predicted together using parallel Neural Network architectures, the authors encoded a loss function considering the relations among variables. Instead, in the case of simpler scenarios, such as the prediction of a single variable (e.g., displacements), strategy (ii) is considerable. As in [37,38], when dealing with displacement predictions, loss functions with easily interpretable results (i.e., directly referred to the displacement values) can be used, e.g., mean absolute error or sum of absolute errors.

3. Case study 1: 2D elastic truss

The proposed methodology was first tested on a 2D elastic steel truss (shown in Fig. 5), modelled through SAP2000 [51] and which assumed the role of RM. The main aim of this section is to assess if the proposed approach can provide, after training the GNN, outcomes in line with the RM. All the noticeable nodal displacements were considered among the outcomes to account for. This strategy allowed the authors to configure a MISM that relates a simple structure defined by generic features (in terms of geometry, structural materials, constraints, restraints, loads) to the desired output (i.e., vertical, and horizontal displacements, and rotation). In addition, a further aim of the MISM should be to provide a comparable result to other ML-based models, with the improvement given by the mechanical base.

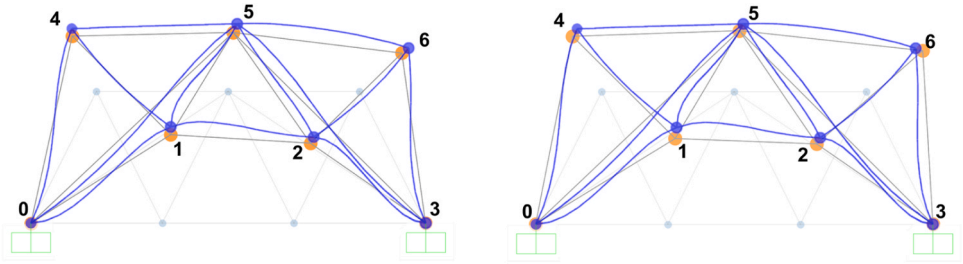

Fig. 18. Comparison between the predicted displacements without considering the deformations of the frames: a) GNN3–5 and b) FFNN.

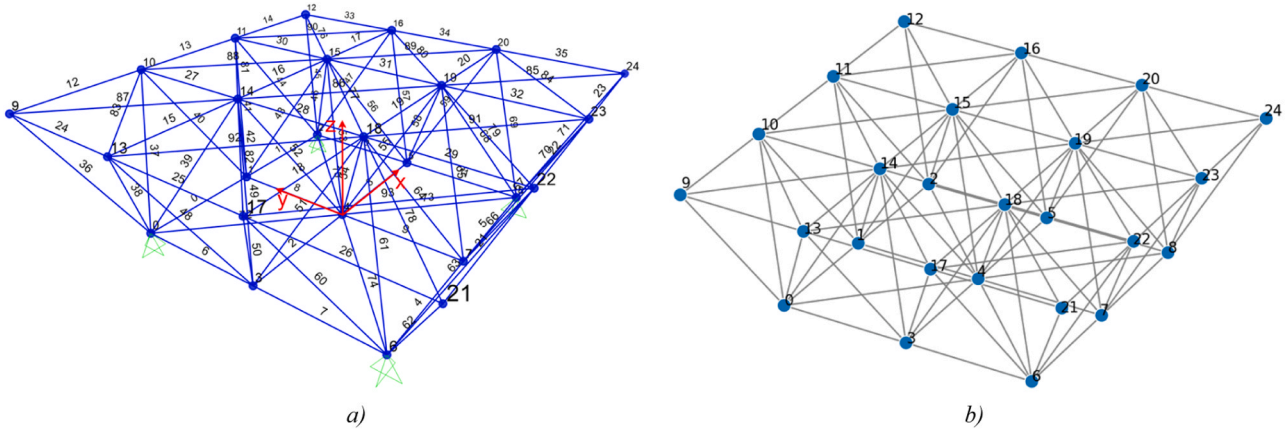

Fig. 19. Spatial truss case study: a) visualization of the RM, b) graph representation considering the nodal coordinates for the visualization.

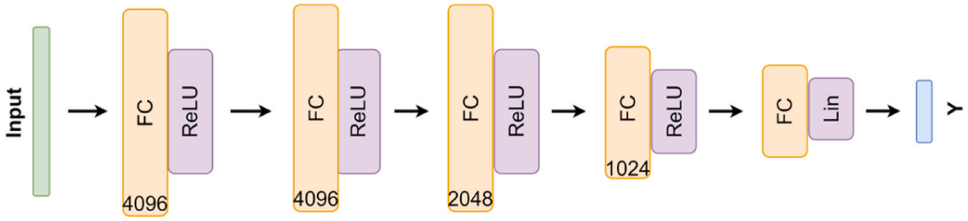

Fig. 20. FFNN architecture used for the spatial structure case study.

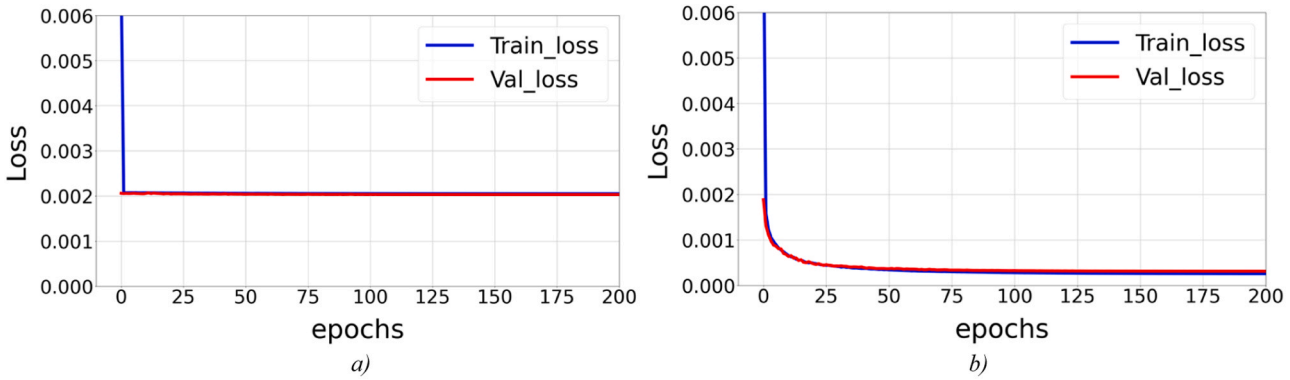

Fig. 21. FFNN training phase for different architectures in spatial structure case study: a) FFNN architecture from Fig. 12, b) Modified FFNN in Fig. 20.

