Journal of Science and Technology in Civil Engineering, HUCE, 2023, 17 (2): 99–109

TOWARDS A GENERALIZED SURROGATE MODEL FOR TRUSS STRUCTURE ANALYSIS USING GRAPH LEARNING

Dang Viet Hung[a,*], Nguyen Trong Phu[a]

[a]*Faculty of Building and Industrial Construction, Hanoi University of Civil Engineering, 55 Giai Phong road, Hai Ba Trung district, Hanoi, Vietnam*

Article history:
Received 12/12/2022, Revised 05/3/2023, Accepted 08/3/2023

Abstract

Truss analysis has been well investigated by researchers and engineers with a large number of structural analysis software that can quickly and reliably provide analysis results. However, these methods require either expensive commercial software or self-developed in-house codes based on structural expertise along with advanced programming skills. Thus, incorporating this software into other complex applications, such as an online structural analysis framework, multiple-objectives truss optimization, structural reliability, etc., is highly challenging. This study proposes a novel and efficient surrogate model for performing truss analysis based on graph theory and deep learning algorithms, which is applicable for various truss topologies with different load scenarios without requiring retraining as other Deep Learning (DL)-based counterparts. The truss connectivity information is expressed through adjacency matrices, while material properties, external loads, and boundary conditions are considered node features. The performance and efficiency of the proposed methods are successfully validated with numerous unseen 2D trusses, providing highly similar results compared to the conventional finite element method.

Keywords: structural analysis; deep learning; truss structure; graph data; finite element method.

https://doi.org/10.31814/stce.huce2023-17(2)-09

1. Introduction

Structural analysis is a fundamental process providing structures' responses, such as internal forces, displacements, etc., in response to different external excitations, helping engineers attain adequate and safe design solutions. In the past, this is a tedious task taking thousands of hours of hand calculations with pen and paper, and prone to human errors. With the invention of the finite element method and the rapid development of computers, many structural analysis softwares have been developed, allowing structural engineers to accomplish the analysis of large-scale structures within a few hours. The realization steps of FEM involve five main steps: (i) draw a computer-aided design model and divide the model into finite elements. (ii) establish an equilibrium equation for each element, (iii) construct the global system of equations from all elements, (iv) Solve the global equations, and (v) obtain and visualize calculation results. One of the state-of-the-art structural analysis software is SAP2000 [1], which can perform various types of analysis, including static, dynamic, transient, linear, and nonlinear analysis. Other powerful software widely used in the construction industry are Etabs, Dlubal, Abaqus, StruCalc, etc. These commercial FEM softwares can address almost all common structures; however, their prices are expensive and not very flexible to be customized for the specific requirements of individual users. Hence, some authors have developed free, open-source structural analysis software, making FEM available to more users, including students and non-profit researchers. Examples of some reliable open-source software are OpenSees by Pacific Earthquake

*Corresponding author. *E-mail address:* hungdv@huce.edu.vn (Hung, D. V.)

Engineering Research Center [2], Code Aster by Électricité de France [3], FEAP by Taylor at UC Berkeley [4], etc.

With a wave of new technologies such as cloud computing, smartphone, and artificial intelligence, it is desirable that structural analysis can be carried out in a near-real-time and collaborative manner, just improving the efficacy and efficiency of the design flow. A leading cloud structural analysis framework is SkyCiv [5], with which users can simulate a whole complex structure and design its structural members according to various international standards without the need to install any software. CloudCalc [6] is another online structural analysis framework that is available on multiple devices, such as PC, tablets, and smartphones. SimScale [7] is a powerful cloud-based FEM with which engineers can perform resource-intensive computations with thousand-element models via a common personal laptop connected to a cloud computing server. However, to have full access to these cloud-based frameworks, one needs to pay a relatively high fee of around 100 dollars per month. Moreover, productivity largely depends on difficult-to-control external factors such as Internet transmission stability, cyber security, shared information management, etc.

A more affordable approach is to build a data driven-surrogate model based on machine learning (ML) or deep learning (DL) algorithms and FEM databases. Such direction has gained increased attention from the engineer and research community thanks to its portability, reasonable accuracy, openness, and low service costs. To quickly estimate structures' limit state functions, the authors in [8] developed an artificial neural network (ANN)-based surrogate model and employed it to perform structural reliability analysis. Su et al. [9] proposed to utilize the Gaussian Process Regression model for forecasting the nonlinear responses of a large-scale suspension bridge. Hung et al. [10] constructed a deep learning-based surrogate model for predicting responses of structures subjected to time-varying wind loads with reasonably accurate results. Thanks to their high efficiency, the surrogate models are applied to other compute-intensive tasks such as incremental dynamic analysis [11], high-dimensional low-failure probability problems [12], etc. Li and Zhang [13] investigated a surrogate model based on physics-informed deep learning to predict dynamic responses of wind turbine structures under both cyclic wind and wave loads. Another deep learning algorithm, namely, convolution neural network, has been widely used by various authors to build surrogate models, e.g., the works of [14, 15]. Unlike FEMs that require structural analysis knowledge and specialized software to set up and run, surrogate models are easy to be integrated into any application on different devices. Besides, the DL-based models benefit from highly optimized deep learning libraries; their implementation and computational time are very fast, especially with the support of parallelization calculation and GPU devices.

