mathematics

MDPI

Article

Deep Neural Networks for Form-Finding of Tensegrity Structures

Seunghye Lee [1], Qui X. Lieu [2,3], Thuc P. Vo [4] and Jaehong Lee [1,*]

[1] Deep Learning Architecture Research Center, Department of Architectural Engineering, Sejong University, 209, Neungdong-ro, Gwangjin-gu, Seoul 05006, Korea; lsh2002437@gmail.com
[2] Faculty of Civil Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City 700000, Vietnam; lieuxuanqui@hcmut.edu.vn
[3] Vietnam National University Ho Chi Minh City (VNU-HCM), Linh Trung Ward, Thu Duc District, Ho Chi Minh City 700000, Vietnam
[4] School of Computing, Engineering and Mathematical Sciences, La Trobe University, Bundoora, VIC 3086, Australia; t.vo@latrobe.edu.au
* Correspondence: jhlee@sejong.ac.kr; Tel.: +82-2-3408-3287; Fax: +82-2-3408-4331

Abstract: Analytical paradigms have limited conventional form-finding methods of tensegrities; therefore, an innovative approach is urgently needed. This paper proposes a new form-finding method based on state-of-the-art deep learning techniques. One of the statical paradigms, a force density method, is substituted for trained deep neural networks to obtain necessary information of tensegrities. It is based on the differential evolution algorithm, where the eigenvalue decomposition process of the force density matrix and the process of the equilibrium matrix are not needed to find the feasible sets of nodal coordinates. Three well-known tensegrity examples including a 2D two-strut, a 3D-truncated tetrahedron and an icosahedron tensegrity are presented for numerical verifications. The cases of the ReLU and Leaky ReLU activation functions show better results than those of the ELU and SELU. Moreover, the results of the proposed method are in good agreement with the analytical super-stable lines. Three examples show that the proposed method exhibits more uniform final shapes of tensegrity, and much faster convergence history than those of the conventional one.

Keywords: tensegrity; form-finding; differential evolution; deep neural network; force density method

MSC: 74P10

Citation: Lee, S.; Lieu, Q.X.; Vo, T.P.; Lee, J. Deep Neural Networks for Form-Finding of Tensegrity Structures. *Mathematics* **2022**, *10*, 1822. https://doi.org/10.3390/math10111822

Academic Editor: Chao Huang

Received: 25 April 2022
Accepted: 19 May 2022
Published: 25 May 2022

1. Introduction

A tensegrity consists of a discontinuous set of compressive elements inside a continuous set of tensile elements to stabilize the entire structure [1]. Many structural engineers have been fascinated by tensegrity systems ever since the new structure was introduced and patented in the USA [2], because their tension elements in the form of cables provide the tensegrity a lightweight appearance. The tensegrities are categorized as self-supporting structures which can keep their self-equilibrium positions without any costly anchorages [3]. They have many benefits, such as efficiency, deployability, redundant and scalability [4].

The design of tensegrity structures requires finding their self-stressed equilibrium configuration, which is known as the form-finding method [5]. This method can be classified into two broad families, namely the kinematical and the statical approach [6]. For the kinematical approach, the lengths of the cables are kept constant, while the strut lengths are increased until a maximum is reached. Moreover, analytical solutions, the force density method (FDM) and the energy method as well as reduced coordinate methods are used to set up a relationship between equilibrium configurations of a structure and the forces in its members. It should be mentioned that in the typical FDM, the eigenvalue decomposition (EVD) process of the force density matrix and the singular value decomposition (SVD) process of the equilibrium matrix are needed to find the feasible sets of nodal coordinates and force densities.

Figure 1 shows results of the total number of papers on the Web of Science homepage using two topics, 'Tensegrity' and 'Tensegrity + Form-finding', from 2001 to 2020. While the total number of articles and papers about the tensegrity have steadily increased for two decades, the results of the form-finding for tensegrity systems show that there are limitations to the methods. The primary reasons for the limitations are the lack of tensegrity examples and the limit of analytical paradigms. In particular, in the statical form-finding processes, optimization methods are needed to support the analytical methods and to find the equilibrium configurations and forces of elements. Meta-heuristic algorithms have been used to solve the form-finding problems of tensegrities. However, because the analytical methods are still limited, new techniques are needed to substitute for the conventional ones.

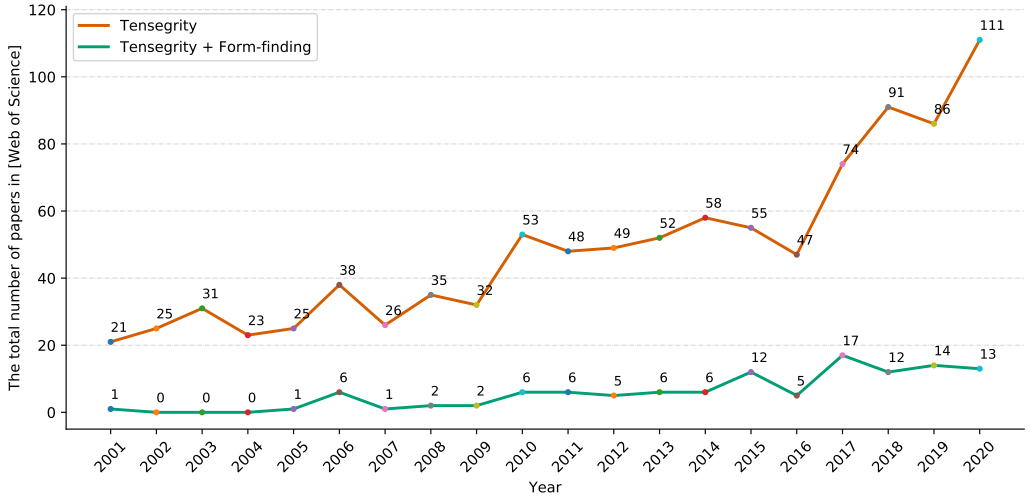


Figure 1. The total number of articles and papers on the Web of Science homepage. Results using subjects, 'Tensegrity' and 'Tensegrity + Form-finding'.

Machine learning (ML) has recently been applied to various engineering fields [7–9]. It can be an alternative approach which helps obtain a solution faster than conventional form-finding methods. Several ML attempts have been made to design a tensegrity structure, and most papers are based on an artificial neural network (ANN) with simple and traditional techniques.The authors in [10] combined the dynamic relaxation method with a NN to improve the accuracy of the form-finding method. In [11], the authors applied an ANN to the FDM for the form-finding process of tensegrity structures. The authors in [12] used ML including feature extraction and regression of form finding for a tensegrity structure. Although the ANNs have been improved by innovative techniques such as deep learning, including a deep belief network (DBN) [13], the rectified linear unit (ReLU) activation function [14] and a dropout algorithm for overfitting problems [15], only a few studies have attempted to apply them to the form-finding process. One research project team applied deep learning methods to a tensegrity robot locomotion topic [16–18]. However, the studies are restricted to the technologies for tensegrity robots despite the papers' aim to fill the gap between computer science and structural engineering and employ a deep neural network (DNN) for tensegrity systems.

In this paper, a new form-finding method for tensegrity systems using a DNN is proposed. Two complicated computational procedures in the typical FDM, which are EVD and SVD, can be eliminated in the whole form-finding process. A differential evolution (DE) algorithm is used as an analytical method of the statical form-finding paradigm. The DE algorithm is one of the meta-heuristic and population-based optimization algorithms [19]. It produces the force densities of the elements of tensegrities and trains the DNN. It then predicts coordinates that correspond to the sets of force densities. To accelerate the bulk-processing of data for a DNN, a Graphics Processing Unit (GPU) is used. Three tensegrity

examples are considered to build a dataset, and then the performance of the proposed DNN-based form-finding model is investigated.

