Development of deep learning-based joint elements for thin-walled beam structures

Jeon, Jaemin; Kim, Jaeyong; Lee, Jong Jun; Shin, Dongil; Kim, Yoon Young

DOI
10.1016/j.compstruc.2021.106714

Publication date
2022

Document Version
Final published version

Published in
Computers and Structures

Citation (APA)
Jeon, J., Kim, J., Lee, J. J., Shin, D., & Kim, Y. Y. (2022). Development of deep learning-based joint elements for thin-walled beam structures. *Computers and Structures*, *260*, Article 106714. https://doi.org/10.1016/j.compstruc.2021.106714

Computers and Structures 260 (2022) 106714

Contents lists available at ScienceDirect

Computers and Structures

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

Development of deep learning-based joint elements for thin-walled beam structures

Jaemin Jeon [a], Jaeyong Kim [a], Jong Jun Lee [a], Dongil Shin [b,*], Yoon Young Kim [a,*]

[a] Department of Mechanical Engineering and Institute of Advanced Machines and Design, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Republic of Korea
[b] Faculty of Mechanical, Maritime, and Materials Engineering, Delft University of Technology, Mekelweg 5, 2628 CD Delft, Netherlands

ARTICLE INFO

Article history:
Received 23 June 2021
Accepted 26 October 2021

Keywords:
Deep learning
Stiffness matrix
Eigendecomposition
Finite element analysis

ABSTRACT

This study presents a new modeling technique to estimate the stiffness matrix of a thin-walled beam-joint structure using deep learning. When thin-walled beams meet at joints, significant sectional deformations occur, such as warping and distortion. These deformations should be considered in the one-dimensional beam analysis, but it is difficult to explicitly express the coupling relationships between the beams' deformations connected at the joint. This study constructed a deep learning-based joint model to predict the stiffness matrix of a higher-order one-dimensional super element that presents the relationships. Our proposition trains the neural network using the eigenvalues and eigenvectors of the joint's reduced stiffness matrix to satisfy the correct number of zero-strain energy modes overcoming the randomly perturbed error of the deep learning. The deep learning-based joint model produced compliance errors mostly within 2% for a given structural system and the maximum error of 4% in the worst case. The newly proposed methodology is expected to be widely applicable to structural problems requiring the stiffness of a reduction model.

1. Introduction

Even though computing power has improved dramatically for the last few decades, low-dimensional beam-based finite element models still receive much attention in the vehicle [1–3] and civil industry fields [4,5] because of their design-friendliness. No doubt, the beam models facilitate the initial concept design [6–8] by the computation efficiency embedded in the models compared to shell or solid models, especially for iterative analyses unavoidable in optimization problems or concept designs [9]. However, the classical six degrees-of-freedom (DOFs) beam models are inaccurate for thin-walled beam structures. The inaccuracy results from complex sectional deformations occurring in the cross-section of a thin-walled beam, especially near the joint as shown in Fig. 1(a), because they cannot be modeled accurately by the classical beam models [10–17]. While higher-order beams employing more degrees of freedom than those used in the classical beam models can predict solution behavior correctly, matching the field quantities at a joint of thin-walled beams or deriving the joint stiffness matrix suitable for the matching is difficult. Here, we propose a new deep learning (DL)-based joint model using a higher-order one-dimensional (1D) beam theory, which involves the DOFs associated with sectional deformations.

Because the joint stiffness or flexibility accounts for significant local sectional deformations of the thin-walled structure, inaccurate estimation of the joint region results in poor solution accuracy over the whole structure. In previous works, one corrected the stiffness of the joint node to represent the joint flexibility while using the classical beam model [2,18], and some researchers focused on defining the coupling relationships between the complicated deformations of beams connected at the joint node considering advanced beam models [6–8,19–21]. However, these methods fundamentally define the joint as a single node which is limited in reflecting the stiffness according to the shape of the joint. On the other hand, some researchers modeled the joint using higher-dimensional elements, such as shell elements, to express the stiffness of complicated joints [1,22–24]; however, these methods require relatively high computational costs not expected in 1D beam modeling. The newly proposed DL-based joint model reliably expresses the joint flexibility as a super element. The joint stiffness matrix is derived from a shell-based shape description [1,22–24] to maintain solution accuracy [1,2,6–8,18–24] and use DL with the higher-order beam theory (HoBT) [10] for the 1D modeling performing in real-time [6–8,19–21].

The DL [25,26] used in this paper could be decisive in structural analysis for creating an analysis model for a challenging problem to

* Corresponding authors.
E-mail addresses: D.Shin-1@tudelft.nl, alamosds@snu.ac.kr (D. Shin), yykim@snu.ac.kr (Y.Y. Kim).

https://doi.org/10.1016/j.compstruc.2021.106714
0045-7949/

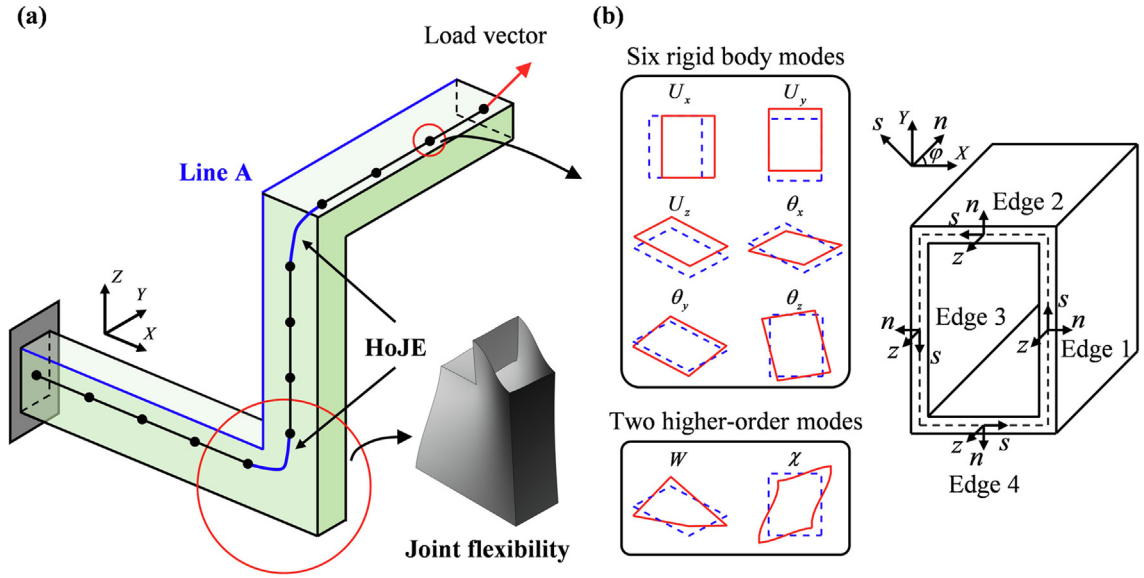

Fig. 1. (a) Illustration of modeling the 3D thin-walled beam structure with the HoJE and the joint flexibility occurring near the joint. (b) Eight DOFs used in this research with the coordinate systems.

obtain an analytic solution [27,28], or that requires a high computational expense [29–34]. Recently, DL has gained attention for enabling new modeling approaches within the finite element analysis due to its ability to predict results in real-time and its powerful performance for regression. Papadopoulos et al. [33] replaced the iterative and time-consuming process for geometric nonlinear analysis of carbon nanotubes using deep neural network (DNN); Lee et al. [35] proposed a learning criterion by comparing the DNN-based static analysis results of a truss problem according to various hyperparameters, and Liang et al. [36] predicted three-dimensional (3D) surface stress distribution by mapping it to an equivalent two-dimensional (2D) plane. Guo et al. [37] proposed a deep collocation method to solve the bending analysis of the Kirchhoff plates, and Samaniego et al. [38] applied DNNs to approximate the solution of boundary value problems. DL also has been used in various engineering problems, such as research on predicting collision load conditions from plastic deformation [39] and studies using a convolutional neural network [36,40–43].

In this study, we propose a DL-based regression model to predict a stiffness matrix for a super element, which has not been attempted before. A DL-based joint model for a two-beam joint system was considered to demonstrate our new methodology. We defined a joint as a super element consisting of nodes of the thin-walled beam elements surrounding it. First, we modeled the joint region where thin-walled beams meet with shell elements and then transformed the shell stiffness information into a stiffness matrix consistent with HoBT DOFs (see Fig. 1(b) and Fig. 2(a)). A higher-order 1D joint element (HoJE) obtained through this process has nodal DOFs of thin-walled beam sections defining the joint. To overcome the computational cost of the process mentioned above and immediately estimate the joint stiffness, we created thousands of stiffness matrix data and trained them using DL. The most intuitive method to train the DNN for the stiffness matrix of HoJE is to set each independent component value of the stiffness matrix as an output node of the DNN. However, this cannot express the stiffness information of the HoJE correctly because the correct matrix rank, or, the correct number of zero-eigenvalues associated with possible rigid-body motions, cannot be guaranteed due to the randomly perturbed machine learning error. In other words, the constructed matrix by DL is not guaranteed to represent the physics embedded in it. Note that beam elements in the 3D space have six zero-strain energy modes consisting of three translation and three rotation modes; thus, the HoJE stiffness matrix should have six zero-eigenvalues [44]. To satisfy this physical requirement or constraints, we propose to train the DNN by adding an eigendecomposition preprocess to stiffness matrix data. Specifically, the necessary physical constraints are considered by training the DNNs with the eigenvalues and eigenvectors of the stiffness matrix as output, as shown in Fig. 2(b). Additionally, the accuracy of the DNN is improved by classifying the eigenvectors according to the similarity of the eigenvectors during the training process. Through this preprocess, the proposed DL-based HoJE can predict a stiffness matrix consistent with the 1D HoBT DOFs while maintaining the accuracy of the shell-based analysis. Therefore, the proposed joint element is applicable regardless of joint angle or the size of the cross-section. We will discuss the necessity of our DL-based joint model in Appendix A.

The proposed DNN-based joint element is developed as follows. First, the derivation of the one-dimensional super element for the training data is introduced in Section 2. Section 3 addresses the DNN learning process, focusing on the training approach proposed in this research. In Section 4, various numerical examples are given using the proposed joint model, and parametric studies about the DNN hyperparameters are discussed. Finally, Section 5 presents the conclusion of this study.

2. Fundamental structural analysis methods

In this section, we define the stiffness of the HoJE from a two-beam joint structure shown in Fig. 2(a). Section 2.1 introduces a stiffness matrix of the HoBT used in this research, and Section 2.2 shows the process of deriving the stiffness matrix of the HoJE.

2.1. Higher-order one-dimensional beam analysis

1D beam analysis expresses the behavior of the target structure using nodes and line elements shown in Fig. 1(a). Each node of the HoBT, a higher-order 1D beam theory considered in this study, has three translation DOFs (U_x,U_y,U_z), three rotation DOFs (θ_x,θ_y,θ_z), and two higher-order DOFs (torsional warping W and distortion χ). We selected two cross-sectional deformation DOFs (W and χ)

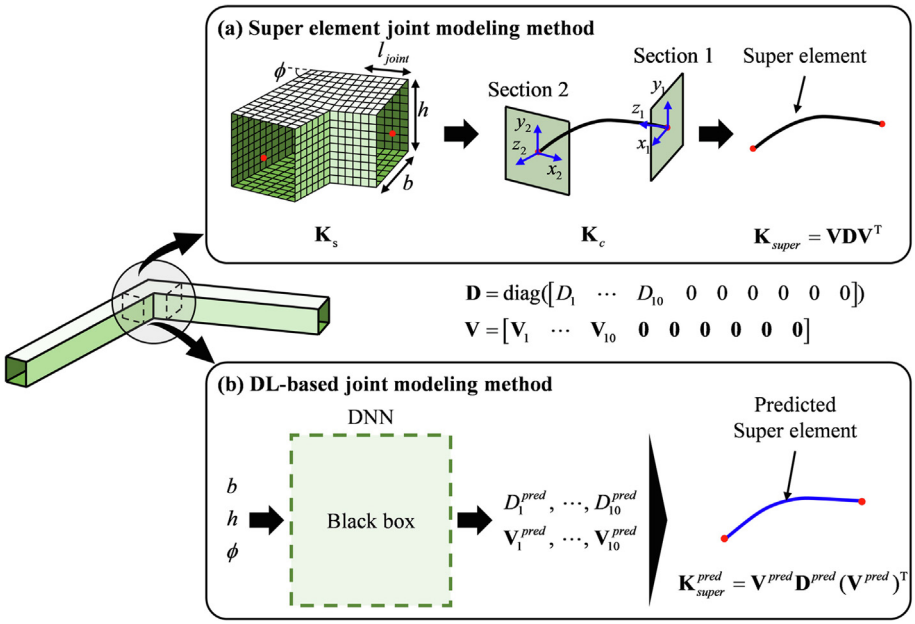

Fig. 2. Illustration of the joint modeling method using (a) super element and (b) DNN.