Though conventional surrogate models are efficient and reasonably accurate, their generality is still in question. In other words, almost surrogate models are problem specific, i.e., they can be well-trained on a specific structure with given geometrical layouts; however, they cannot be directly applied to other structures even with minor changes in topology. In order to improve the generalization of the DL-based surrogate model, we develop a novel framework named Graph-DL, which can be readily applied to new structures and provide satisfied results without retraining. The proposed method consists of two steps. The first step involves converting structures' information into equivalent graph data. The second step is to design a graph deep learning architecture that takes graph data obtained from step one as input and provides quantities of interest, such as node displacements. It is noted that a truss is a special kind of structure composed of lightweight bars only subjected to axial forces, while bending and shear deformations are insignificant and usually neglected in the calculation. That is why the truss analysis process is widely considered more straightforward than other structures such

as frames, dams, bridges, etc. Hence, one selects trusses as the main subject in this study which is one of the first attempts toward a generalized surrogate approach, according to the best of our knowledge.

The remainder of this article is organized as follows. Section 2 describes in detail the components of the proposed graph learning model; Section 3 demonstrates the applicability of the proposed approach via numerical experiments. Finally, Section 4 concludes and suggests some ideas for future works

2. Methodology

2.1. Brief of truss analysis

A truss is a structure composed of multiple straight bars connected together via pinned connections. It is designed, fabricated and constructed to be subjected to only axial forces and longitudinal deformation. The cross-section sizes of the truss bars are significantly smaller than their length. Normally, the axial force is uniform across truss bars; thus uniform cross-sections are utilized. Given a planar truss joint 'i', there are two displacements in horizontal and vertical directions to be determined, which are denoted by $u = \{u_{ix}, u_{iy}\}$. Thus, for the whole truss structure, the displacement vector is $U = \{u_{1x}, u_{1y}, u_{2x}, u_{2y}, \ldots, u_{Nx}, u_{Ny}\}$ with N is the total number of joints. Per conventional design practice for truss structures, loads are only applied at truss joints; therefore, in reality, there is usually a secondary structure directly prone to external sources before transmitting loads to the truss. External loads can be described via a load vector $F = \{F_{1x}, F_{1y}, F_{2x}, F_{2y}, \ldots, F_{Nx}, F_{Ny}\}$.

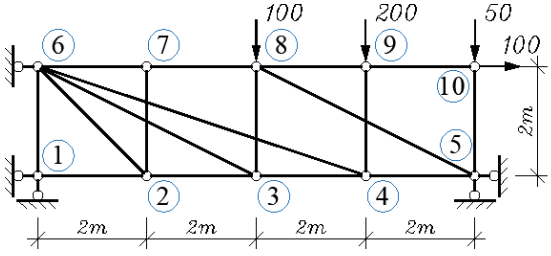
Next, the displacement vector is obtained by solving the following equilibrium equations:

$$K \times U = F \tag{1}$$

where K is the global stiffness matrix obtained by assembling multiple elementary stiffness matrices of truss bars with the help of the truss connectivity and the global-local coordinate transformations.

2.2. Truss description via graph data

From the graph theory perspective, a graph G is fully described by four components $G = (V, E, A, H)$ where V represents the set of graph nodes, E is the set of graph edges, A the adjacency matrix, and H the node feature vector. When applying to truss structures, V naturally is the truss joint, E refers to truss bars, $A_{ij} = 1$ if there is a truss bar between joint i and joint j, if not, $A_{ij} = 0$. Relied on the adjacency matrix A, one introduces $ne(v)$ as a set of neighbor nodes connected to node v. For the node feature vectors $H = [h_1, h_2, \ldots, h_V]$, a vector h_i will encompass joint coordinates, concentrated load values, boundary conditions. Moreover, since the axial rigidity (the product of cross-section area and elasticity module) of truss bars is important information, one converts these edge-related features to node features by introducing an array of axial rigidity values to the feature vector of each node. To clarify these descriptions, a representative example is presented as follows.