2. Form-Finding Process Using the Force Density Method (FDM)

In this section, the procedure of a conventional FDM is briefly described. More details can be found in previous studies [20,21]. A two-dimensional (2D) two-strut tensegrity in which cables and struts are denoted by the thin and thick lines in Figure 2 is used to explain the FDM. It is composed of six elements and four nodes.

2.1. Connectivity and Force Density Matrices

This method uses a linear equation in the nodal coordinates (Equation (1)), which is known as force density [22]:

$$q_k = \frac{f_k}{l_k} \tag{1}$$

where any member k with corresponding force f_k and length l_k $(k = 1, 2, 3, \cdots, b)$.

Total members (b) and free nodes (n) of a tensegrity structure can be expressed via a connectivity matrix $\mathbf{C}$ $(\in \mathbb{R}^{b\times n})$ [23]. If member k connects nodes i and j $(i < j)$, then the ith and jth elements of the kth row of this matrix are set to 1 and -1, as follows:

$$\mathbf{C}_{(k,p)} = \begin{cases} 1 & \text{for } p = i \\ -1 & \text{for } p = j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

For a 2D two-strut tensegrity structure, connectivity matrix $\mathbf{C}$ $(\in \mathbb{R}^{6\times 4})$ is provided in Table 1. A force density matrix, $\mathbf{D}$ $(\in \mathbb{R}^{n\times n})$, is needed to define an equilibrium equation [24].

$$\mathbf{D} = \mathbf{C}^T diag(\mathbf{q})\mathbf{C} \tag{3}$$

If the initial force density is $\mathbf{q} = \{1, 1, 1, 1, -1, -1\}^T$, the force density matrix can be calculated by

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{4}$$

Finally, the force density matrix in Equation (4) can be obtained as:

$$\mathbf{D} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \tag{5}$$

Consider a Cartesian coordinate system (O-xyz), $\mathbf{x}, \mathbf{y}, \mathbf{z}$ $(\in \mathbb{R}^n)$ are denoted as the nodal coordinate vectors of the free node, in x-, y- and z-directions. For a 3D tensegrity system, when the external load and self-weight are ignored, an equilibrium equation is defined such that

$$\begin{bmatrix} \mathbf{D} & 0 & 0 \\ 0 & \mathbf{D} & 0 \\ 0 & 0 & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

Table 1. The incidence matrix of the 2D two-strut tensegrity structure.

Elements	Nodes			
	1	**2**	**3**	**4**
1	1	−1	0	0
2	0	1	−1	0
3	0	0	1	−1
4	1	0	0	−1
5	1	0	−1	0
6	0	1	0	−1

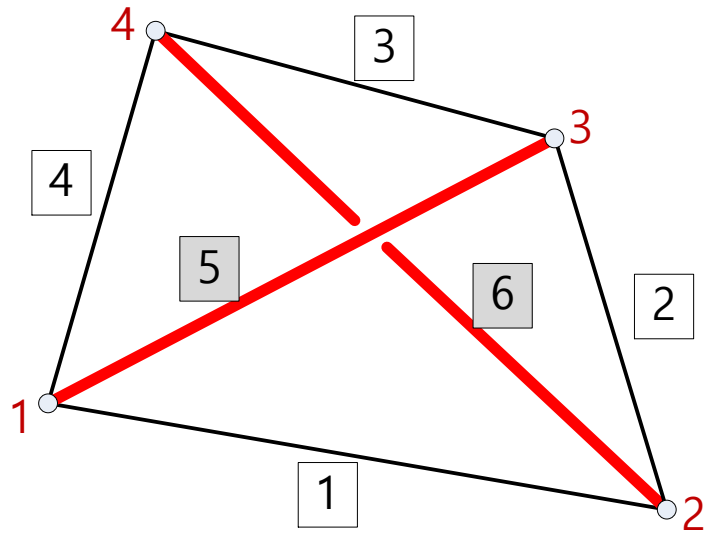


Figure 2. A 2D two-strut tensegrity structure in which cables and struts are denoted by the thin and thick lines.

The force density matrix, **D**, is then square and symmetric. In linear algebra, a sufficient condition for a symmetric matrix to be invertible is that the matrix is positive definite. The positive definite means that the scalar $\mathbf{a}^T\mathbf{D}\mathbf{a}$ is strictly positive for every non-zero column vector **a**. The nodal coordinate vectors, **x**, **y** and **z**, of the free node can be then obtained. However, because of the existence of struts ($q_k < 0$) in the tensegrity structure, the force density matrix of the tensegrity structure is positive semi-definite. The sum of the elements of a row or a column of the force density matrix, **D**, is always equal to zero for a free-standing tensegrity without fixed nodes. It is obvious that a vector $\bar{\mathbf{I}} = \{1, 1, \cdots, 1\}^T$ ($\in \mathbb{R}^{n\times 1}$) is a solution of Equation (6).

When the rank of a matrix is smaller than its dimensions, the matrix is called rank-deficient or singular. Here, the rank of the force density matrix of the two-strut tensegrity (Equation (5)), $rank(\mathbf{D})$, equals one. The rank deficiency, h_D, of the force density matrix is defined as

$$h_D = n - rank(\mathbf{D}) \tag{7}$$

Because the dimensions of the **D** matrix (Equation (5)) equals four, the rank deficiency of the matrix is then three. Only full rank matrices have an inverse [25]. The force density matrices of tensegrity structures have at least d particular solutions except for the above vector $\bar{\mathbf{I}}$. Hence, the minimum rank deficiency of the **D** matrix must be $(d + 1)$. Because the force density matrix is positive semi-definite, the **D** matrix can be factorized as follows by an eigenvalue decomposition (EVD) [26]:

$$\mathbf{D} \quad = \quad \Phi\Lambda\Phi^T \tag{8}$$

where Φ $(\in \mathbb{R}^{n\times n})$ is the orthogonal matrix ($\Phi\Phi^T = \mathbf{I}_n$, in which $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the unit matrix) whose ith column is the eigenvector basis ϕ_i $(\in \mathbb{R}^n)$ of $\mathbf{D}$. The notation Λ $(\in \mathbb{R}^{n\times n})$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, i.e., $\Lambda_{ii} = \lambda_i$. The eigenvector ϕ_i of Φ corresponds to eigenvalue λ_i of Λ. The eigenvalues are in increasing order as

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \tag{9}$$

For a 2D two-strut tensegrity structure, the diagonal matrix, Λ, is obtained as follows:

$$\Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \tag{10}$$

where the diagonal elements are the corresponding eigenvalues, namely $\{0, 0, 0, 4\}$. In the Φ matrix of Equation (8), the first $(d+1)$ columns of Φ are directly taken as potential nodal coordinates as follows:

$$[\mathbf{x}\ \mathbf{y}\ \mathbf{z}] \in \Phi = [\phi_1\ \ \phi_2\ \ \cdots\ \ \phi_{d+1}] \tag{11}$$

For a 2D structure ($d = 2$), the first three columns are chosen as the candidates.