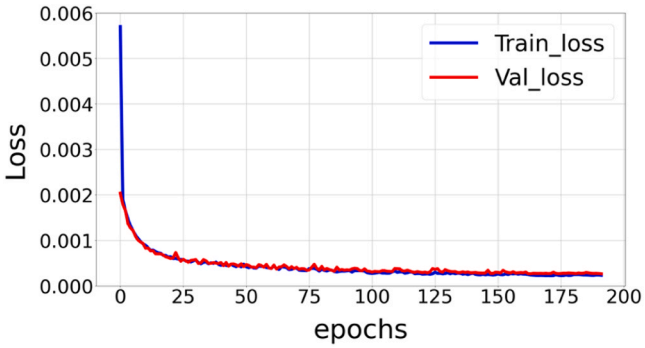

Fig. 22. GNN5–3 training on spatial structure data.

3.1. Application of the methodology

In this section, the methodology introduced in Section 2 was applied to the case of the 2D steel truss shown in Fig. 5. In the following paragraphs, all the phases are specified, as well as the tools employed. The resulting performances are presented at the end of the procedure and compared with a standard architecture eligible for the same task, such as MLP [52].

3.1.1. Problem definition

The main role of the MISM was identified with the capability to predict linear nodal displacements of the 2D truss structure on the base of nodal loads and frame sections. SAP2000 environment [51] was selected as the numerical modelling environment to build the RM by specifying nodal coordinates, nodal loads (forces), and cross-sections. The RM is an internally and externally hyperstatic steel truss, with two fixed restraints in the external nodes (0 and 3 in Fig. 5). Frame-type elements were used to model the trusses. Elastic linear behaviour was considered using a steel material with an elastic modulus of 210 GPa and a density equal to 78.5 kN/m^3.

The generalization space, driving the variability of the input parameters, was defined by the nodal loads and the geometries of the frame sections. This RM variability was first accounted for by varying the frame cross sections among six possible typologies to sample from. Fig. 6 shows the selected set of sections and the related parameters affecting their size (black arrows) and the inertia (i.e., stiffness).

Moreover, as already specified, the generalization space implied that also nodal forces were varied. The nodal displacements in x and z directions (u_x, u_z) and the rotation around the y axis (r_y) were considered as the output of the analysis, according to the reference system shown in Fig. 5.

3.1.2. Data generation and collection

The data generation and collection phase was performed through a script iteratively performing these subsequent steps: (a) definition of the input parameters (i.e., loads and frame sections according to the generalization space) for the RM model to obtain a numerical solution (i.e., no labilities can occur); (b) execution of a linear static analysis according to the defined loads; (c) representation of the model and the results in a graph form; (d) storing of the graph to populate the dataset for the training/test/validation. According to this framework, 15000 looping analyses were performed, and the related dataset instances were obtained.

The model was specified by using SAP2000 OpenAPI through Python programming language [53]: the RM was modified through a custom script by specifying the frame sections and the nodal loads. Each frame was randomly assigned with typologies and dimensions as shown in Table 1 for the example in Fig. 7, while loads in x and z directions were randomly assigned to each node, as depicted in Fig. 7.

After the model specification, a linear static analysis was automatically run, and the nodal displacements for each node were obtained. All the information characterizing the specific numerical problem (i.e., frame sections and loads assignments) and the resulting solutions were represented in the form of graph, according to the specifications in Section 2.3.2. In particular, the following elements were specified:

- The node feature matrix $\boldsymbol{X} = [\boldsymbol{x}_0 \ldots \boldsymbol{x}_i \ldots \boldsymbol{x}_6]$, where $\boldsymbol{x}_i = [x_{g,i}, z_{g,i}, c_{i,x}, c_{i,z}, cr_{i,y}, l_{i,x}, l_{i,z}]^T$ and $x_{g,i}$ and $z_{g,i}$ are the x and z global coordinates of the node $n_i \in \boldsymbol{V}$, $c_{i,x}, c_{i,z}, cr_{i,y}$ are the external constraints (equal to 1 if the node is constrained and equal to 0 otherwise), and $l_{i,x}, l_{i,z}$ are the external loads applied in x and z directions, respectively. Fig. 8 shows a representation of the graph in which the nodes are localized using the nodal coordinates of the structural system.
- The edge features matrix $\boldsymbol{E} = [\boldsymbol{e}_{0,1} \quad \boldsymbol{e}_{0,4} \quad \boldsymbol{e}_{0,5} \ldots \boldsymbol{e}_{6,3}]$ where $\boldsymbol{e}_{ij} = \{a_{ij}, J_{ij}\}$, a_{ij} is the cross-sectional area of the frames and J_{ij} is the inertia moment of the section.
- The adjacency matrix $\boldsymbol{A}$ reporting the connection among nodes via edges.

In this application, the desired output of the model was expressed by the matrix $\boldsymbol{Y} = [y_0 \ldots, y_i \ldots y_6]$, with $y_i = [u_{i,x}, u_{i,z}, r_{i,y}]^T$, where $u_{i,x}$ is the displacement of the node n_i in the x direction, $u_{i,z}$ in the z direction and $r_{i,y}$ the rotation around the y axis, all measured in the global reference system.