	1	2	3	4	5	6	7	8	9	10
1	1	1	0	0	0	1	0	0	0	0
2	1	1	1	0	0	1	1	0	0	0
3	0	1	1	1	0	1	0	1	0	0
4	0	0	1	1	1	1	0	0	1	0
5	0	0	0	1	1	0	0	1	0	1
6	1	1	1	1	0	1	1	0	0	0
7	0	1	0	0	0	1	1	1	0	0
8	0	0	1	0	1	0	1	1	1	0
9	0	0	0	1	0	0	0	1	1	1
10	0	0	0	0	1	0	0	0	1	1

Figure 1. Example of truss structure with node numbering and corresponding adjacency matrix

Fig. 1 shows a truss structure composed of 10 joints and 17 bars with corresponding dimensions, boundary conditions and loadings. For simplicity, it is supposed that all bars have the same cross section $S = 0.02\ \text{m}^2$, Young's modulus $E = 210$ GPa. With this truss configuration, the corresponding adjacency matrix is illustrated on the right of the figure. For example, joint 2 is connected to joints 1, 3, 6, 7 but not to the rest, hence, $A_{12} = A_{23} = A_{26} = A_{27} = 1$, otherwise $A_{24} = A_{25} = A_{28} = A_{29} = A_{2,10} = 0$. In terms of node features, they are represented in the form of tabular data as shown in Table 1.

Table 1. The table of node features including node coordinates, boundary conditions, applied loads and member rigidity

Joint	X (m)	Y (m)	x-BC	y-BC	P_x (kN)	P_y (kN)	Axial rigidity $\times 1e9$ (N)
1	0	0	1	1	0	0	[0, 4.2,0,0,0,4.2,0,0,0,0]
2	2	0	0	0	0	0	[4.2, 0, 4.2, 0,0,4.2,4.2,0,0,0]
3	4	0	0	0	0	0	[0, 4.2, 0, 4.2, 0, 4.2, 0, 4.2, 0, 0]
4	6	0	0	0	0	0	[0, 0, 4.2, 0, 4.2, 4.2, 0, 0, 4.2, 0]
5	8	0	1	1	0	0	[0, 0, 0, 4.2, 4.2, 0, 0, 4.2, 0, 4.2]
6	0	2	1	0	0	0	[4.2, 4.2, 4.2, 4.2, 0, 4.2, 4.2, 0, 0, 0]
7	2	2	0	0	0	0	[0, 4.2, 0, 0, 0, 4.2, 4.2, 4.2, 0, 0]
8	4	2	0	0	0	-100	[0, 0, 4.2, 0, 4.2, 0, 4.2, 4.2, 4.2, 0]
9	6	2	0	0	0	-200	[0, 0, 0, 4.2, 0, 0, 0, 4.2, 4.2, 4.2]
10	8	2	0	0	100	50	[0, 0, 0, 0, 4.2, 0, 0, 0, 4.2, 4.2]

(*) x-BC, y-BC are boundary conditions in x- and y-directions.

2.3. *Graph learning-based prediction model*

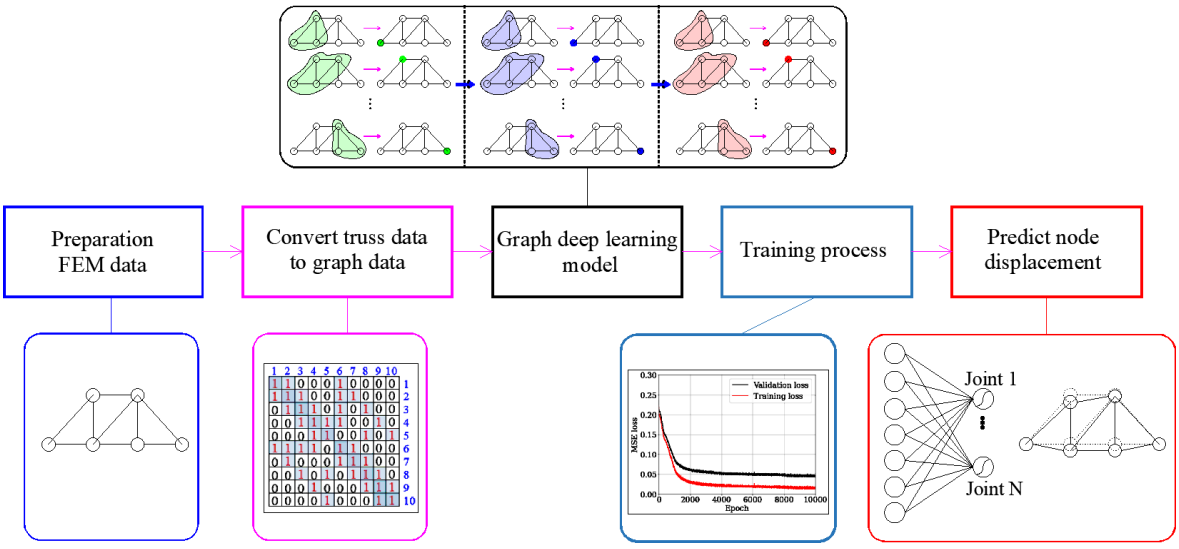


Figure 2. Working flow of the proposed Graph-DL framework

The overall working flow of the proposed Graph-DL framework is presented in Fig. 2. It is acknowledged that neural networks are one of the most important algorithms in artificial intelligence which significantly contribute to a large number of recent advanced technologies. The networks consist of multiple layers of neurons connected in a strategically planned way. At the beginning, the