2.2. Selection of Nodal Coordinates

If the number of zero and negative eigenvalues of the force density matrix is greater or equal to the minimum rank deficiency of the **D** matrix, minimal length and non-zero length conditions can be used to select nodal coordinates from the candidates. For 3D system, the total squared length of the entire structure can be calculated by

$$\sum_{p=1}^{b} l_p^2 = \|\mathbf{C}\phi_1 + \mathbf{C}\phi_2 + \mathbf{C}\phi_3\|^2 \tag{12}$$

except for

$$\mathbf{C}\phi_i = 0\ (i = 1, 2, \cdots, d)\ \text{ or }\ l_p = 0\ (p = 1, 2, \cdots, b) \tag{13}$$

2.3. New Set of Force Densities from Equilibrium Matrix

The force density matrix **D** can be rewritten using a connectivity matrix **C** as discussed in [23]. By substituting the rewritten force density matrix from Equation (3), the second equilibrium equation can be obtained as follows.

$$\mathbf{A}\mathbf{q} \quad = \quad \mathbf{0} \tag{14}$$

where $\mathbf{A}$ $(\in \mathbb{R}^{dn\times b})$ is known as the equilibrium matrix, defined by

$$\mathbf{A} \quad = \quad \begin{pmatrix} \mathbf{C}^T diag(\mathbf{C}\mathbf{x}) \\ \mathbf{C}^T diag(\mathbf{C}\mathbf{y}) \\ \mathbf{C}^T diag(\mathbf{C}\mathbf{z}) \end{pmatrix} \tag{15}$$

For the 2D two-strut tensegrity structure, the equilibrium matrix **A** is given in Table 2.

Equation (3) shows the relationship between the force density matrix **D** and nodal coordinates, and Equation (14) illustrates the relationship between the equilibrium matrix **A** and force densities. Here the EVD process is needed to obtain the set of force densities. However, because the equilibrium matrix **A** is not square, another decomposition method is required. A singular value decomposition (SVD) makes it possible to solve the problem [22]. An SVD is a factorization of a real or complex matrix that generalizes the eigen decomposition of a square or any rectangular matrices.

Table 2. The equilibrium matrix of the 2D two-strut tensegrity structure.

$n \times d$	Total Number of Elements (b)					
	1	**2**	**3**	**4**	**5**	**6**
1	$x_1 - x_2$	0	0	$x_1 - x_4$	$x_1 - x_3$	0
2	$y_1 - y_2$	0	0	$y_1 - y_4$	$y_1 - y_3$	0
3	$-(x_1 - x_2)$	$x_2 - x_3$	0	0	0	$x_2 - x_4$
4	$-(y_1 - y_2)$	$y_2 - y_3$	0	0	0	$y_2 - y_4$
5	0	$-(x_2 - x_3)$	$x_3 - x_4$	0	$-(x_1 - x_3)$	0
6	0	$-(y_2 - y_3)$	$y_3 - y_4$	0	$-(y_1 - y_3)$	0
7	0	0	$-(x_3 - x_4)$	$-(x_1 - x_4)$	0	$-(x_2 - x_4)$
8	0	0	$-(y_3 - y_4)$	$-(y_1 - y_4)$	0	$-(y_2 - y_4)$

The set of all solutions to the homogeneous system of Equation (14) lies in the null space of the equilibrium matrix **A**. The rank deficiency, h_A, of the equilibrium matrix is computed by

$$h_A = b - rank(\mathbf{A}) \tag{16}$$

For a 2D two-strut tensegrity system, the value of the rank deficiency is $h_A = 1$ which indicates that the tensegrity has one state of self-stress. It should mentioned that this study is limited to tensegrity structures with a single state of self-stress, which ensures the existence of one state of self-stress. To obtain the set of force densities from Equation (14), the SVD is carried out as follows:

$$\mathbf{A} \quad = \quad \mathbf{U}\mathbf{V}\mathbf{W}^T \tag{17}$$

where $\mathbf{U}$ $(\in \mathbb{R}^{dn \times dn}) = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_{dn}]$ and $\mathbf{W}$ $(\in \mathbb{R}^{b \times b}) = [\mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_b]$ are orthogonal matrices. $\mathbf{V}$ $(\in \mathbb{R}^{dn \times b})$ is a diagonal matrix with non-negative singular values of $\mathbf{A}$ in decreasing order as

$$\sigma_1 \ \geq \ \sigma_2 \ \geq \ \cdots \ \geq \ \sigma_b \ \geq \ 0 \tag{18}$$

The matrix $\mathbf{W}$ from Equation (17) can be expressed as

$$\mathbf{W} = [\mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_{rank(A)}\ \mid \mathbf{q}_1] \tag{19}$$

A graphical illustration of the SVD of $\mathbf{A}$ is shown in Figure 3, where the notation $m = dn - rank(A)$ denotes inextensional mechanisms including both possible infinitesimal mechanisms and rigid body motions.

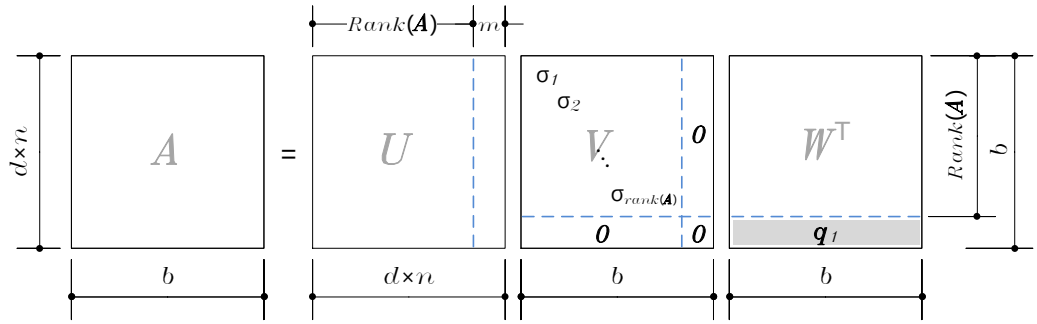


Figure 3. Graphical illustration of the SVD of the equilibrium matrix **A**.

A flowchart of the conventional form-finding process of tensegrity structures using the FDM is shown in Figure 4. In the works completed by [26,27] and many other researchers, two parallel equilibrium equations (Equations (6) and (14)) were iteratively computed to

obtain self-equilibrium configurations of tensegrity structures. The EVD and SVD processes are then used to obtain the force densities and nodal coordinates, respectively. Numerous studies combined the process of the FDM with meta-heuristic algorithms, such as a genetic algorithm. The form-finding problem is then formulated as a minimization problem by a special fitness function, which is based on some of the smallest eigenvalues of the force density matrix. However, the eigenvalue analysis and spectral decomposition are inevitable for the form-finding process using the FDM. In this paper, a DNN-based form-finding method of tensegrity systems is proposed to eliminate the complicated computational procedures in the whole form-finding process.