3.1.3. GNN training

Once conformed the graph dataset, training, test, and validation of the GNN were performed by using a Python library called Spektral [54], which builds on top of Keras [55]. The GNN was trained to predict the

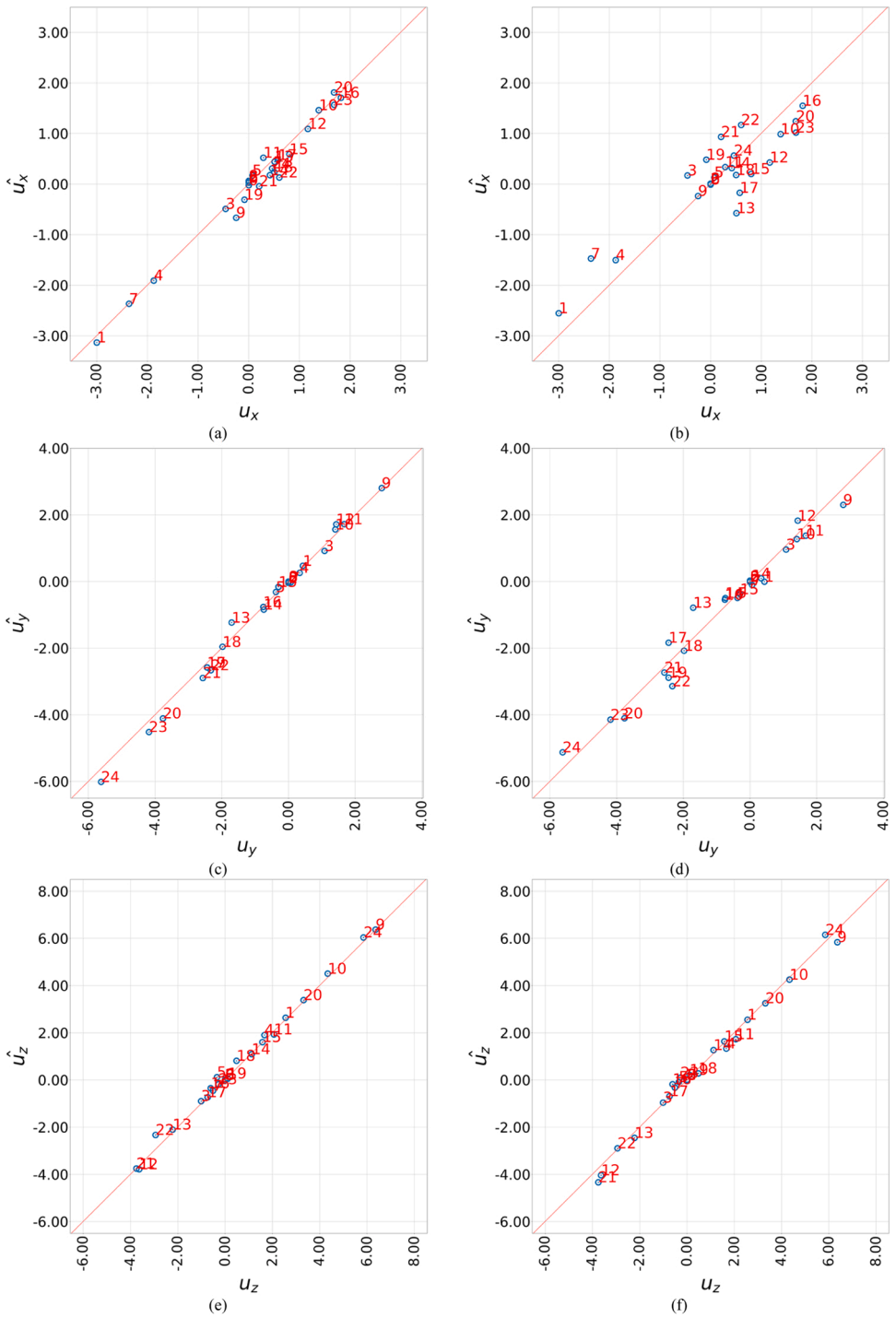

Fig. 23. Regression performance comparison on single spatial test instance: a), c) and e) represent respectively the prediction in mm for u_x, u_y and u_z in the case of GNN5–3; b), d) and f) represent respectively the prediction in mm for u_x, u_y and u_z in the case of FFNN.

nodal displacements $\boldsymbol{Y}$ on the base of graph input data expressed as $\boldsymbol{X}$, $\boldsymbol{E}$, $\boldsymbol{A}$ as highlighted in Section 3.1.2. The models were trained using an NVIDIA V100 Tensor Core GPU hardware with the hyperparameters in Table 2.

Small variations of a baseline model obtained by stacking Dynamic Edge-Conditioned Convolutional layers (ECCConv) and Fully Connected layers (FC) were investigated, as reported in Eq. 6 [49]. Fig. 9 shows the different global architectures investigated, while Table 3 specifies each layer features: the typology of each model referred to Fig. 9, the layers dimension both for ECCConv and FC layers, and the number of trainable parameters of the model. The final FC layer of each model implements a linear activation function to enable the regression task on the output of the network, while the rectified linear unit (ReLU) activation function was used for each layer. As loss function for the neural network, the Mean Absolute Error (MAE) was selected, due to its standard use in DL applications and its representativeness for the problem studied: the MAE value directly represents the difference among the predictions and the ground truth values (obtained by the numerical analysis on RM) in terms of nodal displacements. Basically, the gross performance variation related to the deepness of the network and the layers dimension was investigated, to define an eligible baseline to assess the presented methodology.