networks' weights are randomly initialized. After that, they are progressively adjusted via the back propagation algorithm such that predictions by the model approximate actual values to a satisfied level of precision. Among various neural network architecture, Graph Neural Network is a special one that is designed for handling graph data. Moreover, with the recent development of Deep Learning, deep graph neural networks are devised to significantly improve the performance of GNN. The backbone intuition of the GNN, arguably pioneered by Scarselli et al. [16], is to learn new node representations via the message passing mechanism which aggregates the nodes' features with those of their neighbors. Mathematically, the deep GNN can be described as follows. Given a layer l, and two nodes i and j connected one to the other. The message between these two nodes is computed by:

$$m_{ij}^{l} = \sigma_e(W_e^l, h_i^l, h_j^l, A_{ij}) \tag{2}$$

where h_i^l, h_j^l are node representations of nodes i, j at layer l, respectively. W^l are the model's weights at layer l, σ is the non-linear activation function RELU. Note that if these two nodes are not connected, i.e., $A_{ij} = 0$, then $m_{ij} = 0$. Next, one combines the messages of all neighbors of node i via an aggregation operation (sum, average, max, etc.). Here, the summation is used as below:

$$m_i^l = \sum_{j \in Ne(i)} m_{ij}^l \tag{3}$$

After that, the new representation of node i is computed via a transformation with learnable weight W_h^{l+1} as follows:

$$h_i^{t+1} = \sigma(W_h^{l+1}, m_i^l) \tag{4}$$

Apparently, the parameters of layer l are different from those of other layers.

Furthermore, one can stack multiple GNN layers to make the network deeper and more powerful because it can learn high-order neighborhoods of nodes. A node of L-layer GNN can aggregate features from nodes which are L step away from it. For example, using a 3-layer GNN for the truss structure in Fig. 1, one can incorporate the features from node 10 into the new representation of node 7. However, using a too deep network will significantly increase the model complexity and quickly increase the computational cost. For such reasons, in this study, a 3-layer GNN is utilized. On the other hand, the outputs of interest are node displacement, therefore, one places on top of the deep GNN, a fully-connected layer and an output layer with multiple neurons for predicting the displacements of all truss joints simultaneously. To evaluate the disparity between predicted and actual values, the root mean square (RMS) loss function is employed:

$$RMS = \sqrt{\sum_{i=1}^{N_{joint}} \frac{\left[\left(u_{i,x}^{pred} - u_{i,x}\right)^2 + \left(u_{i,y}^{pred} - u_{i,y}\right)^2\right]}{2 \times N_{joint}}} \tag{5}$$

where N_{joint} is the total number of truss joints.

Note that an advantage of GNN over other DL/ML-based models when working with structural data is that it is insensitive to the element and joint orders, i.e., with different numbering schemes, the model still provides the same results without fine-tuning. This is not the case for other models when the order of features is needed to be strictly respected. This advantage makes the proposed model is more flexible and practicable for further applications.

In terms of the training process, the deep GNN follows a conventional process widely used with the optimizer being the Adam algorithm, the learning rate initially set at 0.001, then halved after each

10 consecutive non-improvement iterations. The training process is terminated after 10000 iterations or until the learning rate falls below 10^{-5}. The framework, including data generation, graph data preparation, model design, training and validation are developed by the authors with the help of the PYG library [17] for the graph neural network base, Pytorch library for the training process pipeline, and Matplotlib library for visualization.

3. Numerical experiments

In order to prepare the truss structure database, one conducted extensive simulations of numerous truss structures with the help of the open source and reliable FEM software OpenSees. In total there are 1000 simulations with different geometrical configurations, boundary conditions, cross-section assignments, applied loads, etc. In terms of structural stability, we did not consider unstable structures because their output results would be NaN (not a number) that could not be taken into account by the RMS loss function. If, in the future, we want the DL-based surrogate model to be able to handle both unstable and stable structures, it should include an additional classifier. If the structure is unstable, a warning error will be raised, and the calculation will be stopped; otherwise, if the structure is stable, the prediction will be carried out. Such a classifier will be trained with a binary classification loss function. Examples of different truss layouts used to generate the database are presented in Fig. 3. Specifically, the truss span varies in the range of [1 m, 10 m], the cross-section area [0.001 m^2, 0.01 m^2], the load intensity [1 kN, 1000 kN], Young modulus [200 MPa, 1000 MPa].