having a major effect on the joint stiffness of the thin-walled beam [10]. The shapes of the eight DOFs are shown in Fig. 1(b), and a shape function $\psi_\tau^\gamma(s)$ $(\tau = n,s,z;\gamma = U_x,U_y,U_z,\theta_x,\theta_y,\theta_z,W,\chi)$ corresponding to each DOF is a function of s, which is a coordinate defined along the edge of the cross-section. The exact formula of the shape functions for the eight DOFs is summarized in Appendix B. The non-zero components of the stiffness matrix of a box beam element with a beam width b, height h, thickness t, the length of a beam element l, the modulus of elasticity E, and Poisson's ratio v are defined as follows [7,10]:

$$\mathbf{K}_{e(1,1)} = -\mathbf{K}_{e(1,9)} = \mathbf{K}_{e(9,9)} = GJ_{F_x}/l \tag{1a}$$

$$\mathbf{K}_{e(1,5)} = \mathbf{K}_{e(1,13)} = -\mathbf{K}_{e(5,9)} = -\mathbf{K}_{e(9,13)} = GJ_{F_x}/2 \tag{1b}$$

$$\mathbf{K}_{e(2,2)} = -\mathbf{K}_{e(2,10)} = \mathbf{K}_{e(10,10)} = GJ_{F_y}/l \tag{1c}$$

$$\mathbf{K}_{e(2,4)} = \mathbf{K}_{e(2,12)} = -\mathbf{K}_{e(4,10)} = -\mathbf{K}_{e(10,12)} = -GJ_{F_y}/2 \tag{1d}$$

$$\mathbf{K}_{e(3,3)} = -\mathbf{K}_{e(3,11)} = \mathbf{K}_{e(11,11)} = EJ_{F_z}/l \tag{1e}$$

$$\mathbf{K}_{e(4,4)} = \mathbf{K}_{e(12,12)} = GlJ_{F_y}/3 + EJ_{M_x}/l \tag{1f}$$

$$\mathbf{K}_{e(4,12)} = GJ_{F_y}/6 - EJ_{M_x}/l \tag{1g}$$

$$\mathbf{K}_{e(5,5)} = \mathbf{K}_{e(13,13)} = GlJ_{F_x}/3 + EJ_{M_y}/l \tag{1h}$$

$$\mathbf{K}_{e(5,13)} = GlJ_{F_x}/6 - EJ_{M_y}/l \tag{1i}$$

$$\mathbf{K}_{e(6,6)} = -\mathbf{K}_{e(6,14)} = \mathbf{K}_{e(14,14)} = GJ_{M_z}/l \tag{1j}$$

$$\mathbf{K}_{e(6,7)} = \mathbf{K}_{e(6,15)} = -\mathbf{K}_{e(7,14)} = -\mathbf{K}_{e(14,15)} = -(b-h)GJ_{M_z}/2(b+h) \tag{1k}$$

$$\mathbf{K}_{e(7,7)} = (b-h)^2 GlJ_{M_z}/3(b+h)^2 + GlJ_Q/3 + E_1J_B/l \tag{1l}$$

$$\mathbf{K}_{e(7,8)} = -\mathbf{K}_{e(7,16)} = \mathbf{K}_{e(8,15)} = -\mathbf{K}_{e(15,16)} = -GJ_Q/2 \tag{1m}$$

$$\mathbf{K}_{e(8,8)} = \mathbf{K}_{e(16,16)} = E_1lC_1/3 + G(J_Q + C_2)/l \tag{1n}$$

$$\mathbf{K}_{e(8,16)} = E_1lC_1/6 - G(J_Q + C_2)/l \tag{1o}$$

$$\mathbf{K}_{e(15,15)} = (b-h)^2 GlJ_{M_z}/6(b+h)^2 + GlJ_Q/6 + E_1J_B/l \tag{1p}$$

where $C_1 = 8t^3/(b+h)$, $C_2 = 2t^3(b^2+4bh+h^2)/15(b+h)$, $E_1 = E/(1-v^2)$, $G = E/2(1+v)$ and $J_{F_x} = 2bt$, $J_{F_y} = 2ht$, $J_{F_z} = 2t(b+h)$, $J_{M_x} = h^2t(3b+h)/6$, $J_{M_y} = b^2t(b+3h)/6$, $J_{M_z} = bht(b+h)/2$, $J_B = b^2h^2t(b+h)/24$, and $J_Q = 2b^2h^2t/(b+h)$ refer to the moment of inertia of each force which is the energy conjugate of each displacement. Note that the components of $\mathbf{K}_e$ related to the U_x, U_z, θ_y DOFs and those related to U_y, θ_x, θ_z, W, χ are decoupled for the box beam [6–8].

2.2. Higher-order one-dimensional joint

This section introduces the process of defining the stiffness matrix of the HoJE, which is the target of the DL training in this study. As mentioned previously, it is crucial to express the stiffness of the thin-walled beam's joint precisely when modeling the thin-walled beam system with 1D beam elements. Modeling the joint part with 2D elements is precise but limited because the size of the shell stiffness matrix is much larger than that of the 1D beam stiffness matrix. Here, we transform the shell stiffness matrix of the joint into a super element stiffness matrix based on the HoBT DOFs to create the HoJE compressing the shell stiffness information. The HoJE is located between two nodes of the beam elements, as shown in Fig. 1(a), representing the coupling relationship among the HoBT DOFs. Creating the HoJE requires two steps: condensation and a beam node reduction process (Fig. 2(a)).

First, the condensation compresses the shell stiffness matrix $\mathbf{K}_S$ into a stiffness matrix $\mathbf{K}_c$, which only consists of global translation DOFs (U_X, U_Y, U_Z) existing on the two cross-sections that meet the beam element using the Guyan reduction [9,45].

For the two-beam joint structure in Fig. 2(a), the finite element static analysis equations are as follows:

$$\mathbf{K}_S\mathbf{d}_S = \mathbf{F}_S \tag{2}$$

$$\mathbf{K}_S = \begin{bmatrix} \mathbf{K}_{oo} & \mathbf{K}_{oi} \\ \mathbf{K}_{io} & \mathbf{K}_{ii} \end{bmatrix} \tag{3a}$$

$$\mathbf{d}_S = \begin{bmatrix} \mathbf{d}_o \\ \mathbf{d}_i \end{bmatrix} \tag{3b}$$

$$\mathbf{F}_S = \begin{bmatrix} \mathbf{F}_o \\ \mathbf{F}_i \end{bmatrix} \tag{3c}$$

$$\begin{bmatrix} \mathbf{K}_{oo} & \mathbf{K}_{oi} \\ \mathbf{K}_{io} & \mathbf{K}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{d}_o \\ \mathbf{d}_i \end{bmatrix} = \begin{bmatrix} \mathbf{F}_o \\ \mathbf{F}_i \end{bmatrix} \tag{4}$$

where S in the stiffness matrix $\mathbf{K}_S$, displacement vector $\mathbf{d}_S$, and force vector $\mathbf{F}_S$ is the notation for the shell model; the notations of the global translation DOFs in the boundary cross-sections and the rest of the DOFs are denoted by the subscript o and i, respectively. Then, we can derive the condensed stiffness matrix $\mathbf{K}_c$, displacement $\mathbf{d}_c$, and force vector $\mathbf{F}_c$ as follows:

$$\mathbf{K}_c = (\mathbf{K}_{oo} - \mathbf{K}_{oi}\mathbf{K}_{ii}^{-1}\mathbf{K}_{io}) \tag{5a}$$

$$\mathbf{d}_c = \mathbf{d}_o \tag{5b}$$

$$\mathbf{F}_c = \mathrm{F}_o - \mathbf{K}_{oi}\mathbf{K}_{ii}^{-1}\mathbf{F}_i \tag{5c}$$

The following beam node reduction converts $\mathbf{K}_c$, $\mathbf{d}_c$, and $\mathbf{F}_c$ to $\mathbf{K}_{super}$, $\mathbf{d}_b$, and $\mathbf{F}_b$ based on the HoBT DOFs. $\mathbf{K}_c$, $\mathbf{d}_c$, and $\mathbf{F}_c$ in Eq. (5) can be reconstructed using subscript *1* and subscript *2*, which are the notations of each cross-section in Fig. 2(a).

$$\mathbf{K}_c = \begin{bmatrix} \mathbf{K}_{c11} & \mathbf{K}_{c12} \\ \mathbf{K}_{c21} & \mathbf{K}_{c22} \end{bmatrix} \tag{6a}$$

$$\mathbf{d}_c = \begin{bmatrix} \mathbf{d}_{c1} \\ \mathbf{d}_{c2} \end{bmatrix} \tag{6b}$$

$$\mathbf{F}_c = \begin{bmatrix} \mathbf{F}_{c1} \\ \mathbf{F}_{c2} \end{bmatrix} \tag{6c}$$

To transform the shell-based DOFs into the HoBT DOFs, the relationship between the global translation displacements $\mathbf{d}_{c1}$ and $\mathbf{d}_{c2}$ and the corresponding HoBT displacements $\mathbf{d}_{b1}$ and $\mathbf{d}_{b2}$ should be predefined. The HoBT displacements $\mathbf{d}_{b1}$ and $\mathbf{d}_{b2}$ are composed of six rigid body displacements (U_x, U_y, U_z, θ_x, θ_y, θ_z), a warping displacement (W), and a distortion displacement (χ); the two vectors $\mathbf{d}_{b1}$, $\mathbf{d}_{b2}$ are expressed as follows:

$$\mathbf{d}_{b1} = [U_x^1\ U_y^1\ U_z^1\ \theta_x^1\ \theta_y^1\ \theta_z^1\ W^1\ \chi^1]^{\mathrm{T}} \tag{7a}$$

$$\mathbf{d}_{b2} = [U_x^2\ U_y^2\ U_z^2\ \theta_x^2\ \theta_y^2\ \theta_z^2\ W^2\ \chi^2]^{\mathrm{T}} \tag{7b}$$

The vector $\mathbf{d}_{c1(k)}$ consisting of the global translation displacements ($U^S_{x(k)}$, $U^S_{y(k)}$, $U^S_{z(k)}$) defined at cross-section *1* (at the position of s_k) can be expressed as the product of $\mathbf{d}_{b1}$, the rotation matrix $\mathbf{R}_1$, and the shape function matrix $\boldsymbol{\Psi}$; the notation k indicates the k-th node among the N number of shell nodes on each section in Fig. 2(a).

$$\mathbf{R}_1(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$\boldsymbol{\Psi}(s_k) = \begin{bmatrix} \psi_n^{U_x} & \psi_n^{U_y} & \psi_n^{U_z} & \psi_n^{\theta_x} & \psi_n^{\theta_y} & \psi_n^{\theta_z} & \psi_s^{W} & \psi_n^{\chi} \\ \psi_s^{U_x} & \psi_s^{U_y} & \psi_s^{U_z} & \psi_s^{\theta_x} & \psi_s^{\theta_y} & \psi_s^{\theta_z} & \psi_s^{W} & \psi_s^{\chi} \\ \psi_z^{U_x} & \psi_z^{U_y} & \psi_z^{U_z} & \psi_z^{\theta_x} & \psi_z^{\theta_y} & \psi_z^{\theta_z} & \psi_z^{W} & \psi_z^{\chi} \end{bmatrix}_{s_k} \tag{9}$$

$$\mathbf{d}_{c1(k)} = \begin{bmatrix} U^S_{x(k)} \\ U^S_{y(k)} \\ U^S_{z(k)} \end{bmatrix} = \mathbf{R}_1(\varphi)\boldsymbol{\Psi}(s_k)\mathbf{d}_{b1} = \mathbf{T}_1^k\mathbf{d}_{b1} \tag{10}$$

Here, φ in Eq. (8) is the angle defined in Fig. 1(b), and $\psi_\tau^\gamma(s_k)$ in Eq. (9) is described in Appendix B. The transformation matrix $\mathbf{T}_1^k$ relates $\mathbf{d}_{b1}$ to the global translation displacement vector $\mathbf{d}_{c1(k)}$ at the position of s_k. The transformation matrix $\mathbf{T}_1$, which defines the relationship between $\mathbf{d}_{c1}$ in Eq. (6b) and $\mathbf{d}_{b1}$ in Eq. (7a), can be obtained by stacking the matrix $\mathbf{T}_1^k$.