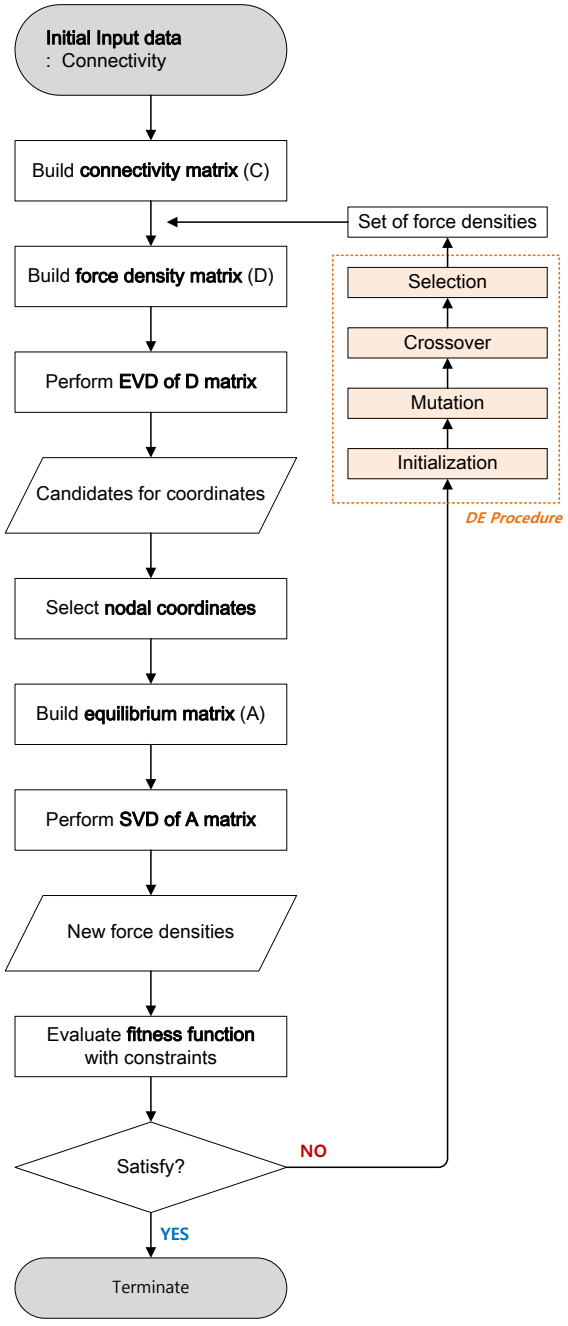


Figure 4. Flowchart of the conventional form-finding process of tensegrity structures using the FDM.

3. DNN-Based Form-Finding Method

The DNN, which is a neural network with multiple hidden layers, is a typical implementation of a deep architecture [28] as shown in Figure 5. The input and output layers are for the force densities and lengths of the elements. Figure 6 shows the flowchart of the DNN-based form-finding model of tensegrity structures. In this model, the linear algebra calculation, namely the EVD and the SVD, can be substituted by the DNN to obtain the nodal coordinate information. The trained DNN can then be used to predict the lengths of all elements. If two end points of one element are fixed by arbitrary locations, other nodal coordinates are determined automatically. As a result, the feasible set of nodal coordinates and force densities can be obtained without a process of the FDM.

3.1. Deep Neural Network Architectures for Training

The procedure for building and training a DNN is divided into three major phases, namely data preparation, training with optimization and prediction [29].

3.1.1. Data Preparation

A total dataset of 50,000 is used to train the DNN. As the set of force density is changed randomly, the lengths of tensegrity members are collected by using the FDM. All conditions, such as the connectivity, grouping and cable-strut labeling, are then fixed. For each input data, namely the set of force densities, the upper and lower bounds of the range $[0, 1]$ were given. Because the force densities are already normalized by the range, $[0, 1]$, data normalization processes are not needed to collect the dataset. However, even though the set of output data, namely length information, is not normalized by a range of meanings, the original dataset is used to train the DNN. The dataset was randomly split 80/20 into train vs. test sets.

3.1.2. Training Phase

To find the optimal architectures of the DNN, a set of experiments with various numbers of hidden layers, where each layer has various total numbers of neurons such as 128, 256, 512 or 1024, are performed. However, because the cases of two-dimensional two-strut tensegrity and three-dimensional-truncated tetrahedron have small number of elements, the number of hidden layers was fixed as three. Instead, various numbers of hidden layers were implemented to decide the best number in the three-dimensional-truncated icosahedron tensegrity example.

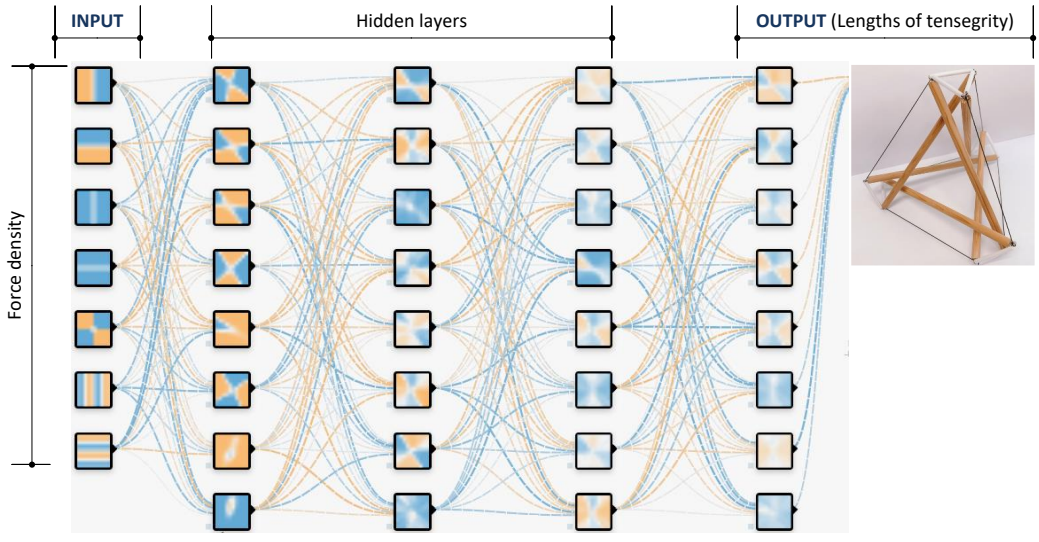


Figure 5. The DNN models with input and output layers denoted for the force densities and lengths of the elements, respectively.

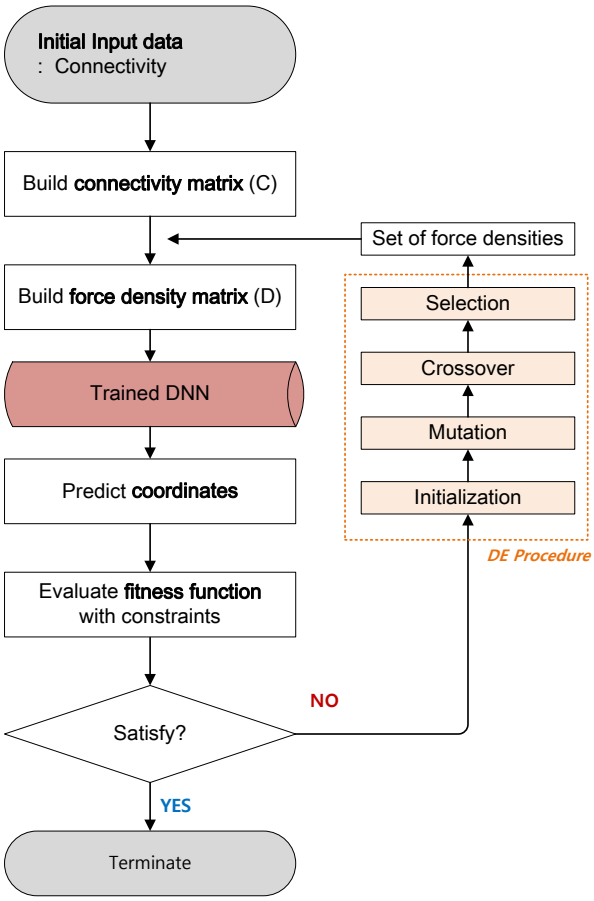


Figure 6. Flowchart of the DNN-based form-finding process of tensegrity structures to eliminate the calculation of EVD and SVD.