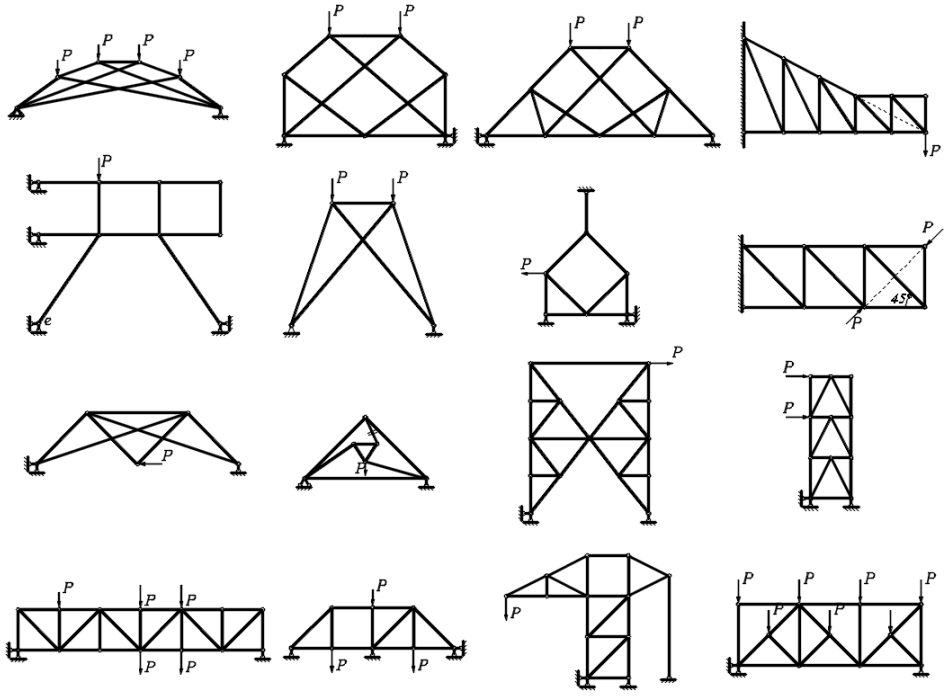


Figure 3. Examples of truss structures utilized for generating training samples

The output of interest of each simulation is the joint displacement in both x- and y-directions. On the other hand, one generates a 200-size validation dataset with truss structures different from those in the training data by modifying to some extent the span length, cross-section size, load conditions, and especially adding or deleting some truss members.

Note that to improve the model performance, one incorporates some engineering knowledge into the model when predicting the truss joint displacements such as no displacement for joints with

boundary conditions, truss constructability check, i.e., the number of truss bars being at least two times greater than the number of joints, etc.

The training process of the graph DL model is carried out with the loss function, and training settings described in the previous section. The learning rate (lr) is an important hyperparameter that affects the balance between model performance and training convergence speed. If a too-small value is used, the training curve is more stable, but it takes significant computational time to achieve the convergence; in contrast, using a high learning rate value, obtained results may not be optimal and even divergent. In order to select an appropriate learning rate for a given deep learning problem, it is often required to perform preliminary experiments because there does not exist a learning rate that is suitable for all problems. In this study, the authors tested with different values of learning rates: 0.005, 0.001, and 0.0001; the corresponding learning curves are presented in Fig. 4 showing that the learning curve with $lr = 0.001$ converges faster than that of $lr = 0.0001$ and provides smaller MSE loss values. More specifically, the MSE loss drastically decreased for around 2000 first iterations (epochs) before slowly improving in the period [2000-9000]. From epoch 9000, noticeable improvement is hardly observed, and the training process is terminated at epoch 10000. The configuration with the lowest MSE loss value on the validation dataset is saved and utilized as the final model. The CPU time of the training process with $lr = 0.001$ is 93 minutes on a computation server equipped with a RTX 3090 GPU, Xeon E5-2643 CPU, and 64 Gb ram memory.