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{T}_1^1 \\ \vdots \\ \mathbf{T}_1^k \\ \vdots \\ \mathbf{T}_1^N \end{bmatrix} \tag{11}$$

The transformation matrix converting $\mathbf{d}_{b2}$ in Eq. (7b) to $\mathbf{d}_{c2}$ in Eq. (6b) is derived by stacking the matrix $\mathbf{T}_2^k$, as done for $\mathbf{T}_1$. The matrix $\mathbf{T}_2^k$ can be calculated by multiplying $\mathbf{T}_1^k$ and the matrix $\mathbf{R}_2$, representing the coordinate transformation according to the joint angle ϕ. Note that ϕ is the degree between two z-directional local coordinates of both sections.

$$\mathbf{R}_2(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \tag{12}$$

$$\mathbf{T}_2^k = \mathbf{R}_2(\phi)\mathbf{T}_1^k \tag{13}$$

$$\mathbf{T}_2 = \begin{bmatrix} \mathbf{T}_2^1 \\ \vdots \\ \mathbf{T}_2^k \\ \vdots \\ \mathbf{T}_2^N \end{bmatrix} \tag{14}$$

Using the transformation matrices $\mathbf{T}_1$ and $\mathbf{T}_2$ from Eqs. (11) and (14), the global translation displacements $\mathbf{d}_{c1}$ and $\mathbf{d}_{c2}$ can be derived.

$$\mathbf{d}_{c1} = \mathbf{T}_1\mathbf{d}_{b1} \tag{15a}$$

$$\mathbf{d}_{c2} = \mathbf{T}_2\mathbf{d}_{b2} \tag{15b}$$

The force vectors $\mathbf{F}_{b1}$ and $\mathbf{F}_{b2}$ defined at the HoBT based nodes in contact with the HoJE and $\mathbf{F}_{c1}$ and $\mathbf{F}_{c2}$ in Eq. (6c) satisfies the following equations, with the displacement vector $\mathbf{d}_{b1}$, $\mathbf{d}_{b2}$, $\mathbf{d}_{c1}$, and $\mathbf{d}_{c2}$ in Eqs. (6b) and (7) by work conservation [1].

$$\mathbf{d}_{b1}^{\mathrm{T}}\mathbf{F}_{b1} = \mathbf{d}_{c1}^{\mathrm{T}}\mathbf{F}_{c1} \tag{16a}$$

$$\mathbf{d}_{b2}^{\mathrm{T}}\mathbf{F}_{b2} = \mathbf{d}_{c2}^{\mathrm{T}}\mathbf{F}_{c2} \tag{16b}$$

$$\begin{bmatrix} \mathbf{K}_{c11} & \mathbf{K}_{c12} \\ \mathbf{K}_{c21} & \mathbf{K}_{c22} \end{bmatrix} \begin{bmatrix} \mathbf{d}_{c1} \\ \mathbf{d}_{c2} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{c1} \\ \mathbf{F}_{c2} \end{bmatrix} \tag{17}$$

Substituting Eqs. (15) and (16) into Eq. (17), the two equations of Eq. (17) are expressed as follows:

$$\mathbf{T}_1^{\mathrm{T}}\mathbf{K}_{c11}\mathbf{T}_1\mathbf{d}_{b1} + \mathbf{T}_1^{\mathrm{T}}\mathbf{K}_{c12}\mathbf{T}_2\mathbf{d}_{b2} = \mathbf{T}_1^{\mathrm{T}}\mathbf{F}_{c1} = \mathbf{F}_{b1} \tag{18a}$$

$$\mathbf{T}_2^{\mathrm{T}}\mathbf{K}_{c21}\mathbf{T}_1\mathbf{d}_{b1} + \mathbf{T}_2^{\mathrm{T}}\mathbf{K}_{c22}\mathbf{T}_2\mathbf{d}_{b2} = \mathbf{T}_2^{\mathrm{T}}\mathbf{F}_{c2} = \mathbf{F}_{b2} \tag{18b}$$

$$\mathbf{K}_{super} = \begin{bmatrix} \mathbf{T}_1^{\mathrm{T}}\mathbf{K}_{c11}\mathbf{T}_1 & \mathbf{T}_1^{\mathrm{T}}\mathbf{K}_{c12}\mathbf{T}_2 \\ \mathbf{T}_2^{\mathrm{T}}\mathbf{K}_{c21}\mathbf{T}_1 & \mathbf{T}_2^{\mathrm{T}}\mathbf{K}_{c22}\mathbf{T}_2 \end{bmatrix} \tag{19a}$$

$$\mathbf{d}_b = \begin{bmatrix} \mathbf{d}_{b1} \\ \mathbf{d}_{b2} \end{bmatrix} \tag{19b}$$

$$\mathbf{F}_b = \begin{bmatrix} \mathbf{F}_{b1} \\ \mathbf{F}_{b2} \end{bmatrix} \tag{19c}$$

$$\mathbf{K}_{super}\mathbf{d}_b = \mathbf{F}_b \tag{20}$$

Finally, we derived the stiffness matrix of the HoJE compressing the shell stiffness information in one-dimensional HoBT DOFs.

3. Deep neural network training process

Here, we discuss the training process using DL to reduce the computational cost required in creating the HoJE. The HoJE created in Section 2.2 ensures the accuracy of the shell model with a few DOFs because it considers a condensed stiffness matrix obtained by the reduction of the shell elements. However, creating the base shell model is fundamental for the process, which requires a high computational expense. Here, we aim to eliminate the process of shell modeling by implementing the DNN for the stiffness matrix of the HoJE, of which the matrix size is condensed to be predictable using DL. When predicting the stiffness matrix of the HoJE (with the parameters determining the joint configuration), the most straightforward idea is predicting each component of the matrix. However, only considering the numbers misses the physical constraints that the stiffness matrix must satisfy, especially the zero-eigenvalue condition [44]. In this study, we propose a data-driven approach predicting the stiffness matrix considering the zero-eigenvalue physical conditions. Our approach divides the stiffness matrix into eigenvalues and eigenvectors through eigendecomposition preprocessing and trains them separately. We define the parameters related to the joint's geometry and describe the process to create the training data in Section 3.1. Section 3.2 discusses the preprocess for the HoJE stiffness matrix data including the eigendecomposition. Section 3.3 introduces the DNN architecture and the post-processing used in the training.

3.1. Joint parameters and training data

The two-box beam joint is defined with four parameters (width b, height h, joint angle ϕ, and joint length l_{joint}), if the thickness t is fixed and the cross-sections of the two beams are the same as shown in Fig. 2(a).

$$l_{joint} = \frac{1}{2}\max(b, h) \tag{21}$$

It has been reported [1] that the local cross-sectional deformations, except the torsional warping and distortion DOFs considered in this research, are damped out for the joint length l_{joint} defined in Eq. (21). For this reason, we fixed the joint length with the definition in Eq. (21) and defined the geometry of the joint by three parameters ($b,\ h,\ \phi$). We used the aspect ratio $\alpha = h/b$ instead of h to consider more general types of cross-sections. Note that these parameters can be extended depending on the problems for the expansion of this work.

We have set up the design space of each parameter as shown in Table 1, considering the beam cross-sections widely used in bus frames [46]. In Table 1, the feasible region refers to the region where the trained DL will be used to replace the shell analysis, and the training region refers to the region we selected for the training data. The training region is wider than the feasible region because generally, the low accuracy of the regression model is observed near the boundary [47]. Sampling points are uniformly chosen by n_b for b, n_α for α, and n_ϕ for ϕ, respectively, within the boundary of the training region, so the total number of data used was $n_b \cdot n_a \cdot n_\phi$. We modeled the joint part with the ABAQUS [48] four-node elements for each sampling point. The mesh size of each sampling point was determined by dividing the short side of the cross-section into ten equal elements, which were found to yield converged results in an earlier box beam analysis [28]. The HoJE stiffness matrix data was collected by transforming the shell stiffness matrix using the process in Section 2.

3.2. Proposed training method of a stiffness matrix

This section explains the training process of the stiffness matrix using DL. The most intuitive method to implement the DNN predicting the stiffness matrix is building a DNN model with the joint parameters as input nodes and with independent components of the stiffness matrix as output nodes. However, even if the DNN is well trained, the output nodes must have errors, which are randomly perturbed. The randomly perturbed stiffness matrix cannot perform as the stiffness matrix because it does not consider the physical constraint of the zero-eigenvalues of the rigid body modes. The HoJE used in this study has six rigid body modes excluding two higher-order modes (warping and distortion); thus, the stiffness matrix must have six zero-eigenvalues. To consider this condition, we propose the following process. The HoJE stiffness matrix can be decomposed into the diagonal matrix $\mathbf{D}$ having eigenvalues as components and into the matrix $\mathbf{V}$ having each eigenvector as a column through an eigendecomposition process in Eq. (22).

$$\mathbf{K}_{super} = \mathbf{V}\mathbf{D}\mathbf{V}^{\mathrm{T}} \tag{22}$$

Considering the HoBT discussed in Section 2, the HoJE stiffness matrix has 16 eigenvalues and eigenvectors. Six sets of eigenvalue and eigenvector are related to rigid body modes, of which the eigenvalues should be zero, and the eigenvectors should represent global rigid motions. The eigenvectors corresponding to the zero-eigenvalues do not affect the stiffness matrix because the eigenvectors of the stiffness matrix are orthogonal to each other [49]. Therefore, we fixed the eigenvectors corresponding to the zero-eigenvalues as a zero-vector. Then, the matrices $\mathbf{D}$, which has ten nonzero-eigenvalues in the diagonal components, and $\mathbf{V}$, which has ten eigenvectors that each have a 16 by 1 size as columns, can be expressed as shown in Fig. 2, and in Eq. (23).

$$\mathbf{D} = \mathrm{diag}([D_1 \cdots D_{10}\ 0, 0, 0, 0, 0, 0]) \tag{23a}$$

$$\mathbf{V} = [\mathbf{V}_1 \cdots \mathbf{V}_{10}\ \mathbf{0}\,\mathbf{0}\,\mathbf{0}\,\mathbf{0}\,\mathbf{0}\,\mathbf{0}] \tag{23b}$$

By predicting the eigenvalues and eigenvectors in Eq. (23) using the DNN, we can derive the stiffness matrix reflecting the zero-eigenvalues. To build DNN models for ten eigenvalues and eigenvectors, a criterion for classifying ten different modes is required for each training data. The two criteria for classifying different modes can be proposed; one is the ascending order of the eigenvalues, and the other is the similarity of the shape of the eigenvector. In this study, we used the latter criterion to sort the uniformly sampled data, and the training result was better for the latter than the former. We will discuss these results in Example 5 (Section 4.2).

Table 1
Lower and upper bound of the feasible region and training region for three joint parameters (beam width, beam height, and joint angle).

Joint parameter	Feasible region		Training region	
	Lower bound	Upper bound	Lower bound	Upper bound
b	30 mm	65 mm	25 mm	70 mm
α	0.5	2.0	0.4	2.1
ϕ	10°	135°	5°	140°

Let us denote the joint parameter sets for the lower boundaries and upper boundaries in the training region in Table 1 as $(b_1,\ \alpha_1,\ \phi_1)$ and $(b_{n_b},\ \alpha_{n_\alpha},\ \phi_{n_\phi})$, respectively. The eigenvector $\mathbf{V}_\zeta^{(p,q,r)}$ is the ζ -th eigenvector for the joint parameter set $(b_p,\ \alpha_q,\ \phi_r)$, in which p, q, and r are the arbitrary integers between one and n_b, n_α, and n_ϕ, respectively. The similarities between the eigenvector $\mathbf{V}_\zeta^{(p,q,r)}$ and the pre-determined eigenvectors $\mathbf{V}_\eta^{(p-1,q,r)}$, $\mathbf{V}_\eta^{(p,q-1,r)}$, and $\mathbf{V}_\eta^{(p,q,r-1)}$ are compared with the modal assurance criterion (MAC) [50,51]. For example, the MAC value between $\mathbf{V}_\zeta^{(p,q,r)}$ and $\mathbf{V}_\eta^{(p-1,q,r)}$ is calculated as follows.

$$\mathrm{MAC}(\mathbf{V}_\zeta^{(p,q,r)},\mathbf{V}_\eta^{(p-1,q,r)}) = \frac{\left|(\mathbf{V}_\zeta^{(p,q,r)})^{\mathrm{T}}\mathbf{V}_\eta^{(p-1,q,r)}\right|^2}{\left((\mathbf{V}_\zeta^{(p,q,r)})^{\mathrm{T}}\mathbf{V}_\zeta^{(p,q,r)}\right)\left((\mathbf{V}_\eta^{(p-1,q,r)})^{\mathrm{T}}\mathbf{V}_\eta^{(p-1,q,r)}\right)} \tag{24}$$

Here, we considered the mode-tracking method to classify the eigenvectors using MAC values [52]. The mode-tracking in this research is performed in three steps shown in Fig. 3. First, ten different eigenvectors are defined on the joint parameter set $(b_1,\ \alpha_1,\ \phi_1)$. In Step 1, the eigenvectors of the stiffness matrix on the b -axis, α -axis, and ϕ -axis are classified tracking the modes along each axis. In Step 2, the eigenvectors of the data on the $b\alpha$ -plane, $\alpha\phi$ -plane, and ϕb -plane are classified along each surface. Finally, the eigenvectors for the rest of the data are classified.

For the mode-tracking details, we calculate the MAC values between the eigenvectors from the stiffness matrix of interest and nearby eigenvectors whose order is predetermined. Note that Step 1 has a single; Step 2 has two, and Step 3 has three nearby determined eigenvector sets. For example, in Step 3, to determine the 2nd mode of the eigenvector among ten eigenvectors defined on the joint parameter set $(b_p,\ \alpha_q,\ \phi_r)$, ten eigenvectors should be compared to the MAC values with three nearby predetermined eigenvectors classified as the 2nd mode (defined on the joint parameter sets $(b_{p-1},\ \alpha_q,\ \phi_r)$, $(b_p,\ \alpha_{q-1},\ \phi_r)$, and $(b_p,\ \alpha_q,\ \phi_{r-1})$).

$$\begin{aligned}\mathbf{\Lambda}_\zeta^{(p,q,r)} = {} & \mathrm{MAC}(\mathbf{V}_\zeta^{(p,q,r)},\mathbf{V}_2^{(p-1,q,r)}) + \mathrm{MAC}(\mathbf{V}_\zeta^{(p,q,r)},\mathbf{V}_2^{(p,q-1,r)}) \\ & + \mathrm{MAC}(\mathbf{V}_\zeta^{(p,q,r)},\mathbf{V}_2^{(p,q,r-1)})\end{aligned} \tag{25}$$

For $\zeta = 1,\ 2,\ \cdots,\ 10$, $\mathbf{\Lambda}_\zeta^{(p,q,r)}$ (the ζ -th component of the tracking vector $\mathbf{\Lambda}^{(p,q,r)}$) in Eq. (25) can be calculated by the sum of the MAC values with the three nearby eigenvectors. Then, we can determine the 2nd mode of the eigenvector defined on the joint parameter set $(b_p,\ \alpha_q,\ \phi_r)$ by considering the maximum component of the $\mathbf{\Lambda}^{(p,q,r)}$. During the mode tracking, we also update the eigenvalues to match the same order. Finally, the total training data is derived into eleven datasets, one for the eigenvalues and ten for the eigenvectors of each mode.