The values of neurons are intimately related with the batch size. Normally, when the algorithms training is performed, the networks take all the dataset as the input. However, if the size of the dataset is too large, the DNN is inefficient even if the computer system has enough memory. To speed-up stochastic convex optimization problems, mini-batch algorithms are used [30]. Because the mini-batches are vectorized in GPU systems, choosing a binary mini-batch size can lead to good performances. Several research groups started with two threads to determine the optimal number of batches or neuron size [31], however, larger batch sizes allow for better utilization of GPU memory bandwidth and improve computational throughput [32]. A set of experiments varying the batch size (or neurons) of 128, 256, 512 or 1024 was performed. This study revealed that the case of 512 neurons in each hidden layer provides the highest accuracy rate. In this study, mean squared error (MSE) loss is used to verify the neural networks.

Figure 7 is indicative of the DNN for training the 2D two-strut tensegrity structure. Three hidden layers using 512 neurons are used to perform the training process. Deep learning techniques such as the rectified linear unit (ReLU) activation function, the dropout algorithm, and the Adam optimizer are used [33]. The activation functions play a key role in a DNN to achieve better performances. Here, the dropout probability of 0.2 is fixed for all experiments. The ReLU activation function [34] is used here with the following form:

$$\mathrm{relu}(x) = \max(0, x), \tag{20}$$

which has the gradient:

$$\frac{d}{dx}\text{relu}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{21}$$

The aim is to solve the vanishing gradient and exploding gradient problems.

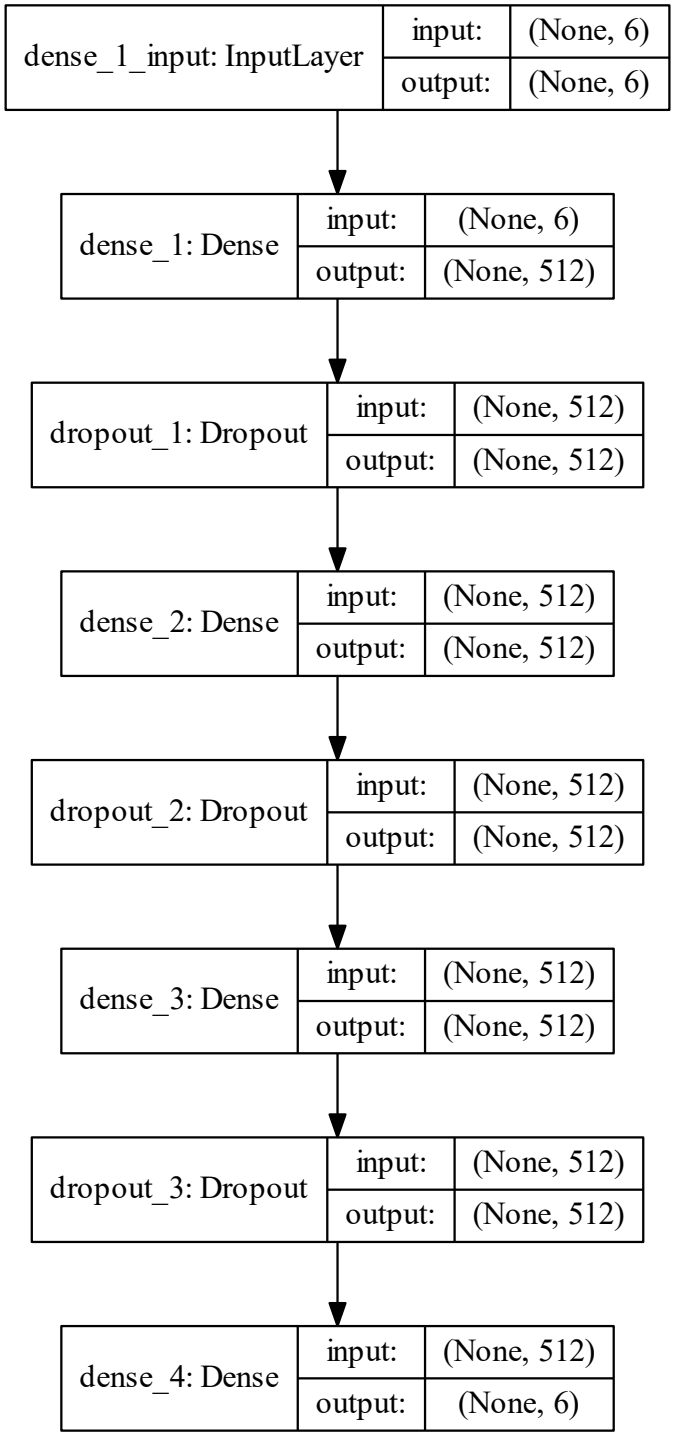


Figure 7. Visualization of the DNN for training the 2D two-strut tensegrity structure.

3.2. Optimization Algorithm

Based on our previous work [19], this study used the modified differential evolution (MDE) algorithm to build a DNN-based surrogate method. The differential evolution (DE) algorithm [35] is used to find the global minimum of the constrained optimization problems.

The main procedure of this algorithm is divided into four phases, namely initialization, mutation, crossover and selection. The bounds of variables, values of tolerance, and population sizes are set to be $[0, 1]$, 10^{-4} and 20, respectively. In evolutionary algorithms, the crossover operator combines features from different parents. Since the mutation operator is based on a recombination of individuals, a recombination rate is needed to create a new candidate solution. Here the recombination rate is set to be 0.9. Table 3 shows the values of the main parameters for the DE algorithm.

Table 3. Main parameters of DE algorithm.

Parameter	Value
Bounds of variables	$[0, 1]$
Population size	20
Recombination rate	0.9
Tolerance	10^{-4}

The fitness function consists of a value as follows:

$$\text{Minimize} : \sum_{i=1}^{b} \frac{1}{|q_i|} \tag{22}$$

$$\text{Subject to} : \mathbf{D}[\mathbf{x}\ \mathbf{y}\ \mathbf{z}] = [\mathbf{0}\ \mathbf{0}\ \mathbf{0}] \tag{23}$$

As suggested by Koohestani [36], the fitness function is a variable that significantly increases its value related to the force densities with zero or near-zero values. Moreover, the constraint, Equation (23), guarantees the equilibrium status.

3.3. Prediction for Lengths and Designation of Coordinates

After fixing the DNN weights, the lengths of whole elements of the tensegrity systems can be predicted by using the input data of the set of force densities. The force density set of the input is obtained from the DE algorithm. The algorithm updates the variables of the force density set in each generation; the information of lengths can then be predicted by using the trained DNN. By using the length information, the coordinates can be designated to obtain the final geometry of the tensegrity systems.

4. Numerical Examples

In this section, three well-known tensegrity examples including 2D two-strut, 3D-truncated tetrahedron and icosahedron tensegrity are investigated to validate the proposed model and show their performance with a conventional one. Three NVIDIA TITAN V GPU with 12GB of memory are used, and all experiments are executed on Ubuntu 16.04 OS with 3840 CUDA core. Unless mentioned otherwise, the rectified linear unit (ReLU) and the Adam method are used as the activation function and optimizer. The dropout probability is fixed as 0.2, and simulations are based on libraries of Tensorflow [37] and Keras [38].