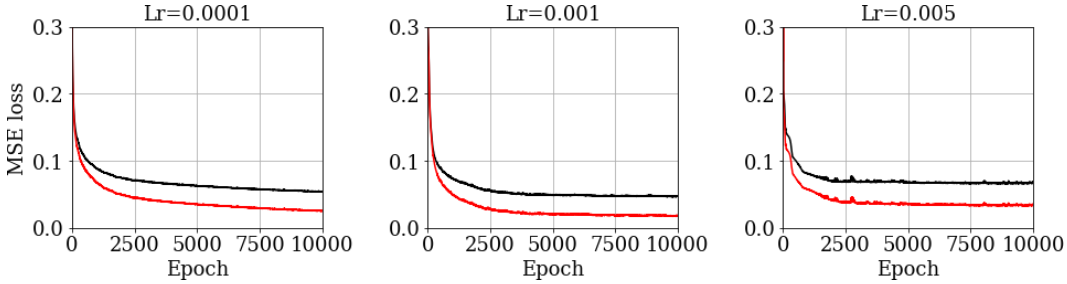


Figure 4. Evolution curves of MSE loss functions on the training and validation datasets with different learning rates

Table 2. Comparison results on joint displacements for 10-node truss

Joint	1		2		3		4		5	
	x	y	x	y	x	y	x	y	x	y
by FEM (mm)	0.0	0.0	−1.6	−4.3	−3.2	10.9	−8.4	−171.4	0.0	0.0
by Graph DL (mm)	0.0	0.0	−1.5	−4.3	−3.2	11.4	−8.3	−170.9	0.0	0.0
Difference (%)	0.0	0.0	−6.3	0.0	0.0	4.6	−1.2	−0.3	0.0	0.0
Joint	6		7		8		9		10	
	x	y	x	y	x	y	x	y	x	y
by FEM (mm)	0.0	−2.7	14.5	−4.4	29.1	12.7	51.7	−175.9	74.4	11.3
by Graph DL (mm)	0.0	−2.6	14.3	−4.6	29.2	12.9	51.4	−176.9	74.5	11.4
Difference (%)	0.0	−3.7	−1.4	4.5	0.3	1.6	−0.6	0.6	0.1	0.9

Examples of joint displacement predictions on the validation dataset are represented in Tables 2 and 3 for trusses in Fig. 5 with 10 joints and 17 bars with 14 joints and 25 bars, respectively. The tables enumerate displacements at all joints obtained by FEM and the Graph DL method Apparently,

Table 3. Comparison results on joint displacements for 14-node truss

Joint	1		2		3		4		5	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*	*x*	*y*
by FEM (mm)	0.0	0.0	0.0	50.5	−9.0	0.0	0.0	0.0	50.5	−9.0
by Graph DL (mm)	0.0	0.0	0.0	51.5	−9.1	0.0	0.0	0.0	51.5	−9.1
Difference (%)	0.0	0.0	0.0	2.0	1.1	0.0	0.0	0.0	2.0	1.1
Joint	6		7		8		9		10	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*	*x*	*y*
by FEM (mm)	−41.3	53.5	−41.4	0.0	−23.8	8.3	−29.5	53.7	−29.6	79.7
by Graph DL (mm)	−41.3	52.4	−41.3	0.0	−23.2	7.7	−29.3	52.0	−29.8	79.0
Difference (%)	0.0	−2.1	−0.2	0.0	−2.5	−7.2	−0.7	−3.2	0.7	−0.9
Joint	11		12		13		14			
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*		
by FEM (mm)	−2.0	9.9	−0.9	9.3	−1.2	5.3	−1.7	9.1		
by Graph DL (mm)	−2.1	10.0	−0.9	9.2	−1.2	5.3	−1.7	9.2		
Difference (%)	5.0	1.0	0.0	−1.1	0.0	0.0	0.0	1.1		

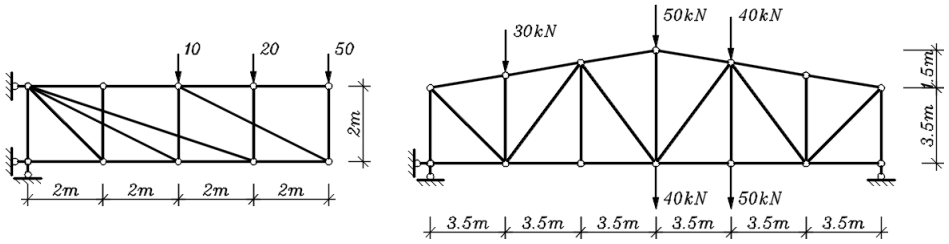


Figure 5. Examples of validation trusses

the proposed method can provide results with relative errors of less than 4% for all node displacements. Note that, these truss structures are geometrically different to those in the training dataset. Normally, the DL/ML-based method can be trained for a specific structure, and then employed for similar structures with slightly different external loadings or structural, material properties but cannot be applied to those with different configurations. In other words, one needs to train two different models for two geometrically different truss structures. Meanwhile, the proposed method is applicable to various truss configurations which is the most noteworthy point of this study.