3.3. Deep neural network training & post-processing

This section introduces the architectures of the DNNs and the hyperparameters used in this study. To estimate the stiffness matrix, we predicted the eigenvalue matrix and the eigenvector matrix independently and reconstructed the stiffness matrix to remain as the zero-eigenvalue condition. The basic architecture of the DNN is a fully connected multilayer perceptron and we implemented three different types of DNNs (two regression models and one binary classification model) shown in Fig. 4(a). First, we propose the architecture for the eigenvalues using 12 hidden layers with 39 nodes for each layer. The input layer is a 13 by 1 vector, including the design variable vector $\mathbf{J}_{(p,q,r)} = [b_p,\ \alpha_q,\ \phi_r]^{\mathrm{T}}$ and the one-hot vector $\mathbf{m}_j^{value}$, which discriminates the eigenvector. For example, when the location of the one-hot vector's only nonzero component is the 3rd (j = 3), it indicates the eigenvalue of that mode. The output layer is the eigenvalue of the corresponding D_j. Note that all vectors $\mathbf{J}_{(p,q,r)}$ are normalized concerning each boundary of the training region in Table 1 for regularization. Once the DNN is trained, the predicted eigenvalue D_j^{pred} can be calculated by the DNN model shown in Fig. 4(b). The predicted eigenvalue matrix $\boldsymbol{D}^{pred}$ for the arbitrary design variables has ten nonzero diagonal components and six zero diagonal components shown in Eq. (26).

$$\mathbf{D}^{pred} = \mathrm{diag}[D_1^{pred},\ \cdots,\ D_{10}^{pred},\ 0,\ \cdots,\ 0] \tag{26}$$

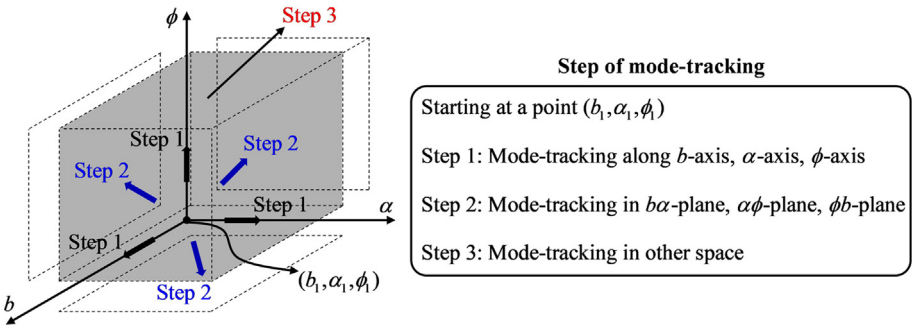

Fig. 3. Illustration of mode-tracking step.

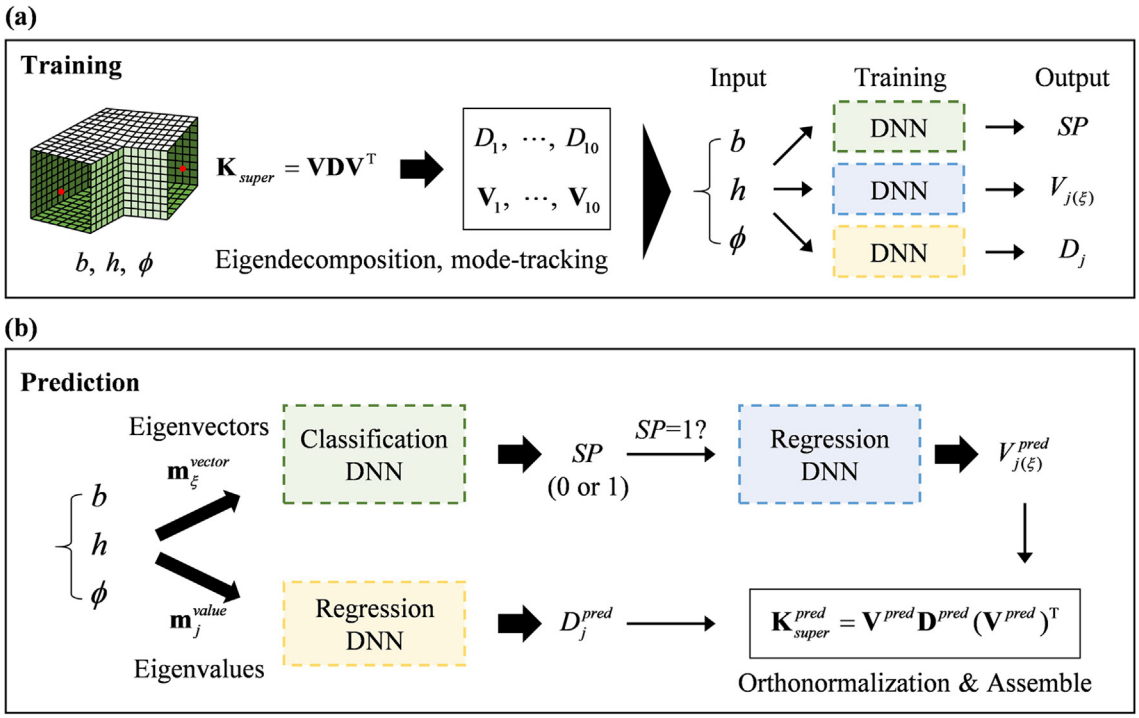

Fig. 4. (a) The description of constructing three different DNNs to predict the stiffness matrix of the super element. (b) The process of predicting the stiffness matrix for a given joint geometry using trained DNNs.

Next, we need to build up the architecture for the eigenvectors. In our box-beam joint problem, the out-of-plane DOFs (U_y, θ_x, θ_z, W, χ) and in-plane DOFs (U_x, U_z, θ_y) are physically decoupled [6–8]. Because of the decoupled characteristic, ten modes of the eigenvectors are divided into the out-of-plane and in-plane modes, and it takes almost 50% of the eigenvector components to be sparse. To reduce the training time of the eigenvectors, we first classified the nonzero components using a classification model and then developed a regression model only for the nonzero components. The classification model can be neglected for general cross-sectional beams [53], of which the DOFs are fully coupled. The effect of the classification model is discussed in Example 5 (Section 4.2).

The DNN classification model of the eigenvector is set to have three hidden layers (each layer has 13 nodes), and the regression model is set to have ten hidden layers (each layer has 57 nodes). The input layer for both of the DNN models is a 19 by 1 vector, including the design variable vector and the one-hot vector $\mathbf{m}_{\xi}^{vector}$ which discriminates 16 components of the eigenvector. The output of the classification model can be the value of 0 or 1 and this value is defined as *SP*, which classifies whether the eigenvector component is zero, shown in Fig. 4(a). The output of the regression model is $V_{j(\xi)}^{pred}$, which is the ξ -th component of the j -th mode eigenvector. We trained the classification model using all the components of the eigenvectors in the training region in Table 1 and trained the regression model only using nonzero components of the eigenvectors. The predicted j -th mode eigenvector $\mathbf{V}_j^{pred}$ for the arbitrary joint parameter set $\mathbf{J}_{(p,q,r)}$ can be derived by the procedure in Fig. 4(b). If *SP* is 0, the ξ -th component for the j -th mode eigenvector is determined to be zero. In other cases, the regression DNN model predicts the nonzero component of each of the eigenvectors. By repeating this process for $\xi = 1, 2, \cdots, 16$, and for $j = 1, 2, \cdots, 10$, the predicted j -th mode eigenvector $\mathbf{V}_j^{pred}$ can be obtained as Eq. (27).

$$\mathbf{V}_j^{pred} = \left[V_{j(1)}^{pred}, \cdots, V_{j(16)}^{pred}\right]^{\mathrm{T}} \tag{27}$$

Finally, the predicted eigenvectors should be orthonormalized to satisfy the characteristic of the eigenvectors of a stiffness matrix. The orthonormality for the predicted eigenvectors $\mathbf{V}_j^{pred}$ can be achieved by normalizing each of the predicted eigenvectors and orthogonalizing in the sets of the predicted eigenvectors. The detailed formulation is referred to in [54,55], which has been discussed with the orthonormalization of the experimental results.

$$\bar{\mathbf{V}}_j^{pred} = \frac{\mathbf{V}_j^{pred}}{\left|\mathbf{V}_j^{pred}\right|} \tag{28}$$

$$\bar{\mathbf{V}}^{pred} = [\bar{\mathbf{V}}_1^{pred}, \cdots, \bar{\mathbf{V}}_{10}^{pred}, \mathbf{0}, \cdots, \mathbf{0}] \tag{29}$$

$$\mathbf{V}^{pred} = 2\bar{\mathbf{V}}^{pred}\left(\boldsymbol{I} + (\bar{\mathbf{V}}^{pred})^{\mathrm{T}}\bar{\mathbf{V}}^{pred}\right)^{-1} \tag{30}$$

Note that the normalized eigenvector matrix $\bar{\mathbf{V}}^{pred}$ in Eq. (29) has ten normalized eigenvectors and six zero-column vectors. Using the predicted eigenvalue matrix $\mathbf{D}^{pred}$ and the orthonormalized eigenvector matrix $\mathbf{V}^{pred}$, the final predicted stiffness matrix $\mathbf{K}_{super}^{pred}$ can be expressed in Eq. (31), Figs. 2(b) and 4(b).

$$\mathbf{K}_{super}^{pred} = \mathbf{V}^{pred}\mathbf{D}^{pred}(\mathbf{V}^{pred})^{\mathrm{T}} \tag{31}$$

During the buildup of the DNN models, defining the loss function is essential. We set the cross-entropy (CE), mean absolute percentage error (MAPE), and mean absolute error (MAE) for the loss function of the classification model, the eigenvalue regression model, and the eigenvector regression model, respectively [56–58]. The MAPE that minimizes the percentage error is selected for the eigenvalue regression model because it is important to pre-

dict each of the eigenvalues with similar percent errors. On the other hand, the MAE is selected for the eigenvector regression model because predicting the large components of the eigenvector, which dominantly affects the stiffness matrix, more accurately is crucial.

Finally, note that the activation function of all the DNNs is the eLU function in Eq. (32), and the sigmoid function in Eq. (33) was used for the output layer of the classification model [59,60]. In Eqs. (32) and (33), *in* and *out* indicate the input and output of the node, respectively. The performances of the DNNs according to the activation functions are described in Appendix C. We applied the dropout technique with the parameter 0.5 to all training to reduce overfitting and used the Adam optimizer [61,62]. All DNNs in this research were trained using Tensorflow [63].

$$out = \begin{cases} in & \text{if } in > 0 \\ \exp(in) - 1 & \text{if } in \le 0 \end{cases} \tag{32}$$

$$out = \frac{1}{1 + \exp(-in)} \tag{33}$$

4. Results

Here, the validation of the DL-based HoJE by various numerical examples is presented, and the proposed training approaches in Section 3 are verified. Section 4.1 shows the static analysis results for several structures composed of thin-walled box beam joints using the proposed joint model. First, the accuracy of the DL-based joint model was verified for two-beam joint structures shown in Fig. 5(a) and more general 2D and 3D beam structures shown in Figs. 5(b) and 1(a) were also validated. All the beam structures are assumed to have a thickness of 2 mm and be made of steel whose material properties are $E = 200$ GPa and $\nu = 0.3$. The numerical analysis results using the predicted HoJE with HoBT modeling were compared with the entire shell modeling results and other beam modeling results.

In Section 4.2, we discuss the results according to the proposed training approaches. Parametric studies for the eigenvalues and eigenvectors were performed on the number of layers and nodes of the DNNs. In addition, the results were checked when we vary the training dataset. Lastly, it was shown that the proposed mode-tracking method is effective by comparing the results from the different methodologies of classifying eigenvectors.

4.1. Static analysis

4.1.1. Example 1: Two-beam joint under bending and torsion

This section presents the training results when the loss converges and the static analysis results for a single joint structure connecting two beams. The hyperparameters of DNNs are described in Section 3 and the learning rates are 0.0005 and 0.00007 for the eigenvalue and the eigenvector, respectively. We used a total of 5040 data ($n_b = 10, n_\alpha = 18, n_\phi = 28$). Table 2 shows the average error of the eigenvalue was less than 1 %, and the minimum $\min(\mathrm{MAC}(\mathbf{V}_j, \mathbf{V}_j^{pred}))$ for $j = 1, 2, \cdots, 10$ among all data in the feasible region was 0.9976. The histogram of $\min(\mathrm{MAC}(\mathbf{V}_j, \mathbf{V}_j^{pred}))$ is plotted in Fig. 6(a), and the convergence histories of the loss function are summarized in Appendix D.