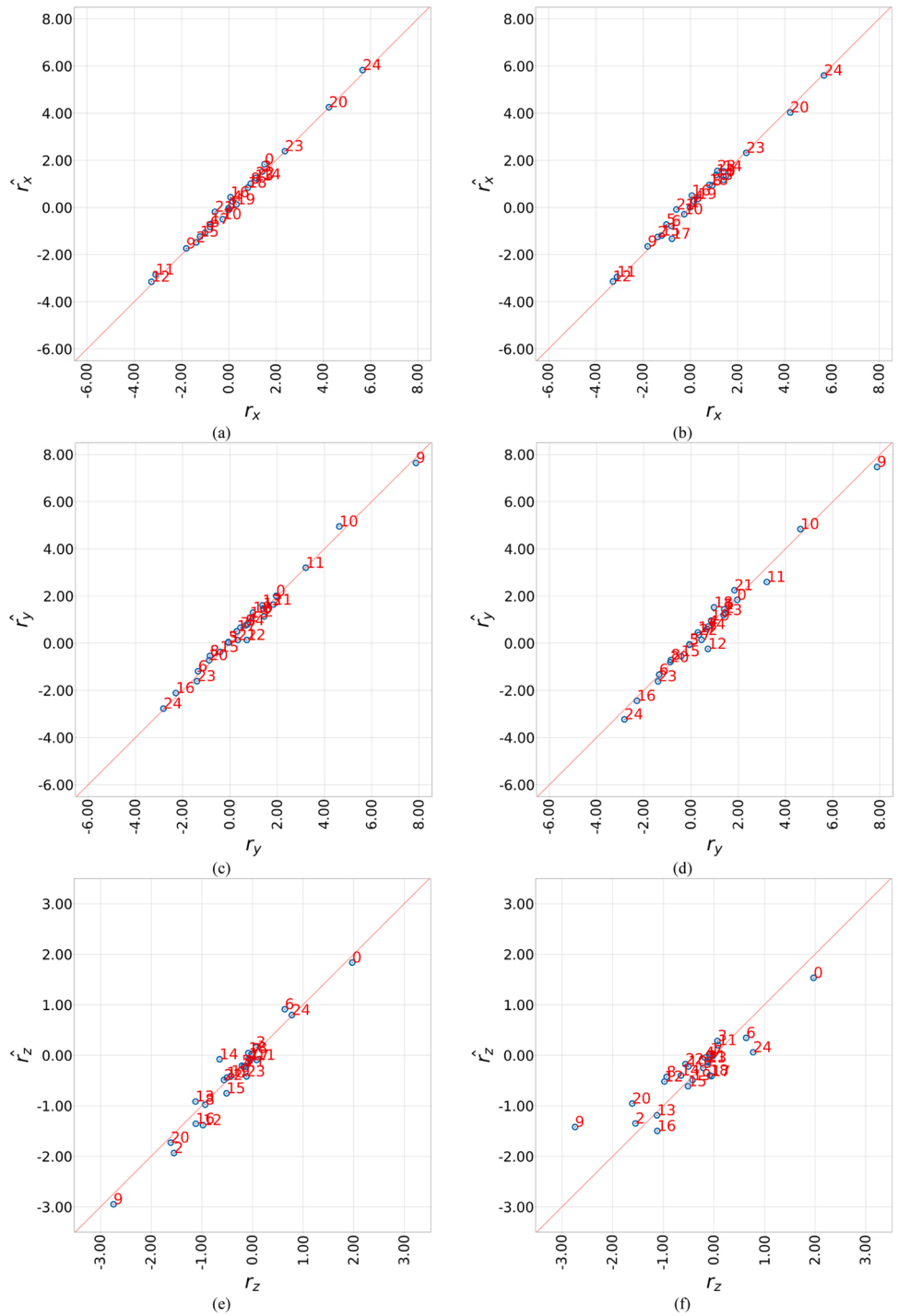

Fig. 24. Regression performance comparison on single spatial test instance: a), c) and e) represent respectively the prediction in radians/1000 for r_x, r_y and r_z in the case of GNN5–3; b), d) and f) represent respectively the prediction in radians/1000 for r_x, r_y and r_z in the case of FFNN.

Fig. 10 shows the performance of all the GNNs trained according to the architectures described in Table 3, and drives the selection of the most promising one. The results highlighted that GNN5 is the architecture showing the best performance (i.e., lower validation loss), contextually with a restrained number of trainable parameters (e.g., differently from GNN6 in Table 3 that results quite big), and therefore it was selected for further investigation.

This evidence allowed us to further explore the dimensionality of GNN5 and improve the performance shown in Fig. 10(e). The model dimensionality was gradually incremented obtaining GNN5–2 and GNN5–3, whose architectures are specified in Table 4. GNN5–3 resulted the best eligible model, as shown in Fig. 11(b) and was selected to investigate the performance of the GNN model in comparison with standard the architecture in Section 3.2.

3.2. Comparison with standard SM

To assess the efficiency of the proposed MISM, the performance of GNN5–3 was compared with a Feed Forward Neural Network (FFNN), a standard architecture widely used in the literature for surrogate modelling in structural engineering. The choice of such an architecture is driven by its acknowledged trade-off between complexity and

Table 8
Comparison metrics between GNN3–5 and FFNN for a single spatial test prediction.

Model displacements		metrics			
		R^2	MAE	SAE	max
GNN5–3	$\mathbf{u_x}$	0.97	1.44e-04	3.61e-03	2.99e-03
	$\mathbf{u_y}$	0.99	1.51e-04	3.77e-03	5.61e-03
	$\mathbf{u_z}$	0.99	1.28e-04	3.20e-03	6.36e-03
	$\mathbf{r_x}$	0.99	1.31e-04	3.28e-03	5.66e-03
	ry	0.99	1.88e-04	4.69e-03	7.87e-03
	$\mathbf{r_z}$	0.94	1.57e-04	3.92e-03	2.75e-03
FFNN	$\mathbf{u_x}$	0.80	3.93e-04	9.82e-03	2.99e-03
	$\mathbf{u_y}$	0.94	2.91e-04	6.92e-03	5.61e-03
	$\mathbf{u_z}$	0.96	1.96e-04	4.81e-03	6.36e-03
	$\mathbf{r_x}$	0.95	1.74e-04	4.36e-03	5.66e-03
	ry	0.94	2.18e-04	5.46e-03	7.87e-03
	$\mathbf{r_z}$	0.77	2.98e-04	7.45e-03	2.75e-03

performance and its extensively use in problems in which there is no temporal dependency of variables [56–59]. This comparison aimed to assess the effectiveness of the performance provided by the further level of information exploited by the MISM (i.e., the topology of the structural system).

3.2.1. FFNN training

To set the regression problem with a standard FFNN, $\boldsymbol{X}$ and $\boldsymbol{E}$ were horizontally stacked to configure the input, as well as Y, to make both input and output compliant with the data structure this architecture. With this approach, every instance of the dataset was no longer represented in the form of a graph, but as a single row of the input matrix characterized by $f = N_v{\cdot}N_{x,f} + N_e{\cdot}N_{e,f} = 61$ features, where $N_v = 7, N_{x,f} = 5$, $N_e = 13$ and $N_{e,f} = 2$. Analogously, the output/target for each instance of the input matrix was a vector of dimension $n_y = N_v{\cdot}n_{displ} = 21$, where $N_v = 7$ is the number of nodes and $n_{displ} = 3$ is the number of displacements predicted for each node.