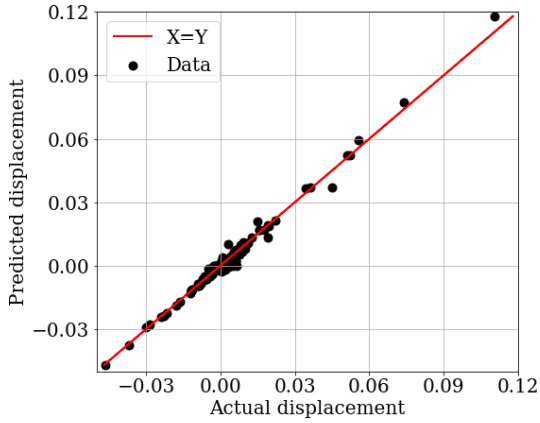


Figure 6. Comparison results on joint displacements for all the validation dataset

For an overview of the model performance, one plots the predicted values on the whole validation dataset in Fig. 6, where the x and y-coordinates of scatter points denote actual and predicted values obtained by FEM and Graph DL, respectively.

The red 45-degree line denotes the ideal curve where the predicted results are exactly equal to the actual ones. It can be seen that almost scatter points lie around the ideal curves, i.e., the proposed method yield qualitatively accurate displacement predictions. On the other hand, to quantitatively assess the errors committed by Graph DL, one computes the relative error between predicted values with actual ones and plot a histogram in Fig. 7 graph showing the computed errors' statistics. It can be seen that the obtained histogram has a typical bell shape of a normal distribution with almost all values ranging between [−6%, 6%]. In other words, the histogram is not skewed in any sides (positive or negative), a major part of errors concentrates around zero values, which means the Graph DL model provides highly accurate results most of the time. Furthermore, data-driven predictive model is a complementary and supplementary method to the conventional FEM which is easy to use without installing any specific software. The user simply needs to input the table of node features and adjacency matrix, and the model will quickly yield displacements at all joints, bypassing a sequence of FEM operations such as node and element numbering, coordinate transformations, constructing stiffness matrix, etc.

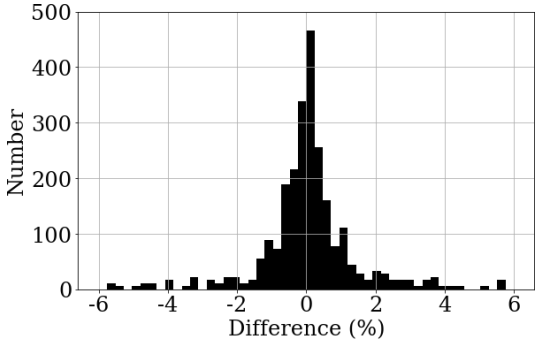


Figure 7. Comparison results on joint displacements for all the validation dataset

However, there is still a long way ahead to achieve a generalized prediction model which can provide satisfied results on any truss configuration. For example, one tests Graph DL on a completely new truss configuration with more joints and more complex geometry (19 joints and 36 bars) than any truss in the training data, shown in Fig. 8 The prediction results for joints 2, 3, 5, 6, 7 are listed in Table 4, showing large errors up to 54.2%.

In terms of computation time, at the inference time, the FEM using OpenSees requires 9.8 ms for 100 calculations, whereas the CPU times needed by the trained Graph-DL method is 8.1 ms when executed on a RTX 3090 GPU and 17.3 ms on a Xeon E5-2643 CPU. It can be seen that the DL-based surrogate model is also highly efficient, and currently, almost DL libraries are fully supported to run on GPU devices for boosting the computational times. On the other hand, not all FEM programs are able to run on GPU devices, especially in-house codes, because it requires advanced programming knowledge, which could be a technical obstacle for most structural engineers.

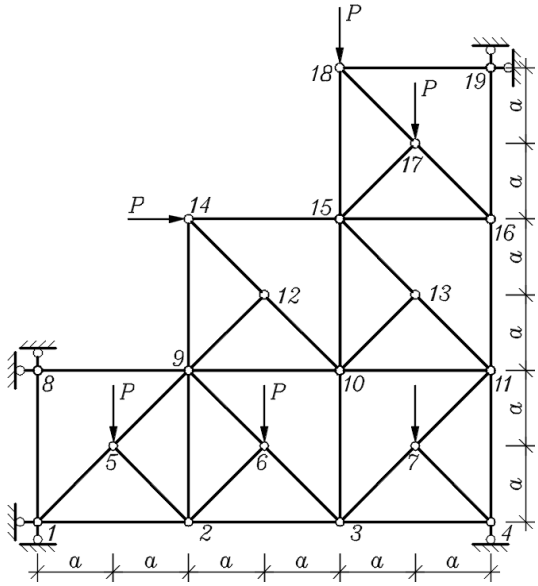


Figure 8. Schematic representation of a 19-joint and 36-bar truss structure

Table 4. Comparison results on joint displacements for the 19-joint and 36-bar truss structure