To validate the predicted stiffness matrix, the static analysis results using the predicted stiffness matrix are compared with those using the label (super element stiffness matrix) of the training. The static analysis target is the two-beam joint structures shown in Fig. 5(a). Note that the end section of Beam 2 is not rigidly constrained to apply the zero resultant loads B and Q [6],

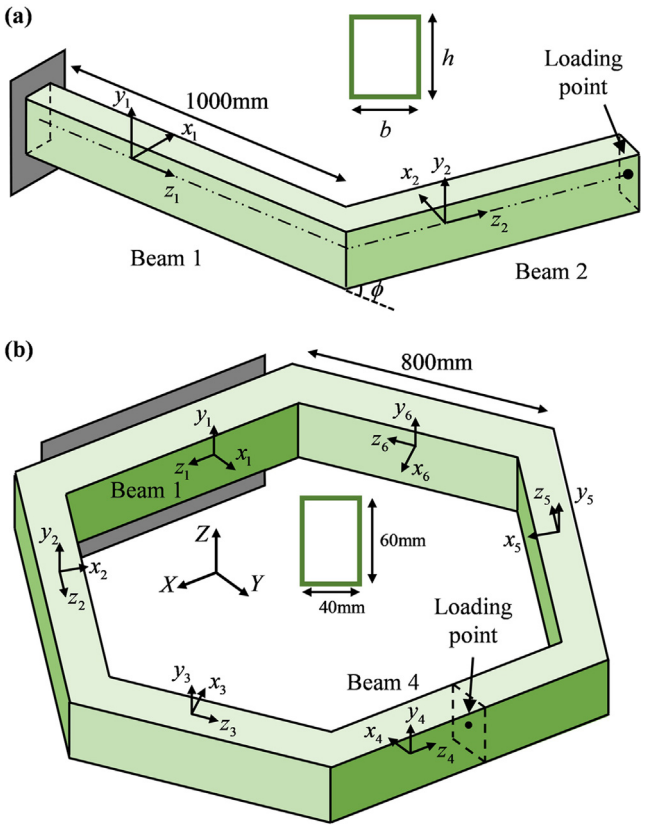

Fig. 5. The geometries of (a) a two-beam joint structure for Example 1 and (b) hexagonal loop structure for Example 2.

which are the work conjugate forces with the sectional deformations DOFs χ and W. The compliance error between the two tip displacements d^R_{super} and d^R_{pred} for the load condition R is defined as follows:

$$\text{Compliance error} = \max\left(\frac{|d^R_{super} - d^R_{pred}|}{d^R_{super}} \times 100\right) \ [\%] \quad (R = F_{x_2}, F_{y_2}, F_{z_2}, M_{x_2}, M_{y_2}, M_{z_2}, B_2, Q_2) \tag{34}$$

R is defined as the work conjugate force with eight DOFs acting on the end section of Beam 2 based on the local coordinates (x_2, y_2, z_2), and d^R_{super} is the conjugate displacement of R at the end section of Beam 2 using the label of training, while d^R_{pred} corresponds to the result from predicted stiffness matrix. We completely checked the compliance errors for all data within the feasible region. In Table 2, mostly under 2% compliance error occurred (98.7 %), and a case with an error of more than 4 % does not exist. The histogram of the compliance error in Eq. (34) is plotted in Fig. 6(b).

To compare the performance of the predicted HoJE with those of the shell element and other beam elements, we analyzed the single joint structures in Fig. 5(a) with two different geometries:

- Model A: $b = 47$ mm, $h = 48$ mm, $\phi = 57^\circ$.
- Model B: $b = 63$ mm, $h = 123$ mm, $\phi = 133^\circ$.

Note that two joints of the above models were not used to train the DNNs. In these examples, we disclosed that the end section of Beam 2 is constrained to be rigid. A tip load is applied at the end section of Beam 2, and we used 50 HoBT elements for each beam.

Table 2
Training results of the eigenvalue and eigenvector and the compliance error in Example 1.

Eigenvalue		Eigenvector		The number of compliance error under 4 % (2 %) [%]
Avg. validation loss	Avg. error [%]	Avg. validation loss	Min. MAC($\mathbf{V}_j, \mathbf{V}_j^{pred}$)*	
0.0085	0.77	2.7803e-04	0.9976	100 (98.7)

*Minimum value among all data in the feasible region.

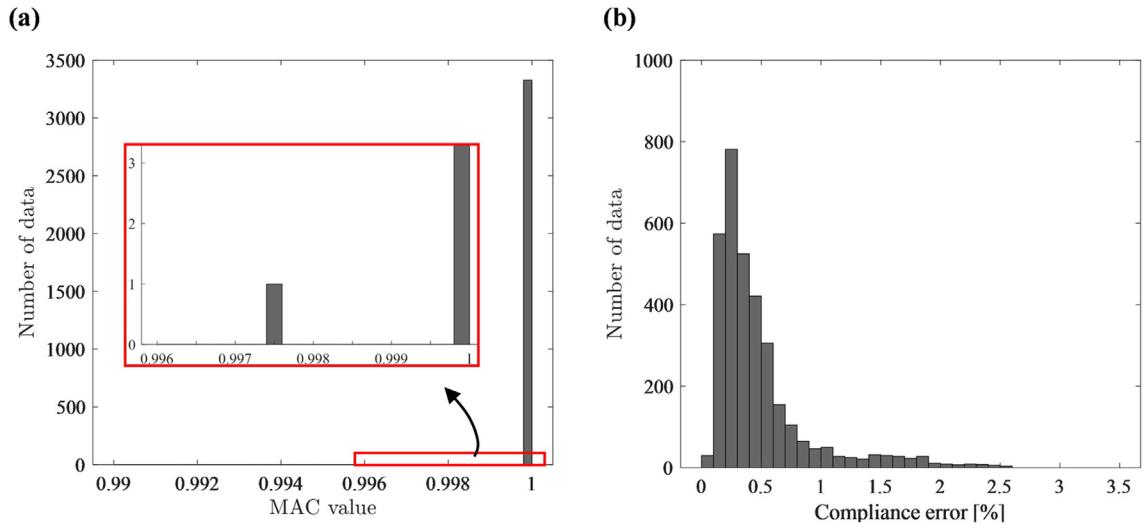

Fig. 6. (a) Histograms of min(MAC($\mathbf{V}_j, \mathbf{V}_j^{pred}$)) for $j = 1, 2, \cdots, 10$ and (b) histograms of the compliance errors in Eq. (34) (the total number of data: 3328).

As shown in Figs. 7 and 8, five out-of-plane related DOFs (U_y, θ_x, θ_z, W, χ) are plotted according to the local coordinates z_1 (Beam 1) and z_2 (Beam 2). The result of applying a y_2 directional bending force of 100 N for *Model A* is shown in Fig. 7, and the result of applying a z_2 directional torsional moment of 100 Nm for *Model B* is shown in Fig. 8.

The red lines in both figures are the results obtained with the ABAQUS shell elements. We determined the mesh sizes of the shell models for the converged result by dividing the short side of the cross-section into 15 equal elements. These shell models are discretized finer than the mesh used for training. The result of using the predicted stiffness matrix is almost identical to the result of using the stiffness matrix of the super element, which is used as the label in DL shown in Fig. 7. These results show that the predicted matrix can consider the characteristics as the stiffness matrix, and the accuracy is on par with the shell analysis result. On the other hand, the Timoshenko beam analysis result in Fig. 8 has a large error in the global behavior, because the classical theory cannot represent the sectional deformations, especially near the joint [6].

In the joint angle of 133° for *Model B*, the HoBT modeling with a single node matching [6] is less accurate than our proposed DL-based method. It shows that when the joint structure has a large joint angle, the difference between the actual joint structure and the structure expressed by sharing a single node becomes significant, and our super element-based modeling is more accurate. The results in Figs. 7 and 8 show that the proposed joint model has high fidelity similar to that of the shell model with a small number of DOFs in real-time. Furthermore, the finite element modeling with our HoJE can be easily expressed with points and lines, as shown in Fig. 1(a).

4.1.2. Example 2: Hexagonal loop structure and simple 3D structure

To verify the validity of the proposed joint model for more general 2D and 3D structures, static analyses were performed for a 2D loop structure shown in Fig. 5(b) and a 3D structure shown in Fig. 1 (a). The DNN and Timoshenko results were analyzed with 50 beam elements for each beam part same as Example 1. The 2D loop structure has a hexagonal shape, and each beam's length is 800 mm with a cross-section of $b = 40$ mm,$h = 60$ mm. Beam 1 of the loop structure is fixed, and a load is applied to the center of Beam 4. The load vector (F_X, F_Y, F_Z) is $(0\ \text{N}, 1000\ \text{N}, 1000\ \text{N})$. The graphs in Fig. 9 plot the eight HoBT DOFs along with the local coordinate system $z_1, z_2, \cdots, z_6$ in each beam of the loop structure. As shown in Fig. 9, the proposed DL-based joint model can accurately represent the joint stiffness, which the Timoshenko beam cannot capture. In the case of the 3D structure example, we analyze the structure in Fig. 1(a) which has an 800 mm length for each beam with a cross-section of $b = 30$ mm,$h = 60$ mm. One end of the 3D structure is fixed and the other end is rigidly constrained. The load vector (F_X, F_Y, F_Z) is (1000 N, 1000 N, 1000 N). The global displacements U_X,U_Y, and U_Z defined in the global coordinates X,Y, and Z along line A in Fig. 1(a) are plotted in Fig. 10. From the results in Figs. 9 and 10, we verified that our DL-based HoJE could be used for static analysis of general thin-walled joint structures.

4.2. Training results

4.2.1. Examples 3: Parametric studies for the number of layers and nodes

In Example 3, we present the proper layers and nodes for the DNNs predicting the eigenvalues and eigenvectors by comparing the training results. The same dataset with Example 1 was considered for training, and parametric studies were conducted by changing the number of layers and nodes of the DNN. First, in the case of the eigenvalue, the DNN was trained by increasing the layer number from 6 to 14 in increments of 1, and by increasing the node from 13 to 39 in increments of 13, which indicates the length of the input vector [35]. Table 3 shows the results of sum-

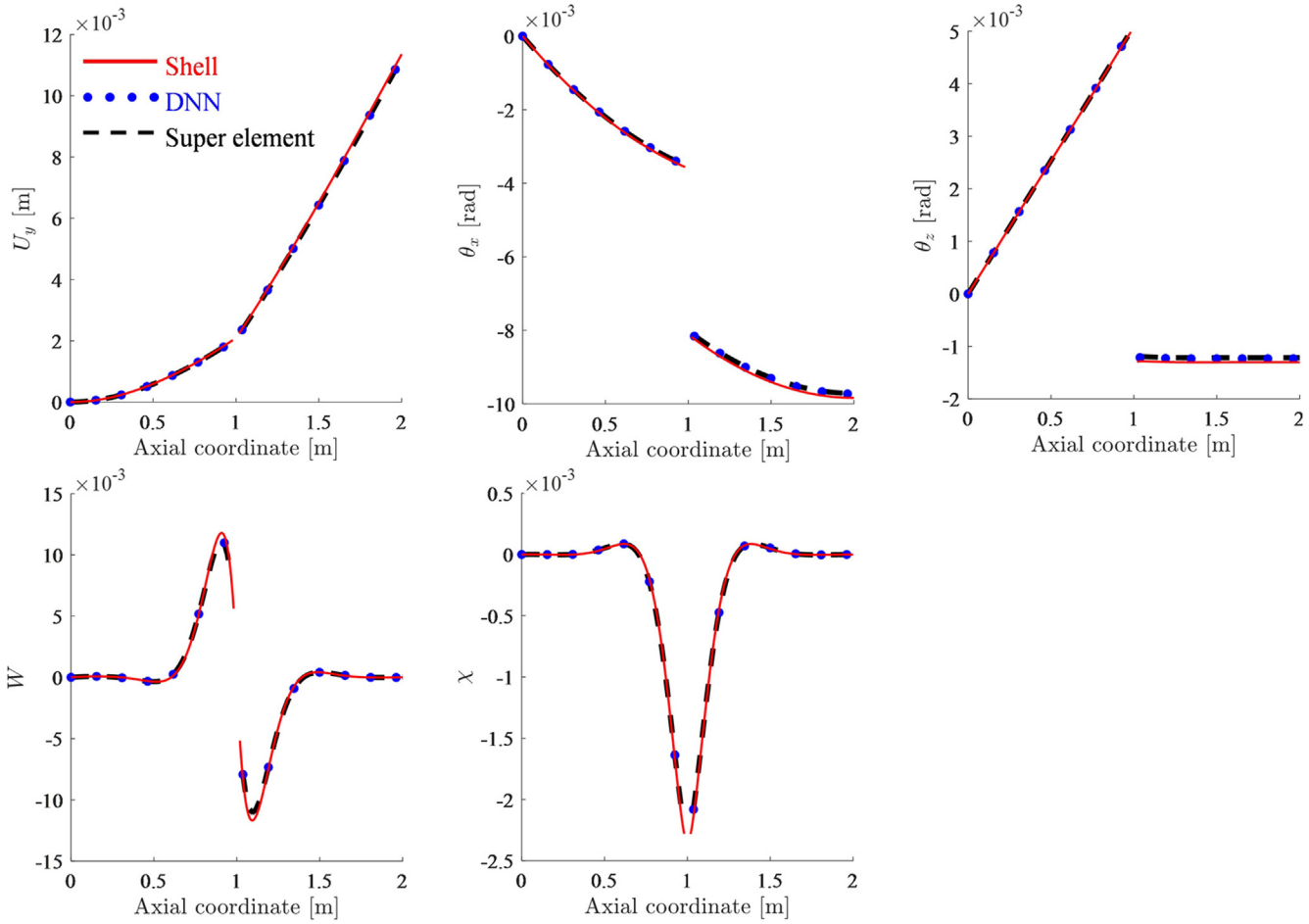

Fig. 7. The results of the five DOFs (U_y, θ_x, θ_z, W, χ) in the local coordinate systems for *Model A* of Example 1.

marizing the maximum MAPE loss and average MAPE loss for the data in the feasible region. When the number of nodes in each layer of the DNN is three times the length of the input vector, the loss becomes sufficiently small, and the maximum MAPE loss is the smallest for 12 layers. To apply the DL-based joint model to real engineering problems, the worst prediction of the joint model should be considered the representative performance rather than the average prediction. Therefore, we propose a fully connected DNN architecture for the eigenvalues with 12 layers with 39 nodes per layer, which yields the lowest maximum prediction error on the data within a given feasible region.

On the other hand, the parametric studies for the eigenvectors were performed by increasing the layer from 8 to 17 in increments of 1 and by increasing the node from 19 to 57 in increments of 19. Table 4 shows the results of the average MAE loss for the data in the feasible region. As in the results of Table 3, when the nodes of the DNN predicting the component of the eigenvector are three times that of the length of the input vector, the MAE loss becomes sufficiently small, and the loss is minimized for 10 layers. Based on these results in Table 4, we proposed a fully connected DNN with 10 layers with 57 nodes per layer for the training of the eigenvector.