In line with the existing literature, the entire dataset was split in train/validation/test by using a standard 70/30 split [60]. The generated input matrices, $\boldsymbol{X}_{tr}$, $\boldsymbol{X}_{va}$, $\boldsymbol{X}_{te}$, were characterized by dimension $\dim(\boldsymbol{X}_{tr}) = [l_{tr}, f]$, $\dim(\boldsymbol{X}_{va}) = [l_{va}, f]$, $\dim(\boldsymbol{X}_{te}) = [l_{te}, f]$, respectively, where $l_{tr} = 10150$, $l_{va} = 4350$, $l_{te} = 500$, while $f = 61$ is the global number of features**.** The output/target matrices $\boldsymbol{Y}_{tr}$, $\boldsymbol{Y}_{va}$, $\boldsymbol{Y}_{te}$ were obtained with the same approach, with dimensions $\dim(\boldsymbol{Y}_{tr}) = [l_{tr}, n_y]$, $\dim(\boldsymbol{Y}_{va}) = [l_{va}, n_y]$, $\dim(\boldsymbol{Y}_{te}) = [l_{te}, n_y]$, respectively.

Fig. 12 shows the architecture used for the FFNN where, differently from Fig. 9, standard FC layers were considered, with a total number of trainable parameters for this architecture equal to 10.639.381.

Fig. 13 shows training phase of the FFNN. It is evident that the network overfitted on the data after few epochs since the training loss was improving, while the validation loss remained quite constant around a value of the loss equal to 0.012 [61]. The validation loss value was also a symptom of the lower accuracy in comparison with the trained GNN5–3. This aspect is further specified in Section 3.2.2.

3.2.2. Performance comparison

In this section, the GNN performance was assessed in comparison with the FFNN. First, both models were compared considering an average behaviour over a determined number of test instances (i.e., 500). After, the same evaluation was performed over a single instance.

Considering a test set of $l_{te} = 500$ instances, Fig. 14 shows the comparison among the predicted values (signed with an hat " " on the y axis and) and the ground truth values (on the x axis) of the displacements. The red line represents the ideal condition in which the displacements from the models perfectly match the test set. The more the blue points distribute close to the red line, the more the model can interpret the test data. In particular, the graphs on the left (a), (c), (e) depict predicting performance of GNN5–3 for u_x, u_z, r_y, respectively, while the graphs on the right (b), (d), (f) refer to FFNN. From the graphical comparison, it is evident that GNN5–3 provided better performance for all the displacement components. Instead, the prediction by FFNN was less squeezed on the red line, which implied more scattered points and thus, a less precise predictions.

In line with the existing literature, the performance highlighted in Fig. 14 are expressed in terms of multiple summary metrics in Table 5, where the coefficient of determination, R^2, the MAE, and the sum of the absolute errors (SAE) are reported. GNN5–3 globally overcame FFNN in each metric, with a higher value of R^2, a MAE that is generally equal to half of the same metric given by FFNN, and a SAE that is noticeably lower.

Regarding the prediction capability of the single example instance, randomly chosen input data were set to generate a new RM. Fig. 15 shows the loads condition applied, while Table 6 highlights the sections randomly assigned to the frames.

Applying both GNN5–3 and FFNN models, the new results obtained are shown in Fig. 16 and resumed by multiple metrics in Table 7. Each node was labelled (in red), and its prediction according to both the model was represented. GNN5–3 clearly provided a better prediction than FFNN, showing results closer to the ground truth expressed by the red line. The same evidence is highlighted in Table 7, where all MAE and SAE are noticeable lower for GNN5–3, while R^2 is higher.

The results in Fig. 16 were also represented in Fig. 17, which shows the deformed shape obtained by numerical analysis on RM, and in Fig. 18, where the nodal displacements predicted respectively by GNN3–5 (a) and FFNN (b) were assigned to the nodal coordinates (the images does not consider deformations of the trusses). The deformed shape predicted by GNN-3 was clearly closer the ground truth represented by the RM in Fig. 17, further specifying the results already highlighted in Fig. 16 and Table 7. This outcome clearly showed the improvements achieved by using a MISM (i.e., GNN3–5) instead of a generic ML-based SM (i.e., FFNN).

4. Case study 2: spatial-truss case study

After the application of the procedure to the 2D case, the procedure was tested on a tri-dimensional (3D) structure shown in Fig. 19. The steps are not described in detail here as they are the same illustrated in Section 3. After defining the problem, the RM and the data for the training were generated, by accounting for the variability in terms of cross sections and nodal loads. The training and test phase results of the MISM were herein presented, along with the comparison with the considered ML-based method, the FFNN. Still, the performance of both models was directly highlighted only for purpose of prediction of a single instance.

In this case, a four meters-spanning spatial structure was considered, internally hyperstatic and externally isostatic (simply supported), consisting of 25 joints and 188 frames, randomly loaded by forces on the nodes. The boundary conditions are highlighted in Fig. 19(a), while the graph representation, using the nodal coordinates to locate the nodes of the graph, is shown in Fig. 19(b).

Differently from the 2D case study, the output researched consisted of a matrix considering all the displacement along the six degrees of freedom characterizing each joint of the spatial structure in the 3D space. Hence, the global output was represented by $\boldsymbol{Y} = [\boldsymbol{y}_0 \ldots \boldsymbol{y}_i \ldots \boldsymbol{y}_{24}]$ with $\boldsymbol{y}_i = [u_{i,x} \quad u_{i,y} \quad u_{i,z} \quad r_{i,x} \quad r_{i,y} \quad r_{i,z}]^T$, where $u_{i,x}$ is the displacement of the node i in the x direction, $u_{i,y}$ is the displacement of the node i in the y direction, $u_{i,z}$ is the displacement of the node i in the z direction, $r_{i,x}$ the rotation around the x axis, $r_{i,y}$ the rotation around the y axis, $r_{i,z}$ the rotation around the z axis.

4.1. Models training

Both the GNN and FFNN models were trained on 15000 instances

obtained by the data generation and collection phase. In this case, the FFNN architecture was expanded from the 2D to the 3D case study, and the different architecture is shown in Fig. 20: the former version was unable to train on the data related to the spatial structure because of the dimensionality. This aspect was also highlighted by the training metrics shown in Fig. 21(a). Differently, Fig. 21(b) represents the training metrics of the modified architecture in Fig. 20, characterized by 28.812.438 total trainable parameters: it clearly represents a big network, especially if compared to GNN5–3. The training of the new version of FFNN is reported in Fig. 22.