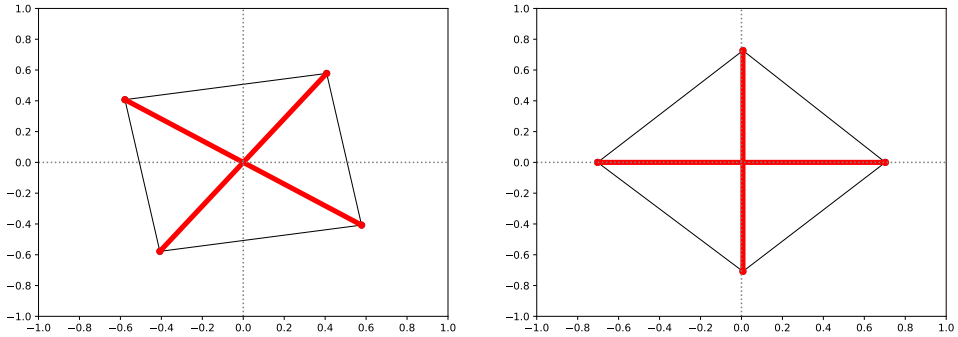
4.1. Two-Dimensional Two-Strut Tensegrity

4.1.1. Data Preparation and Training Phase

A dataset of 50,000 is obtained for the training of the 2D two-strut tensegrity. The force densities are randomly produced by using the force density analysis; the set of lengths is then obtained. The two groups, namely cables and struts, are used for generating the dataset. Each input data has upper and lower bounds of the range $[0, 1]$. Three hidden layers using 512 neurons are used to perform the training process (Figure 7). The number of neurons in the input layer is three, and they represents a set of force densities. The output layer indicates the element lengths.

4.1.2. Optimization Algorithm

The DE algorithm is performed to find the global minimum of the constrained optimization problems. Both methods use only one iteration for the DE process. Each iteration does not analyse FDM. Instead, the trained DNN predicts the set of element lengths. The results of the conventional FDM and the proposed DNN-based form-finding model with the DE algorithm are plotted in Figure 8 and given in Table 4. The final configuration by the conventional FDM seems to be the same as those of the proposed DNN-based form-finding model. However, their force density results are different. The results of the force density in Table 4 are identical to those of the previous study [39]. Thus, this proposed model can be validated. Moreover, the absolute value of $|\, f_{mean} - f_{best} \,|$ is lower that of the conventional FDM method.



(**a**) (**b**)

Figure 8. The final results of 2D two-strut tensegrity structure using the (**a**) FDM and (**b**) DNN-based form-finding model with DE algorithm.

Table 4. The final results of the 2D two-strut tensegrity structure using the FDM and DNN-based form-finding model with the DE algorithm.

	DE + FDM	**DE + DNN**	**[27]**
$\lvert\, f_{mean} - f_{best} \,\rvert$	0.66	0.56	
Solution [cable, strut]	[1.00, −0.97]	[1.00, −1.00]	
Force density * [cable, strut]	[1.00, −0.97]	[1.00, −1.00]	[1.00, −1.00]

* Normalized with respect to the solution of Element 1.

4.1.3. Prediction for Lengths and Designation of Coordinates

Six force density values are updated using the DE algorithm. The new selected set of force densities is then used to predict the lengths of whole elements of the tensegrity system. The trained DNN gives a set of element lengths, $[1.0, 1.0, 1.0, 1.0, 1.4, 1.4]$, as the prediction results. However, because the nodal coordinates have not yet been obtained, a designation process is needed to determine its geometry.

Figure 9 shows the designation process of the nodal coordinates of the 2D two-strut tensegrity structure. The left-hand side of Figure 9 indicates the obtained length conditions. When the midpoint of the fifth element is fixed at the $O(0, 0)$ point, Nodes 1 and 3 can be assigned as shown in Figure 9. The last two nodal points, Nodes 2 and 4, can be calculated by Equation (12), which is reorganized as

$$\sum_{p=1}^{6} l_p^2 = \|\mathbf{C}\phi_1 + \mathbf{C}\phi_2\|^2 \tag{24}$$

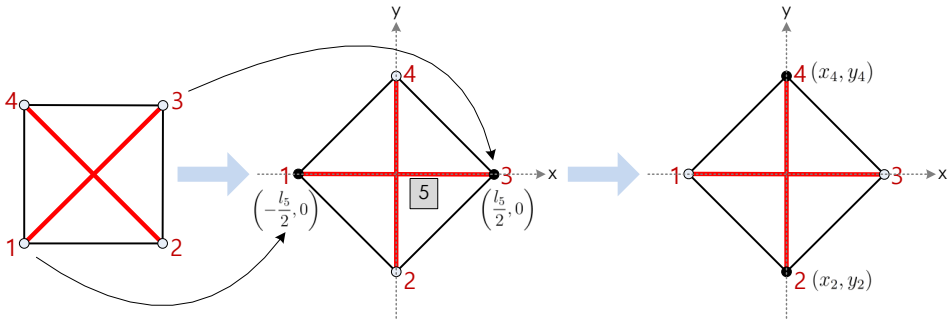


Figure 9. Designation process of the nodal coordinates of the 2D two-strut tensegrity structure.

By using the obtained lengths and the coordinates of Nodes 1 and 3, Equation (24) can be rewritten as a matrix-form:

$$\begin{bmatrix} (1.0)^2 \\ (1.0)^2 \\ (1.0)^2 \\ (1.0)^2 \\ (1.4)^2 \\ (1.4)^2 \end{bmatrix} = \begin{bmatrix} \left(-\frac{l_5}{2} - x_2\right)^2 + y_2^2 \\ \left(x_2 - \frac{l_5}{2}\right)^2 + y_2^2 \\ \left(\frac{l_5}{2} - x_4\right)^2 + y_4^2 \\ \left(-\frac{l_5}{2} - x_4\right)^2 + y_4^2 \\ \left(-\frac{l_5}{2} - x_3\right)^2 \\ (x_2 - x_4)^2 + (y_2 - y_4)^2 \end{bmatrix} \tag{25}$$

The above six simultaneous equations in Equation (25) can be used to draw the final geometry.

4.2. Three-Dimensional Truncated Tetrahedron

A 3D-truncated tetrahedron tensegrity invented by Buckminster [40] has 12 nodes and 24 members (6 struts and 18 cables) as shown in Figure 10. In geometry, it is constructed by cutting off the vertices of a tetrahedron.

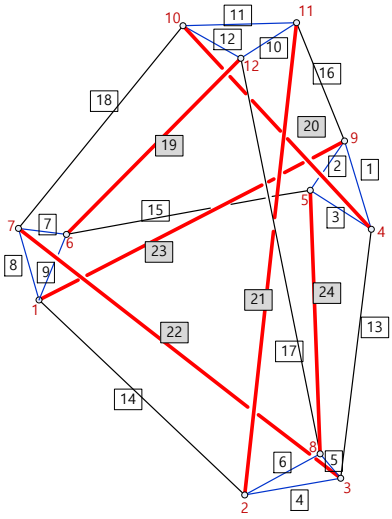


Figure 10. Connectivity of the 3D-truncated tetrahedron tensegrity.