4.2.2. Example 4: Results according to the sampling levels

Example 4 covers the results according to the sampling levels used for training. The DNN architecture is the same as Example 1. Based on the boundary of the training region in Table 1, *Dataset A-1* to *Dataset C-3* are defined as follows:

- *Dataset A-1*: $n_b = 6,\ n_\alpha = 18,\ n_\phi = 28$ ($n_b \cdot n_\alpha \cdot n_\phi = 3024$).
- *Dataset A-2*: $n_b = 10,\ n_\alpha = 10,\ n_\phi = 28$ ($n_b \cdot n_\alpha \cdot n_\phi = 2800$).
- *Dataset A-3*: $n_b = 10,\ n_\alpha = 18,\ n_\phi = 15$ ($n_b \cdot n_\alpha \cdot n_\phi = 2700$).
- *Dataset B*: $n_b = 10,\ n_\alpha = 18,\ n_\phi = 28$ ($n_b \cdot n_\alpha \cdot n_\phi = 5040$).
- *Dataset C-1*: $n_b = 19,\ n_\alpha = 18,\ n_\phi = 28$ ($n_b \cdot n_\alpha \cdot n_\phi = 9576$).
- *Dataset C-2*: $n_b = 10,\ n_\alpha = 35,\ n_\phi = 28$ ($n_b \cdot n_\alpha \cdot n_\phi = 9800$).
- *Dataset C-3*: $n_b = 10,\ n_\alpha = 18,\ n_\phi = 55$ ($n_b \cdot n_\alpha \cdot n_\phi = 9900$).

Dataset A-1, *A-2*, and *A-3* have around half of the data points compared to *Dataset B*, and *Dataset C-1*, *C-2*, and *C-3* have around double data points compared to *Dataset B*. To compare the training results under consistent conditions, we compared the errors of the DNNs predicted results for 3328 data points (8 levels for b, 16 levels for a, and 26 levels for ϕ) in the feasible region. Refer to the average losses of the eigenvalue and eigenvector in Table 5; the losses decreased when additional data points were used, but the improvement in the losses is minor when the losses in *Dataset B* are compared with those of *Datasets C-1*, *C-2*, and *C-3*. Based on this observation, *Dataset B* is used to obtain the main results in Section 4.

4.2.3. Example 5: Results according to the mode-tracking methods

Example 5 presents the effect of the mode-tracking preprocess for predicting the eigenvalues and eigenvectors proposed in Section 3.2. To verify the validity of the proposed mode-tracking preprocess, the results of the following three methods were compared.

- *Method A*: Classify the eigenvectors from 1st mode to 10th mode in ascending order of eigenvalues.

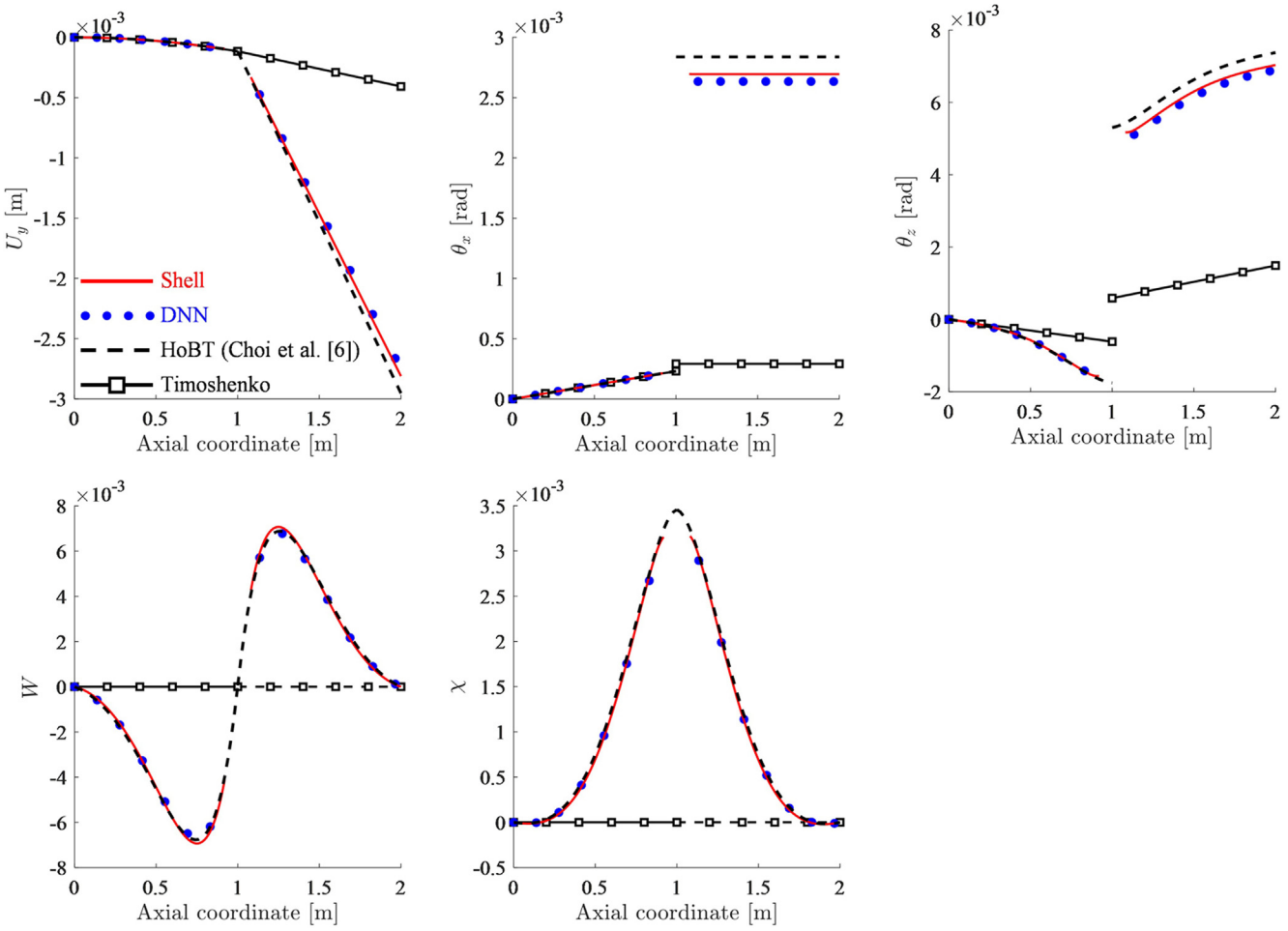

Fig. 8. The results of the five DOFs (U_y, θ_x, θ_z, W, χ) in the local coordinate systems for *Model B* of Example 1.

- *Method B*: Classify the eigenvectors according to the similarity of vectors as described in Section 3-2.
- *Method C*: Same as *Method B* except for the DNN classification model classifying nonzero-components of the eigenvector as proposed in Section 3-3.

Method A and *B* have only two types of Regression DNN models in Fig. 4(b), except the Classification DNN model. The architectures of the DNN regression models corresponding to eigenvalues and eigenvectors are the same as those used in Example 1. The dataset used in training is the same as in Example 1, and we compared the training and static analysis results for 3328 data points, which are a subset of the training data, as in Example 4. Table 6 shows the training results of the eigenvectors and the compliance errors. In *Method A*, the minimum $\min(\mathrm{MAC}(\mathbf{V}_j, \mathbf{V}_j^{pred}))$ for $j = 1, 2, \cdots, 10$ among all data points in the feasible region was close to zero, which means it was not predicting the correct eigenvector. It was happening because eigenvectors having different shapes were classified in the same mode. It is difficult to capture the moment when the shape of the eigenvector changes so that a compliance error of more than 4% exists.

As discussed in Section 3.3, nearly 50% of the components in the eigenvector have zero-values, because out-of-plane bending DOFs (U_y, θ_x, θ_z, W, χ) and in-plane bending DOFs (U_x, U_z, θ_y) are entirely decoupled [6–8]. For this reason, if only the nonzero-terms of the eigenvectors are trained after excluding the zero-terms of the eigenvectors (*Method C*), the training time is reduced by half compared to the other case (*Method B*) shown in Table 6. In detail, *Method C* took 106 minutes to train the classification model and 483 minutes to train the regression model. This result shows that the training time can be significantly reduced if the zero-terms are first classified for the other regression problems with many zero-terms. It would show a more significant effect when we need to consider a larger number of data in the future extension. For asymmetric joint structures, whose stiffness matrix is not so sparse, the classification process is not necessary; as such, the training time accordingly increases, but the performance of the DNN is maintained. To train the DNNs, a desktop computer (CPU: Intel Core i7-8700, 3.2 GHz, RAM: 32.0 GB, and GPU: Nvidia Geforce GTX 1050 Ti) was used.

5. Conclusion

This paper proposed a deep learning approach to predict the stiffness of a thin-walled beam joint using one-dimensional super elements. Because the size of the stiffness matrix of the super element is much smaller than that of a shell-based model, the joint stiffness could be predicted through deep learning. The dataset used to train the deep neural network predicting the joint stiffness was created using shell element-based results, but the resulting stiffness matrix is formed to be consistent with the degrees-of-freedom of a higher-order beam theory used to analyze the remaining part of thin-walled joint structures. A critical step in

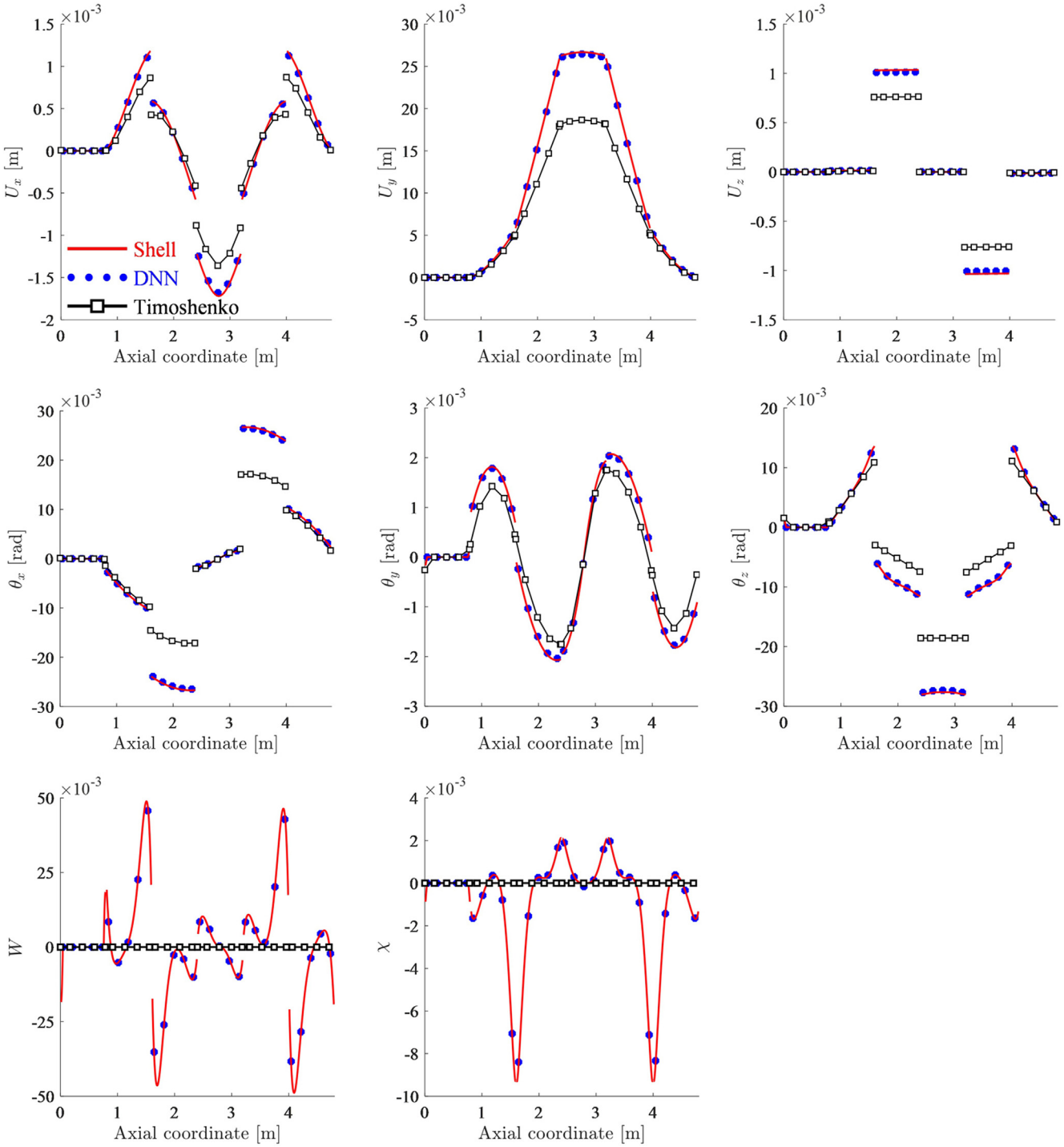

Fig. 9. The results of the eight DOFs (U_x, U_y, U_z, θ_x, θ_y, θ_z, W, χ) in the local coordinate systems for the hexagonal loop structure (Fig. 5(b)) in Example 2.