4.2. Comparison between GNN5-3 and FFNN

According to the comparison between the MISM and the SM model, the prediction performance over a single instance was processed and the results are graphically shown in Fig. 23 and Fig. 24 and resumed by multiple metrics in Table 8. Also in this case, Fig. 23 and Fig. 24 show that GNN5–3 presents higher performance than FFNN in terms of prediction, showing results closer to the ground truth mainly for u_x and r_x, as expressed by the red line. The same evidence is highlighted in Table 8, where all metrics are noticeable lower for GNN5–3 (except for the R^2, which presents higher values), especially in the case of u_x (according to the evidence in Fig. 23(a) and Fig. 23(b)) and r_x (according to Fig. 24(a) and Fig. 24(b).

5. Discussion, conclusions, and further developments

In this work, a new method for configuring mechanics-informed surrogate modelling for structural engineering problems was presented. The proposed solution, named Mechanics-Informed Surrogate Models (MISM), consisted of a data-driven procedure subdivided into four subsequent phases. The first phase of the method dealt with the problem definition, in which the role and the end-usage of the MISM were defined. In the second phase, once the numerical modelling environment was selected, a reference model was generated: it was characterized by specific parameters to vary, as the input variables according to the generalization space identified in the first phase. The third phase dealt with the data generation and collection, which consisted of an automated structural analysis conducted on the Reference Model, iteratively specified by varying input parameters. In the last phase, the Graph Neural Network (GNN) based model (i.e., MISM) was trained and tested on the generated data for its scenario of use. The proposed approach was tested on two scenarios in which a specific MISM was configured to predict nodal displacements in the case of 2D and 3D truss structures. The performance of the MISM was compared with a standard surrogate model, SM, such as the Feed-Forward Neural Network, FFNN. The results highlighted that MISM performed better than the involved SM, in terms of multiple evaluation metrics. Also, the related images graphically visualizing the predicted quantities over the ground truth supported this evidence, showing a less scattered MISM prediction in comparison to the SM.

In general, this work aims to contribute to the research field of Physical-Informed Artificial Intelligence tools, which can empower the application of new technologies in the structural engineering field. GNNs could represent an outbreaking technology for DL applications in structural engineering, specifically due to the strong mechanics-based nature of the investigated problems. Even if DL is already deeply used in many structural engineering tasks, its use still appears to be an attempt to transfer alien practices from the far research field.

Looking at the proposed application, some limitations should be highlighted. First, it is necessary to dispose of dedicated performative hardware: the training phase of these models is high-demanding in terms of computational effort on standard hardware. The problem could be overcome by accessing paid computational resources available (e.g., Google cloud computing services). Still, another con is related to the model training phase, which results slower than standard non-graph-based architectures. In fact, the FFNN was trained in about twenty minutes on the hardware used, while the MISM required about one hour for training. However, the main advantage of the proposed approach is the achievement of better predictive performance. This aspect supports the statement about the improved representativeness of physical and structural systems and allows the authors to accept this trade-off between performance and effort in training. In the end, it is worth specifying that graph-based technologies are still poorly explored in their applications to fields such as structural engineering and then, they feature less maturity than standard DL and ML, which are already strongly employed.

Although the application of the methodology was tested on simple cases, the potentialities of the approaches are evident, and many directions could be explored in the future using physically informed models, such as new optimization tools for designing new structures. Still, different fields of structural engineering could take advantage of the proposed approach, for example, to provide new automated protocols in the production of new structural materials (e.g., [62–64]). Finally, the nonlinear behaviour of existing structures could be predicted by reducing the computational efforts required when dealing with the nonlinearities of complex structures.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The second author acknowledges funding by Italian Ministry of University and Research, within the project ‘PON-Ricerca e Innovazione-2014–2020, CODICE CUP (D.M. 10/08/2021, n. 1062): D95F21002140006; (D.M. 25/06/2021, n. 737): D95F21002160001.

References

[1] Sun H, Burton HV, Huang H. Machine learning applications for building structural design and performance assessment: state-of-the-art review. J Build Eng 2021;33:101816. https://doi.org/10.1016/j.jobe.2020.101816.
[2] Xie Y, Sichani ME, Padgett J, DesRoches R. The promise of implementing machine learning in earthquake engineering: a state-of-the-art review. Earthq Spectr 2020;36(4):1769–801. https://doi.org/10.1177/8755293020919419.
[3] Zhang Z, Zhou H, Ma J, Xiong L, Ren S, Sun M, Wu H, Jiang S. Space deployable bistable composite structures with C-cross section based on machine learning and multi-objective optimization. Comp Struct 2022;297:115983. https://doi.org/10.1016/j.compstruct.2022.115983.
[4] Liu R, Kumar A, Chen Z, Agrawal A, Sundararaghavan V, Choudhary A. A predictive machine learning approach for microstructure optimization and materials design. Sci Rep 2015;5(1):11551. https://doi.org/10.1038/srep11551.
[5] Liu F, Jiang X, Wang X, Wang L. Machine learning-based design and optimization of curved beams for multistable structures and metamaterials. Extr Mech Lett 2020;41:101002. https://doi.org/10.1016/j.eml.2020.101002.
[6] Mangalathu S, Sun H, Nweke CC, Yi Z, Burton HV. Classifying earthquake damage to buildings using machine learning. Earthq Spectr 2020;36(1):183–208. https://doi.org/10.1177/8755293019878137.
[7] Lin YZ, Nie ZH, Ma HW. Structural damage detection with automatic feature-extraction through deep learning. Comp Aided Civ Infrastruct Eng 2017;32(12):1025–46. https://doi.org/10.1111/mice.12313.
[8] Hung DV, Hung HM, Anh PH, Thang NT. Structural damage detection using hybrid deep learning algorithm. J Sci Technol Civ Eng (STCE)-NUCE 2020;14(2):53–64. https://doi.org/10.31814/stce.nuce2020-14(2)-05.
[9] Dang HV, Raza M, Nguyen TV, Bui-Tien T, Nguyen HX. Deep learning-based detection of structural damage using time-series data. Struct Infrastruct Eng 2021;17(11):1474–93. https://doi.org/10.1080/15732479.2020.1815225.
[10] Pathirage CSN, Li J, Li L, Hao H, Liu W, Wang R. Development and application of a deep learning–based sparse autoencoder framework for structural damage identification. Struct Health Monit 2019;18(1):103–22. https://doi.org/10.1177/1475921718800363.
[11] Dang HV, Tran-Ngoc H, Nguyen TV, Bui-Tien T, De Roeck G, Nguyen HX. Data-driven structural health monitoring using feature fusion and hybrid deep learning. IEEE Trans Autom Sci Eng 2021;18(4):2087–103. https://doi.org/10.1109/TASE.2020.3034401.