4.2.1. Data Preparation and Training Phase

A dataset of 50,000 is obtained for the training of the 3D-truncated tetrahedron tensegrity. The three groups, namely edge cables q_e, vertical cables q_v and struts q_s, are used for generating the dataset. The DNN for training is the same as those of the 2D two-strut tensegrity structure. A set of experiments is performed with various numbers of hidden layers such as 2, 3, 4 or 5 with 128, 256, 512 or 1024 neurons, respectively. Figure 11 shows the comparison of the test loss for various mini-batch sizes. The results for cases of 128, 256 and 512 show the lowest loss values compared to those for the case of 1024. Because training with a large batch size is attractive due to its performance benefits, the mini-batch size of 512 is selected to train the architecture. A multilayer perceptron with three hidden layers is used. Each layer has 512 neurons; the mini-batch size is 512.

Figure 11. Comparison of the validation loss for various mini-batch sizes with log-scale for 3D-truncated tetrahedron tensegrity.

4.2.2. Optimization Algorithm

A total number of five iterations is implemented to obtain the final set of force densities. The final results are given in Table 5 and comparison of the force densities of the 3D-truncated tetrahedron tensegrity using an FDM, a DNN-based form-finding model and an analytical solution is plotted in Figure 12. Tibert and Pellegrino [6] found the analytical solution for this structure. The result of the proposed DNN-based form-finding model, $[1.00, 1.00, -0.58]$, is in good agreement with the analytical solution, which again validates this approach. However, the result of the conventional FDM, $[1.00, 0.73, -0.39]$, does not match with the super-stable line. To obtain more accurate results by using the conventional FDM, more iterations are demanded. Figure 13 indicates the convergence history and their relation to the iteration and error of the two methods. The proposed method using the DNN shows a much faster convergence than those of the conventional one.

Table 5. The final results of the 3D-truncated tetrahedron tensegrity using an FDM and a DNN-based form-finding model with a DE algorithm.

	DE + FDM	DE + DNN
Iteration	5	5
$\lvert f_{mean} - f_{best} \rvert$	0.22	0.14
Solution $[q_e, q_v, q_s]$	[0.96, 0.70, −0.38]	[1.00, 1.00, −0.58]
Force density * $[q_e, q_v, q_s]$	[1.00, 0.73, −0.39]	[1.00, 1.00, −0.58]

* Normalized with respect to the solution of Element 1.

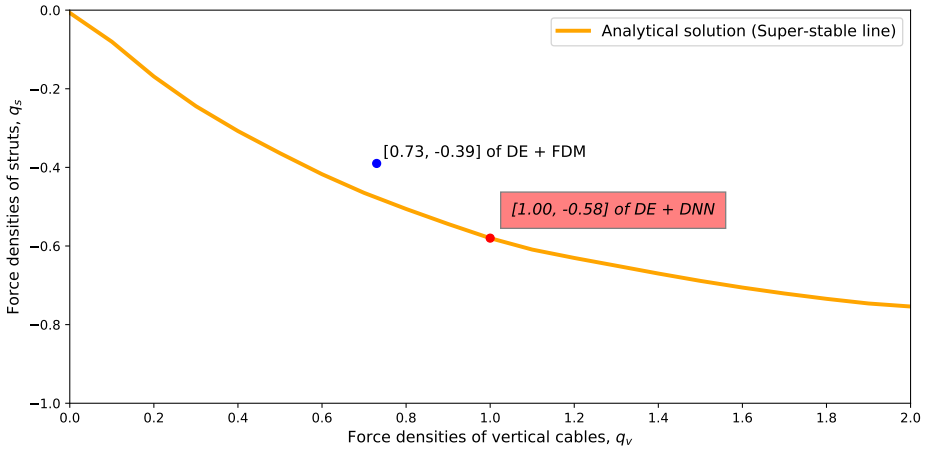


Figure 12. Comparison of the force densities of the 3D truncated tetrahedron tensegrity using an FDM, a DNN-based form-finding model with a DE algorithm and a analytical solution [6].

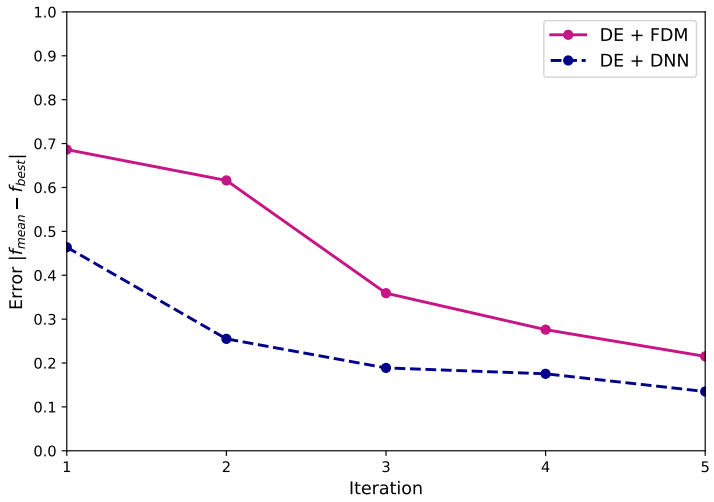


Figure 13. Relationship between the iteration and error for the 3D-truncated tetrahedron tensegrity using an FDM and a DNN-based form-finding method with a DE algorithm.

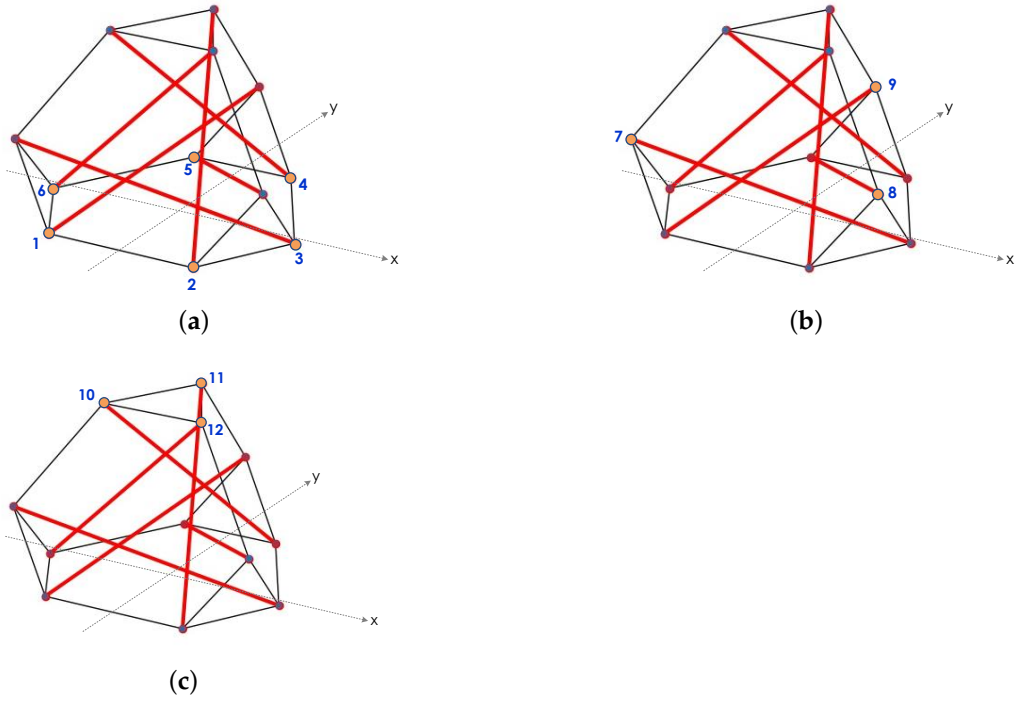
4.2.3. Prediction for Lengths and Designation of Coordinates