training is to make the stiffness matrix possess the correct number of zero-eigenvalues; otherwise, the constructed stiffness matrix violates the fundamental physical condition having six zero-eigenvalues. Our training strategy ensured that this condition is correctly satisfied by predicting the eigenvectors as well as eigenvalues of the stiffness matrix, excluding those related to six zero-eigenvalues, not elements of the stiffness matrix directly. Additionally, the regression of the eigenvectors using mode-tracking-based preprocessing was found to be effective. It was shown that the deep learning-based joint model constructed by the proposed approach yielded numerical results nearly as accurate as the shell-based results (mostly under 2 % and maximum under 4 % errors) for a wide range of joint geometries once the deep neural network model was constructed. We used 5040 data points to train the deep neural network having 12 (10) layers with 39 (57) nodes for the eigenvalues (eigenvectors). Because the constructed neural network can yield the joint stiffness matrix in real-time, it can significantly accelerate design optimization that requires plenty of

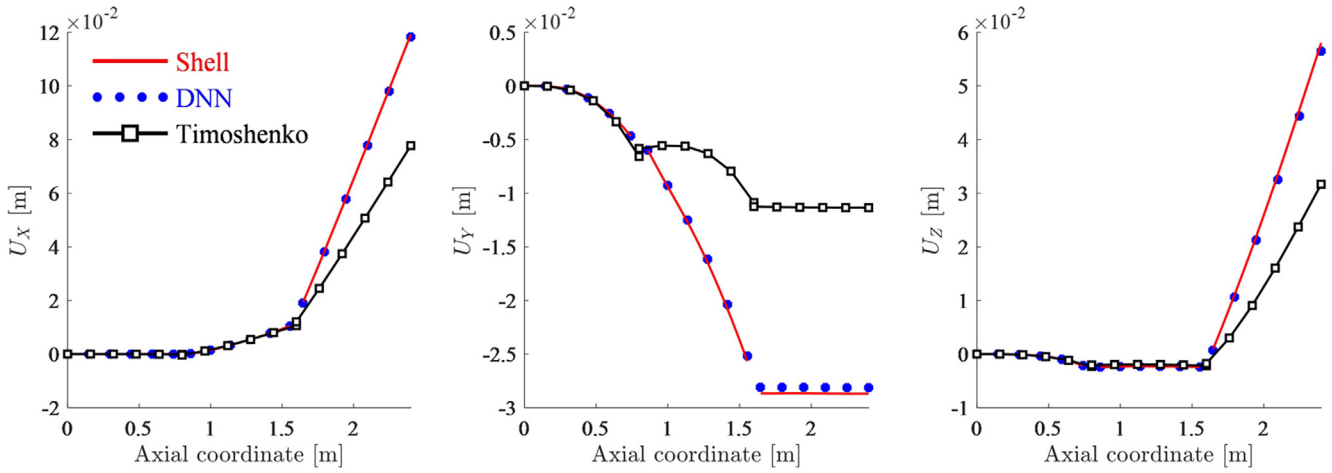

Fig. 10. The results of the global displacements (U_X, U_Y, U_Z) for the 3D structure in Example 2.

Table 3
Maximum and average MAPE loss of the eigenvalues for the data in the feasible region according to the layers and nodes of the DNN.

Layer	Maximum MAPE loss (average MAPE loss) [%]								
	6	7	8	9	10	11	12	13	14
13 node	124.09 (10.09)	48.63 (3.09)	28.51 (1.96)	24.94 (1.53)	32.35 (1.95)	13.88 (1.62)	11.86 (1.11)	14.87 (1.17)	13.61 (1.35)
26 node	17.29 (1.37)	15.79 (1.11)	13.64 (1.07)	10.10 (0.96)	8.09 (0.87)	7.18 (0.84)	6.20 (0.90)	7.73 (0.87)	5.66 (0.93)
39 node	11.90 (0.92)	10.65 (0.85)	7.00 (0.86)	7.69 (0.77)	4.84 (0.80)	7.02 (0.73)	4.74 (0.77)	4.86 (0.68)	6.00 (0.72)

Table 4
Average MAE loss of the eigenvector components for the data in the feasible region according to the layers and nodes of the DNN.

Layer	Mean absolute error (unit: 10^{-4})									
	8	9	10	11	12	13	14	15	16	17
19 node	1.907	1.660	1.693	1.696	1.493	1.641	1.645	1.590	1.426	1.697
38 node	1.514	1.475	1.374	1.328	1.283	1.376	1.416	1.433	1.328	1.442
57 node	1.233	1.196	1.150	1.267	1.153	1.161	1.369	1.319	1.319	1.326

Table 5
Training results and compliance errors according to the sampling levels.

Dataset	*A-1*	*A-2*	*A-3*	*B*	*C-1*	*C-2*	*C-3*
Number of data	3024	2800	2700	5040	9576	9800	9900
Eigenvalue average loss [MAPE]	1.11	1.05	0.94	0.77	0.74	0.77	0.73
Eigenvector average loss [MAE]	1.867 e-04	1.661 e-04	1.941 e-04	1.150 e-04	0.9624 e-04	1.022 e-04	0.9721 e-04
Average compliance error [%]	0.51	0.59	0.59	0.48	0.39	0.49	0.45

Table 6
The eigenvector accuracy and the compliance errors according to the mode tracking methods.

Method	*Method A*	*Method B*	*Method C*
Min. MAC $(\boldsymbol{V}_j, \boldsymbol{V}_j^{pred})^*$	1.519e-06	0.9976	0.9976
The number of compliance error under 4 % [%]	94.1	100	100
Training time [min.]	1160	1166	589

* Minimum value among all data in the feasible region.

changes in the joint geometries. We expect that our methodology of predicting the stiffness using neural networks will be expanded to other structural problems requiring the reduction model's stiffness estimating.

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

This research was supported by the Global Frontier R&D Program on Center for Wave Energy Control based on Metamaterials (CAMM-2014M3A6B3063711) contracted through the Institute of Advanced Machines and Design at Seoul National University funded by the Korea government.

Appendix A. Example of the DL-based joint model application

The newly proposed DL-based joint model can be applied to expedite structural optimization. For example, a bus frame to be structurally optimized is sketched in Fig. A1. As apparent in the figure, a typical frame structure consists of thin-walled box beams connected at many joints. For efficient frame structure optimization, the thin-walled beams should be modeled by higher-order 1D finite elements, and thus joints should be modeled accordingly to represent joint stiffness matrices accurately. In such cases, the newly proposed DL-based joint model becomes critical because the DL-based joint model can predict the joint stiffness matrix as accurately as that directly calculated by the shell model and allows fast calculation of joint stiffness for any geometric changes occurring during structural optimization. Although it requires significant time to build a DL-based model, its use dramatically expedites the actual structural optimization process once it is constructed.

Appendix B. Shape functions for the eight DOFs in the HoBT

This section explicitly presents the shape functions $\psi_\tau^\gamma(s)$. $\psi_\tau^\gamma(s)$ is the displacement along the τ direction at point s, when the cross-section is deformed by the displacement of γ. The index τ indicates the local coordinates n, s, z and the index γ indicates the displacement corresponding to the eight DOFs (U_x,U_y,U_z,θ_x,θ_y,θ_z,W,χ) shown in Fig. 1(b). The detailed formulas are as follows [6–8]:

$$\psi_n^{U_x}(s) = \begin{cases} 1 & (edge\ 1) \\ 0 & (edge\ 2) \\ -1 & (edge\ 3) \\ 0 & (edge\ 4) \end{cases} \tag{B.1}$$

$$\psi_s^{U_x}(s) = \begin{cases} 0 \\ -1 \\ 0 \\ 1 \end{cases} \tag{B.2}$$

$$\psi_z^{U_x}(s) = 0 \tag{B.3}$$

$$\psi_n^{U_y}(s) = \begin{cases} 0 \\ 1 \\ 0 \\ -1 \end{cases} \tag{B.4}$$

$$\psi_s^{U_y}(s) = \begin{cases} 1 \\ 0 \\ -1 \\ 0 \end{cases} \tag{B.5}$$

$$\psi_z^{U_y}(s) = 0 \tag{B.6}$$

$$\psi_n^{\theta_x}(s) = 0 \tag{B.7}$$

$$\psi_s^{\theta_x}(s) = 0 \tag{B.8}$$

$$\psi_z^{\theta_x}(s) = \begin{cases} s \\ \frac{h}{2} \\ -s \\ -\frac{h}{2} \end{cases} \tag{B.9}$$

$$\psi_n^{\theta_y}(s) = 0 \tag{B.10}$$

$$\psi_s^{\theta_y}(s) = 0 \tag{B.11}$$

$$\psi_z^{\theta_y}(s) = \begin{cases} -\frac{b}{2} \\ s \\ \frac{b}{2} \\ -s \end{cases} \tag{B.12}$$

$$\psi_n^{\theta_z}(s) = -s \tag{B.13}$$

$$\psi_s^{\theta_z}(s) = \begin{cases} \frac{b}{2} \\ \frac{h}{2} \\ \frac{b}{2} \\ \frac{h}{2} \end{cases} \tag{B.14}$$

$$\psi_z^{\theta_z}(s) = 0 \tag{B.15}$$

$$\psi_n^{W}(s) = 0 \tag{B.16}$$

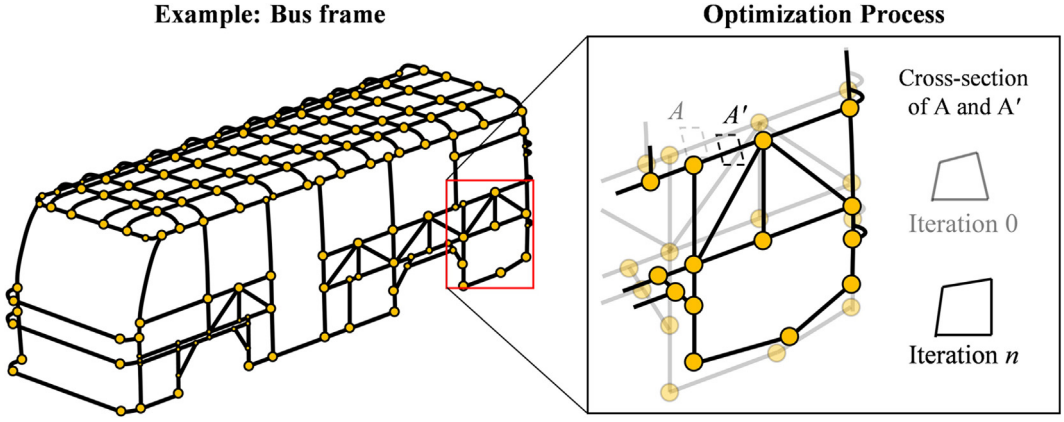

Fig. A1. Illustration of a bus frame that possibly experiences a number of structural alterations during the optimization process.

Table C1
Maximum and average MAPE loss of the eigenvalues and average MAE loss of the eigenvector components for the data in the feasible region according to the activation functions of the DNN.

Activation function	eLU	ReLU	softplus	tanh
Eigenvalue Maximum (average) loss [MAPE]	4.74 (0.77)	6.56 (0.85)	5.90 (0.72)	100 (81.43)
Eigenvector average loss [MAE]	1.150 e-04	1.305 e-04	1.565 e-04	1.396 e-04

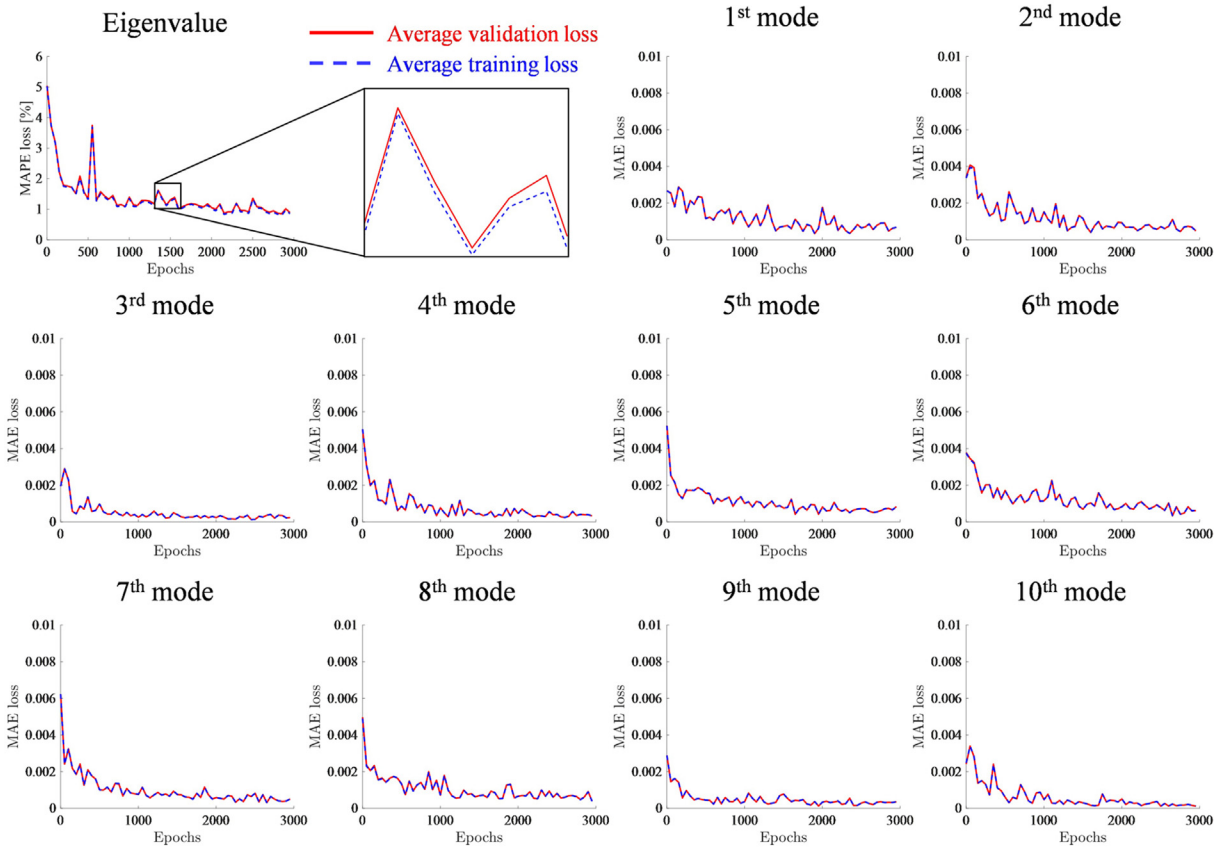

Fig. D1. The convergence history of the validation loss and training loss for the eigenvalue regression model (MAPE loss) and ten eigenvector regression models (MAE loss).