[12] Parisi F, Mangini AM, Fanti MP, Adam Adam JM. Automated location of steel truss bridge damage using machine learning and raw strain sensor data (doi.org:) Autom Constr 2022;138:104249. https://doi.org/10.1016/j.autcon.2022.104249.

[13] Ruggieri S, Cardellicchio A, Leggieri V, Uva G. Machine-learning based vulnerability analysis of existing buildings. Autom Constr 2021;132:103936. https://doi.org/10.1016/j.autcon.2021.103936.

[14] Cardellicchio A, Ruggieri S, Nettis A, Renò V, Uva G. Physical interpretation of machine learning-based recognition of defects for the risk management of existing bridge heritage. Eng Fail Anal 2023;149:107237. https://doi.org/10.1016/j.engfailanal.2023.107237.

[15] Mangalathu S, Hwang SH, Choi E, Jeon JS. Rapid seismic damage evaluation of bridge portfolios using machine learning techniques. Eng Struct 2019;201:109785. https://doi.org/10.1016/j.engstruct.2019.109785.

[16] Xu JG, Feng D, Mangalathu S, Jeon JS. Data-driven rapid damage evaluation for life-cycle seismic assessment of regional reinforced concrete bridges. Earth Eng Struct Dyn 2022;51(11):2730–51. https://doi.org/10.1002/eqe.3699.

[17] Noureldin M, Ali A, Sim S, Kim J. A machine learning procedure for seismic qualitative assessment and design of structures considering safety and serviceability. J Build Eng 2022;50:104190. https://doi.org/10.1016/j.jobe.2022.104190.

[18] Mangalathu S, Hwang SH, Choi E, Jeon JS. Rapid seismic damage evaluation of bridge portfolios using machine learning techniques. Eng Struct 2019;201:109785. https://doi.org/10.1016/j.engstruct.2019.109785.

[19] Lagaros ND, Fragiadakis M. Fragility assessment of steel frames using neural networks. Earthq Spectr 2007;23(4):735–52. https://doi.org/10.1193/1.279824128.

[20] Gavin HP, Yau SC. High-order limit-state functions in the response surface method for structural reliability analysis. Struct Saf 2008;30(2):162–79. https://doi.org/10.1016/j.strusafe.2006.10.00329.

[21] Gidaris I, Taflanidis AA, Mavroeidis GP. Kriging metamodeling in seismic risk assessment based on stochastic ground motion. Earth Eng Struct Dyn 2015;44(14):2377–99. https://doi.org/10.1002/eqe.2586.

[22] Mai HT, Kang J, Lee J. A machine learning-based surrogate model for optimization of truss structures with geometrically non-linear behavior. Fin Elem Anal Des 2021;196:103572. https://doi.org/10.1016/j.finel.2021.103572.

[23] Parida SS, Bose S, Butcher M, Apostolakis G, Shekhar P. SVD enabled data augmentation for machine learning based surrogate modeling of non-linear structures. Eng Struct 2023;280:115600. https://doi.org/10.1016/j.engstruct.2023.115600.

[24] Yang DY. Adaptive risk-based life-cycle management for large-scale structures using deep reinforcement learning and surrogate modeling. J Eng Mech 2022;148(1):04021126. https://doi.org/10.1061/(ASCE)EM.1943-7889.0002028.

[25] Vurtur Badarinath P, Chierichetti M, Davoudi Kakhki F. A machine learning approach as a surrogate for a finite element analysis: Status of research and application to one dimensional systems (2021) Sensors 2021;21(5):1654. https://doi.org/10.3390/s21051654.

[26] Shen L, Shen Y, Liang S. Reliability analysis of RC slab-column joints under punching shear load using a machine learning-based surrogate model. Buildings 2022;12(10):1750. https://doi.org/10.3390/buildings12101750.

[27] Soleimani-Babakamali MH, Esteghamati MZ. Estimating seismic demand models of a building inventory from non-linear static analysis using deep learning methods. Eng Struct 2022;266:114576. https://doi.org/10.1016/j.engstruct.2022.114576.

[28] Dabiri H, Faramarzi A, Dall'Asta A, Tondi E, Micozzi F. A machine learning-based analysis for predicting fragility curve parameters of buildings. J Build Eng 2022;62:105367. https://doi.org/10.1016/j.jobe.2022.105367.

[29] Bhatta S, Dang J. Seismic damage prediction of RC buildings using machine learning. Earth Eng Struct Dyn 2023. https://doi.org/10.1002/eqe.3907.

[30] Ruggieri S, Chatzidaki A, Vamvatsikos D, Uva G. Reduced-order models for the seismic assessment of plan-irregular low-rise frame buildings. Earth Eng Struct Dyn 2022;51(14):3327–46. https://doi.org/10.1002/eqe.3725.

[31] Naser MZ. An engineer's guide to explainable artificial intelligence and interpretable machine learning: Navigating causality, forced goodness, and the false perception of inference. Autom Constr 2021;129:103821. https://doi.org/10.1016/j.autcon.2021.103821.

[32] Tapeh ATG, Naser MZ. Discovering graphical heuristics on fire-induced spalling of concrete through explainable artificial intelligence. Fire Technol 2022;58:2871–98. https://doi.org/10.1007/s10694-022-01290-7.

[33] Somala SN, Karthikeyan K, Mangalathu S. Time period estimation of masonry infilled RC frames using machine learning techniques. Struct 2021;34:1560–6. https://doi.org/10.1016/j.istruc.2021.08.088.