Joint	2		3		5		6		7	
	x	y	x	y	x	y	x	y	x	y
by FEM (mm)	39.0	−321.2	77.9	−366.2	−9.0	126.5	−256.1	49.8	−354.5	−49.8
by Graph DL (mm)	56.2	−415.4	91.6	−485.2	−9.1	195.1	−179.4	23.2	−524.4	−43.5
Difference (%)	44.3	29.3	17.6	32.5	1.1	54.2	−29.9	−53.4	47.9	−12.7

4. Conclusions

In this paper, a novel alternative method for the analysis of truss structures has been presented, implemented, and validated. The method explicitly takes the truss connectivity as input in addition to a feature tensor, including joint coordinates, member properties, and applied loads. Moreover, a multi-output deep graph neural network is devised to effectively handle these inputs and provide estimations of node displacements. Based on the obtained computation results, some conclusions can be drawn as follows:

- The method is extendable and applicable for various truss configurations without the need for retraining models, unlike other DL-based surrogate models only applied to a specific truss structure.

- At the inference step, the proposed method does not require any specialized software or special hardware devices, i.e., it can run on a common personal computer and quickly provide prediction results given the trusses' geometrical, mechanical, and loading data. Thus, it can be embedded into low-performance edge devices such as cloud applications, smartphones, robot brains, etc.

For the next step study, it is noteworthy to improve the model performance when working with truss configuration completely different from trained data, i.e., significantly larger in size, curved layout. This can be done by preparing big data of truss structures, using a deeper graph network, and including more engineering knowledge in the framework. It is also desirable to extend the proposed method for 3D truss structures with up to thousands of members. Another promising direction is to extend the proposed method to non-linear analysis. For this purpose, it is necessary to prepare relevant databases in advance with the help of FEM; the database should include the structures' responses in different regimes such as linear elastic, non-linear elastic, plastic, etc. Next, the principle of the proposed method, as described in Fig. 2, can be utilized in a straightforward fashion. However, it will require more complex deep learning architectures and take more training time, and it is also expected that discrepancies between predicted non-linear results with actual ones will be greater than those corresponding to linear results.

Acknowledgement

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 107.02-2021.03.

References

[1] Wilson, E. L. (2015). *CSI Analysis Reference Manual For SAP 2000, ETABS, SAFE and CS I Bridge*. Berkeley: Computer & Structures Inc.

[2] McKenna, F. (2011). OpenSees: A Framework for Earthquake Engineering Simulation. *Computing in Science & Engineering*, 13(4):58–66.

[3] Proix, J. M., Laurent, N., Hemon, P., Bertrand, G. (2000). *Code Aster, manuel de référence*. Fascicule R, 8.

[4] Taylor, R. L. (2014). *FEAP-A finite element analysis program*.

[5] SkyCiv Engineering (2022). *Online 3D structural analysis and design software*. Visited on 12/10/2022.

[6] CloudCalc, Inc. (2022). *Engineering Software in the Cloud*. Visited on 20/10/2022.
[7] SimScale (2022). *Innovate faster with cloud-native simulation*. Visited on 05/11/2022.
[8] Chau, K. W. (2007). Reliability and performance-based design by artificial neural network. *Advances in Engineering Software*, 38(3):145–149.
[9] Su, G., Peng, L., Hu, L. (2017). A Gaussian process-based dynamic surrogate model for complex engineering structural reliability analysis. *Structural Safety*, 68:97–109.
[10] Hung, D. V., Thang, N. T. (2022). Predicting dynamic responses of frame structures subjected to stochastic wind loads using temporal surrogate model. *Journal of Science and Technology in Civil Engineering (STCE) - HUCE*, 16(2):106–116.
[11] Nguyen, T.-T., Dang, V.-H. (2022). Probabilistic incremental dynamic analysis of structures using temporal surrogate model. *Applied Intelligence*.
[12] Nguyen, T.-T., Dang, V.-H., Nguyen, H. X. (2022). Efficient framework for structural reliability analysis based on adaptive ensemble learning paired with subset simulation. *Structures*, 45:1738–1750.
[13] Li, X., Zhang, W. (2022). Physics-informed deep learning model in wind turbine response prediction. *Renewable Energy*, 185:932–944.
[14] Xue, J., Xiang, Z., Ou, G. (2021). Predicting single freestanding transmission tower time history response during complex wind input through a convolutional neural network based surrogate model. *Engineering Structures*, 233:111859.
[15] Li, H., Wang, T., Wu, G. (2021). Probabilistic safety analysis of coupled train-bridge system using deep learning based surrogate model. *Structure and Infrastructure Engineering*, 19(8):1138–1157.
[16] Scarselli, F., Tsoi, A. C., Gori, M., Hagenbuchner, M. (2004). Graphical-Based Learning Environments for Pattern Recognition. In *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 42–56.
[17] Fey, M., Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*.