$$\psi_s^W(s) = 0 \tag{B.17}$$

$$\psi_z^W(s) = \begin{cases} \frac{b}{2}s \\ -\frac{h}{2}s \\ \frac{b}{2}s \\ -\frac{h}{2}s \end{cases} \tag{B.18}$$

$$\psi_n^\chi(s) = \begin{cases} -\frac{4}{h(b+h)}s^3 + \frac{2b+h}{b+h}s \\ \frac{4}{b(b+h)}s^3 - \frac{b+2h}{b+h}s \\ -\frac{4}{h(b+h)}s^3 + \frac{2b+h}{b+h}s \\ \frac{4}{b(b+h)}s^3 - \frac{b+2h}{b+h}s \end{cases} \tag{B.19}$$

$$\psi_s^\chi(s) = \begin{cases} \frac{bh}{b+h} \\ -\frac{bh}{b+h} \\ \frac{bh}{b+h} \\ -\frac{bh}{b+h} \end{cases} \tag{B.20}$$

$$\psi_z^\chi(s) = 0 \tag{B.21}$$

where $-\frac{h}{2} \le s \le \frac{h}{2}$ for edge 1, 3 and $-\frac{b}{2} \le s \le \frac{b}{2}$ for edge 2, 4.

Appendix C. Performances of DNNs according to the activation functions

This section compares the performances of the DNNs to predict eigenvalues and eigenvectors according to the activation functions. Other problem settings except the activation functions are the same as those for Example 1 in Section 4.1.1. Note that because the classification of the eigenvector part is not much affected by choice of an activation function, it is not considered here. The formulations using ReLU, softplus, and tanh are described in Eqs. (C.1), (C.2), and (C.3), respectively. The maximum and average MAPE loss of the eigenvalues and average MAE loss of the eigenvector components for the data points in the feasible region are summarized in Table C1. The result based on tanh showed significant errors in estimating eigenvalues, while other activation functions generally worked well. These results suggest that eLU was effective as an activation function for the problem considered in this study, but ReLU or softplus could be good candidates.

$$out = \max(0, in) \text{ for ReLU} \tag{C.1}$$

$$out = \ln(e^{in} + 1) \text{ for softplus} \tag{C.2}$$

$$out = \frac{e^{in} - e^{-in}}{e^{in} + e^{-in}} \text{ for tanh} \tag{C.3}$$

Appendix D. The loss function convergence history for training the regression model

This section discusses the convergence history of the loss function. Fig. D1 plots the convergence histories for the eigenvalue regression model and ten eigenvector regression models for every 50 epochs when the DNN is trained using the entire data points at each epoch. After the training is completed, the DNNs with the minimum average validation loss (up to 3000 epochs) are employed to predict the eigenvalues and eigenvectors. For example, Fig. D1 shows that the average MAPE validation loss of the eigenvalue regression model becomes its minimum (0.85 %) at 2900 epochs. It also shows that the mean of the minima of the average MAE validation losses of the eigenvector regression models over ten modes is $2.780{\cdot}10^{-4}$. These values are higher than the values in Table 5 because the latter is calculated for the feasible region, while the former is calculated for the entire training region. It is also noted that the average training and validation losses are close to each other, and this appears to occur because uniform sampling is used.

References

[1] Nguyen N-L, Jang G-W, Choi S, Kim J, Kim YY. Analysis of thin-walled beam-shell structures for concept modeling based on higher-order beam theory. Comput Struct 2018;195:16–33.
[2] Zuo W. An object-oriented graphics interface design and optimization software for cross-sectional shape of automobile body. Adv Eng Softw 2013;64:1–10.
[3] Zuo W, Fang J, Zhong M, Guo G. Variable cross-section rectangular beam and sensitivity analysis for lightweight design of bus frame. Int J Automot Technol 2018;19:1033–40.
[4] Dinis PB, Santana KG, Landesmann A, Camotim D. Numerical and experimental study on CFS spherically-hinged equal-leg angle columns: Stability, strength and DSM design. Thin-Walled Structures 2021;106862.
[5] Gonçalves R, Camotim D, Henriques D. GBT Analysis of Steel-Concrete Composite Beams: Recent Developments. Int J Struct Stab Dyn 2020;20:2041007.
[6] Choi S, Jang G-W, Young Kim Y. Exact matching condition at a joint of thin-walled box beams under out-of-plane bending and torsion. J Appl Mech 2012;79.
[7] Choi S, Kim YY. Exact matching at a joint of multiply-connected box beams under out-of-plane bending and torsion. Eng Struct 2016;124:96–112.
[8] Choi S, Kim YY. Analysis of two box beams-joint systems under in-plane bending and axial loads by one-dimensional higher-order beam theory. Int J Solids Struct 2016;90:69–94.
[9] Kim DM, Kim SI, Choi S, Jang GW, Kim YY. Topology optimization of thin-walled box beam structures based on the higher-order beam theory. Int J Numer Meth Eng 2016;106:576–90.
[10] Kim YY, Kim JH. Thin-walled closed box beam element for static and dynamic analysis. Int J Numer Meth Eng 1999;45:473–90.
[11] Silvestre N, Camotim D. First-order generalised beam theory for arbitrary orthotropic materials. Thin-Walled Structures 2002;40:755–89.
[12] Carrera E, Giunta G, Nali P, Petrolo M. Refined beam elements with arbitrary cross-section geometries. Comput Struct 2010;88:283–93.
[13] Yu W, Hodges DH, Ho JC. Variational asymptotic beam sectional analysis–an updated version. Int J Eng Sci 2012;59:40–64.
[14] Genoese A, Genoese A, Bilotta A, Garcea G. A mixed beam model with non-uniform warpings derived from the Saint Venànt rod. Comput Struct 2013;121:87–98.
[15] Shin D, Choi S, Jang G-W, Kim YY. Finite element beam analysis of tapered thin-walled box beams. Thin-Walled Structures 2016;102:205–14.
[16] Shin D, Choi S, Jang G-W, Kim YY. Higher-order beam theory for static and vibration analysis of composite thin-walled box beam. Compos Struct 2018;206:140–54.
[17] Schafer BW. The direct strength method of cold-formed steel member design. J Constr Steel Res 2008;64:766–78.
[18] Zuo W, Li W, Xu T, Xuan S, Na J. A complete development process of finite element software for body-in-white structure with semi-rigid beams in. NET framework. Adv Eng Softw 2012;45:261–71.
[19] Basaglia C, Camotim D. Buckling analysis of thin-walled steel structural systems using generalized beam theory (GBT). Int J Struct Stab Dyn 2015;15:1540004.
[20] Basaglia C, Camotim D, Coda HB. Generalised beam theory (GBT) formulation to analyse the vibration behaviour of thin-walled steel frames. Thin-Walled Structures 2018;127:259–74.
[21] Basaglia C, Camotim D, Silvestre N. Torsion warping transmission at thin-walled frame joints: Kinematics, modelling and structural response. J Constr Steel Res 2012;69:39–53.
[22] Donders S, Takahashi Y, Hadjit R, Van Langenhove T, Brughmans M, Van Genechten B, et al. A reduced beam and joint concept modeling approach to optimize global vehicle body dynamics. Finite Elem Anal Des 2009;45:439–55.
[23] De Gaetano G, Mundo D, Cosco F, Maletta C, Donders S. Concept modelling of vehicle joints and beam-like structures through dynamic FE-based methods. Shock Vib 2014;2014.
[24] Carrera E, Zappino E. Carrera unified formulation for free-vibration analysis of aircraft structures. AIAA Journal 2016;54:280–92.
[25] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521:436–44.
[26] Schmidhuber J. Deep learning in neural networks: An overview. Neural Netw 2015;61:85–117.
[27] Kirchdoerfer T, Ortiz M. Data-driven computational mechanics. Comput Meth Appl Mech Eng 2016;304:81–101.
[28] Shin D, Kim YY. Data-driven approach for a one-dimensional thin-walled beam analysis. Comput Struct 2020;231:106207.
[29] Yang C, Kim Y, Ryu S, Gu GX. Prediction of composite microstructure stress-strain curves using convolutional neural networks. Mater Des 2020;189:108509.
[30] Bessa M, Bostanabad R, Liu Z, Hu A, Apley DW, Brinson C, et al. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. Comput Methods Appl Mech Eng 2017;320:633–67.
[31] Bessa M, Pellegrino S. Design of ultra-thin shell structures in the stochastic post-buckling range using Bayesian machine learning and optimization. Int J Solids Struct 2018;139:174–88.
[32] Bessa MA, Glowacki P, Houlder M. Bayesian Machine Learning in Metamaterial Design: Fragile Becomes Supercompressible. Adv Mater 2019;31:1904845.
[33] Papadopoulos V, Soimiris G, Giovanis D, Papadrakakis M. A neural network-based surrogate model for carbon nanotubes with geometric nonlinearities. Comput Methods Appl Mech Eng 2018;328:411–30.
[34] Yu Y, Hur T, Jung J, Jang IG. Deep learning for determining a near-optimal topological design without any iteration. Struct Multidiscip Optim 2019;59:787–99.
[35] Lee S, Ha J, Zokhirova M, Moon H, Lee J. Background information of deep learning for structural engineering. Arch Comput Methods Eng 2018;25:121–9.
[36] Liang L, Liu M, Martin C, Sun W. A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. J R Soc Interface 2018;15:20170844.
[37] Guo H, Zhuang X, Rabczuk T. A Deep Collocation Method for the Bending Analysis of Kirchhoff Plate. Comput Mater Continua 2019;59:433–56.
[38] Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh VM, Guo H, Hamdia K, et al. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. Comput Methods Appl Mech Eng 2020;362:112790.
[39] Chen G, Li T, Chen Q, Ren S, Wang C, Li S. Application of deep learning neural network to identify collision load conditions based on permanent plastic deformation of shell structures. Comput Mech 2019;64:435–49.
[40] Cha YJ, Choi W, Büyüköztürk O. Deep learning-based crack damage detection using convolutional neural networks. Comput-Aided Civ Infrastruct Eng 2017;32:361–78.
[41] Cecen A, Dai H, Yabansu YC, Kalidindi SR, Song L. Material structure-property linkages using three-dimensional convolutional neural networks. Acta Mater 2018;146:76–84.
[42] Bhatnagar S, Afshar Y, Pan S, Duraisamy K, Kaushik S. Prediction of aerodynamic flow fields using convolutional neural networks. Comput Mech 2019;64:525–45.
[43] Tan RK, Zhang NL, Ye W. A deep learning–based method for the design of microstructural materials. Struct Multidiscip Optim 2019:1–22.
[44] Friberg O. A method for selecting deformation modes in flexible multibody dynamics. Int J Numer Meth Eng 1991;32:1637–55.
[45] Guyan RJ. Reduction of stiffness and mass matrices. AIAA J 1965;3:380.
[46] Bai J, Meng G, Zuo W. Rollover crashworthiness analysis and optimization of bus frame for conceptual design. J Mech Sci Technol 2019;33:3363–73.
[47] Fornberg B, Driscoll TA, Wright G, Charles R. Observations on the behavior of radial basis function approximations near boundaries. Comput Math Appl 2002;43:473–90.
[48] Simulia DS. ABAQUS 6.13 User's manual. Dassault Systems, Providence, RI 2013;305:306.
[49] Strang G, Strang G, Strang G, Strang G. Introduction to linear algebra. Wellesley, MA: Wellesley-Cambridge Press; 1993.
[50] Ewins DJ. Modal testing: theory and practice, vol. 15. Letchworth: Research studies press; 1984.

[51] Allemang RJ. The modal assurance criterion–twenty years of use and abuse. Sound Vibr 2003;37:14–23.
[52] Kim TS, Kim YY. Mac-based mode-tracking in structural topology optimization. Comput Struct 2000;74:375–83.
[53] Kim J, Choi S, Kim YY, Jang G-W. Hierarchical derivation of orthogonal cross-section modes for thin-walled beams with arbitrary sections. Thin-Walled Structures 2021;161:107491.
[54] Targoff W. Orthogonality check and correction of measured modes. AIAA J 1976;14:164–7.
[55] Salane H, Baldwin Jr J. Identification of modal properties of bridges. J Struct Eng 1990;116:2008–21.
[56] Willmott CJ, Matsuura K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Clim Res 2005;30:79–82.
[57] Murphy KP. Machine learning: a probabilistic perspective. MIT press; 2012.
[58] De Myttenaere A, Golden B, Le Grand B, Rossi F. Mean absolute percentage error for regression models. Neurocomputing 2016;192:38–48.
[59] Clevert D-A, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:151107289; 2015.
[60] Mitchell TM. Machine learning. McGraw-hill New York; 1997.
[61] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15:1929–58.
[62] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980; 2014.
[63] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:160304467; 2016.