[34] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comp Phys 2019;378:686–707.

[35] Raissi M, Karniadakis GE. Hidden physics models: machine learning of nonlinear partial differential equations. J Comp Phys 2018;357:125–41.

[36] Haghighat E, Raissi M, Moure A, Gomez H, Juanes R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Comp Met Appl Mech Eng 2021;379:113741. https://doi.org/10.1016/j.cma.2021.113741.

[37] A. Sanchez-Gonzalez, N. Heess, J.T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, P. Battaglia. Graph networks as learnable physics engines for inference and control. In International Conference on Machine Learning (2018, July) (pp. 4470–4479).

[38] P. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y., Li.R. Pascanu. Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv (2018):1806.01261. https://doi.org/10.48550/arXiv.1806.01261.

[39] Song LH, Wang C, Fan JS, Lu HM. Elastic structural analysis based on graph neural network without labeled data. Comp Aided Civ Infrastruct Eng 2022;38(10):1307–23. https://doi.org/10.1111/mice.12944.

[40] Sanchez-Lengeling B, Reif E, Pearce A, Wiltschko AB, A. B. A gentle introduction to graph neural networks. Distill 2021;6(9):e33.

[41] Alizadeh R, Allen JK, Mistree F. Managing computational complexity using surrogate models: a critical review. Res Eng Des 2020;31:275–98. https://doi.org/10.1007/s00163-020-00336-7.

[42] Bathe KJ. Finite element method. Wiley Encycl Comp Sci Eng 2007:1–12. https://doi.org/10.1002/9780470050118.ecse159.

[43] Aliabadi FM. Boundary element methods. Encycl Contin. Mech. Berlin, Heidelberg: Springer Berlin Heidelberg; 2020. p. 182–93. https://doi.org/10.1007/978-3-662-55771-6_18.

[44] Bićanić N. Discrete element methods. Wiley Encycl Contin. Mech.; 2007. https://doi.org/10.1002/0470091355.ecm006.pub2.

[45] Leggieri V, Ruggieri S, Zagari G, Uva G. Appraising seismic vulnerability of masonry aggregates through an automated mechanical-typological approach. Aut Constr 2021;132:103972. https://doi.org/10.1016/j.autcon.2021.103972.

[46] Ruggieri S, Calò M, Cardellicchio A, Uva G. Analytical-mechanical based framework for seismic overall fragility analysis of existing RC buildings in town compartments. Bull Earthq Eng 2022;20:8179–216. https://doi.org/10.1007/s10518-022-01516-7.

[47] K. O'Shea, R. Nash. An introduction to convolutional neural networks. *arXiv*. https://doi.org/10.48550/arXiv.1511.08458.

[48] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv (2013):1312.6203. https://doi.org/10.48550/arXiv.1312.6203.

[49] M. Simonovsky, N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *arXiv*. https://doi.org/10.48550/arXiv.1704.02901.

[50] Murtagh F, F. Multilayer perceptrons for classification and regression. Neurocomputing 1991;2(5–6):183–97. https://doi.org/10.1016/0925-2312(91)90023-5.

[51] Computer and Structures I. SAP2000, Advanced 24. Structural Analysis Program–Manual – Computer andStructures, Inc, Berkeley, California, USA, 2023.

[52] Harirchian E, Lahmer T, Rasulzade S. Earthquake hazard safety assessment of existing buildings using optimized multi-layer perceptron neural network. Energies 2020;13(8):2060. https://doi.org/10.3390/en13082060.

[53] Python Software Foundation, Python: A Dynamic, Open-Source Programming Language, Python Software Foundation, 2015. https://www.python.org/psf/.

[54] Grattarola D, Alippi C. Graph neural networks in tensorflow and keras with spektral. IEEE Comp Intel Mag 2021;16(1):99–106. https://doi.org/10.1109/MCI.2020.3039072.

[55] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* , 2017, pp. 1251–1258.

[56] Asteris PG, Nozhati S, Nikoo M, Cavaleri L. Krill herd algorithm-based neural network in structural seismic reliability evaluation. Mech Adv Mat Struct 2019;26(13):1146–53. https://doi.org/10.1080/15376494.2018.1430874.

[57] Le V, Caracoglia L. A neural network surrogate model for the performance assessment of a vertical structure subjected to non-stationary, tornadic wind loads. Comp Struct 2020;231:106208. https://doi.org/10.1016/j.compstruc.2020.106208.

[58] Makoond N, Pelà L, Molins C. Robust estimation of axial loads sustained by tie-rods in historical structures using artificial neural networks. Struct Health Monit 2022;22(4). https://doi.org/10.1177/14759217221123326.

[59] Rahmanpanah H, Mouloodi S, Burvill C, Gohari S, Davies HM. Prediction of load-displacement curve in a complex structure using artificial neural networks: a study on a long bone. Intern J Eng Sci 2020;154:103319. https://doi.org/10.1016/j.ijengsci.2020.103319.

[60] A. Boyd, A. Czajka, K. Bowyer, Deep learning-based feature extraction in iris recognition: Use existing models, fine-tune or train from scratch? In Proceedings: *2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems* (2019), pp. 1–9, https://doi.org/10.1109/BTAS46853.2019.9185978.

[61] Ying X. An overview of overfitting and its solutions. J Phys: Conf Ser 2019;1168:022022. https://doi.org/10.1088/1742-6596/1168/2/022022.

[62] Akbarpour A, Mahdikhani M, Moayed RZ. Effects of natural zeolite and sulfate ions on the mechanical properties and microstructure of plastic concrete. Front Struct Civ Eng 2022;16(1):86–98. https://doi.org/10.1007/s11709-021-0793-x.

[63] Akbarpour A, Mahdikhani M, Moayed RZ. Mechanical behavior and permeability of plastic concrete containing natural zeolite under triaxial and uniaxial compression. J Mat Civ Eng 2022;34(2):04021453. https://doi.org/10.1061/(ASCE)MT.1943-5533.0004093.

[64] Akbarpour A, Mahdikhani M. Effects of natural zeolite and sulfate environment on mechanical properties and permeability of cement–bentonite cutoff wall. Eur J Env Civ Eng 2023;27(3):1165–78. https://doi.org/10.1080/19648189.2022.2075940.