As mentioned above, new lengths of the 3D-truncated tetrahedron tensegrity can be predicted at each generation by using the trained DNN. The lengths should be grouped into three kinds of attributes. The trained network gives a set of element lengths, $[0.36, 0.52, 0.95]$, for the three groups, namely edge cables, vertical cables and struts. Figure 14 shows three steps of the coordinate designation for the 3D-truncated tetrahedron tensegrity. First, the pin joints, Nodes 1 to 6, are set to zero z-axis level; the centroid of the bottom face consisting of Nodes 1 to 6 lies at the origin point. The six nodal points can then be designated automatically. After definite values of Nodes 1 to 6 are assigned, the values of

the next level depend upon the value of the bottom level. Nodes 7 to 9 can be assigned by Equation (12). For the 3D-truncated tetrahedron tensegrity system, Equation (12) is reorganized as

$$\sum_{p=1}^{24} l_p^2 = \|\mathbf{C}\phi_1 + \mathbf{C}\phi_2 + +\mathbf{C}\phi_3\|^2 \tag{26}$$

The above 24 simultaneous equations in Equation (26) can used to obtain the final geometry. After obtaining the coordinates of Nodes 7 to 9, the last Nodes 10 to 12 can be assigned by using the equations corresponding to the nodal points.

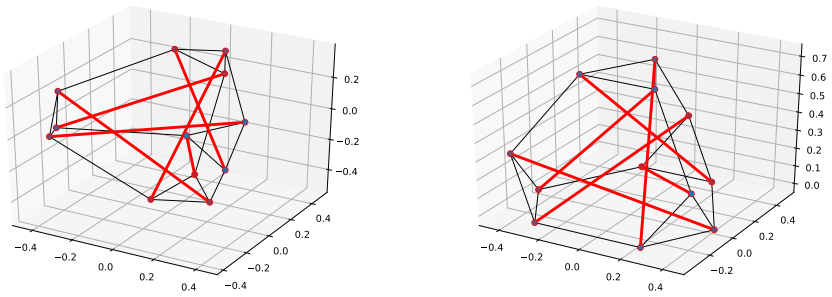


(**a**) (**b**)

(**c**)

Figure 14. Three steps of the coordinate designation for the 3D-truncated tetrahedron tensegrity. (**a**) Step 1: Nodes 1 to 6. (**b**) Step 2: Nodes 7 to 9. (**c**) Step 3: Nodes 10 to 12.

The final geometry obtained by the proposed DNN-based form-finding model shows more uniform shape than that of the FDM in Figure 15. Without the FDM, the feasible set of force densities and nodal coordinates can be readily obtained by using the proposed model.

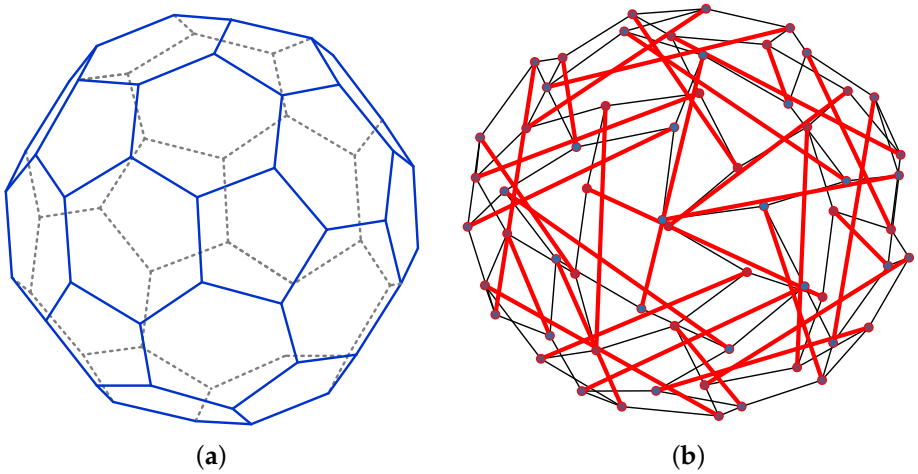


(**a**) (**b**)

Figure 15. Final geometry of five iterations of the 3D truncated tetrahedron tensegrity using an FDM and a DNN-based form-finding model with a DE algorithm. (**a**) FDM. (**b**) DNN-based form-finding model.

4.3. Three-Dimensional-Truncated Icosahedron Tensegrity

The 3D-truncated icosahedron tensegrity in Figure 16 is formed by cutting off 12 vertices along a plane perpendicular to the radius emanating from the center of the icosahedron. This creates 12 new pentagon faces and leaves the original 20 triangle faces as regular hexagons. It has 60 nodes and 120 elements that consist of 30 identical compression members and 90 tension cables. Figure 17 and Table 6 show the connectivity of the 3D-truncated tetrahedron tensegrity. The blue dotted lines denote vertical cables, while the green solid line represents edge cables. The nodal points of the 3D-truncated tetrahedron tensegrity are numbered on the model as shown in Figure 18.



(**a**) (**b**)

Figure 16. 3D-truncated icosahedron tensegrity in which the red thick lines denote strut members, and others are cable members. (**a**) Geometry. (**b**) Isometric view.

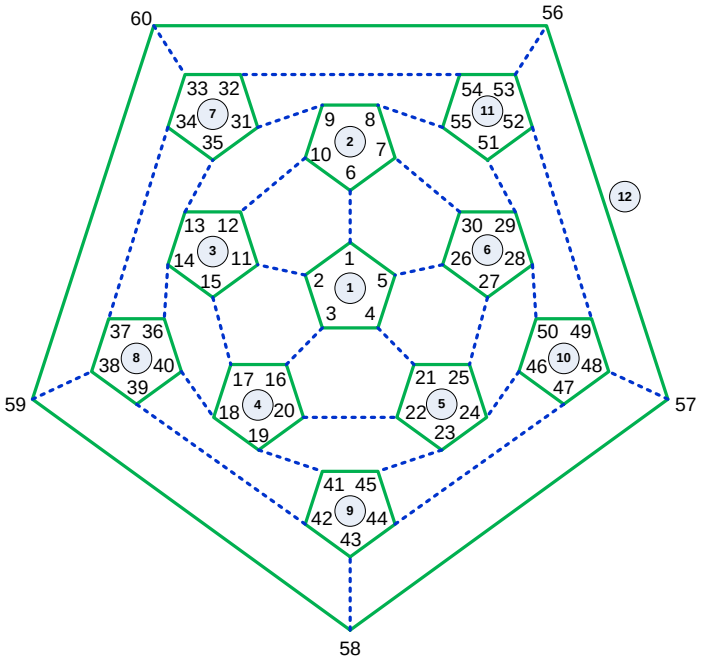


Figure 17. Connectivity of the 3D-truncated tetrahedron tensegrity in which the blue dotted lines denote vertical cables, and others represent edge cables.

Table 6. Connectivity of the strut members for the 3D-truncated tetrahedron tensegrity.

Element	i	j	Element	i	j	Element	i	j
91	37	58	101	24	26	111	13	40
92	14	16	102	27	49	112	12	34
93	2	20	103	23	50	113	33	36
94	1	15	104	22	44	114	32	59
95	5	10	105	43	46	115	8	35
96	6	29	106	42	57	116	9	11
97	28	55	107	19	21	117	7	54
98	48	51	108	18	45	118	31	53
99	4	30	109	38	41	119	47	56
100	3	25	110	17	39	120	52	60

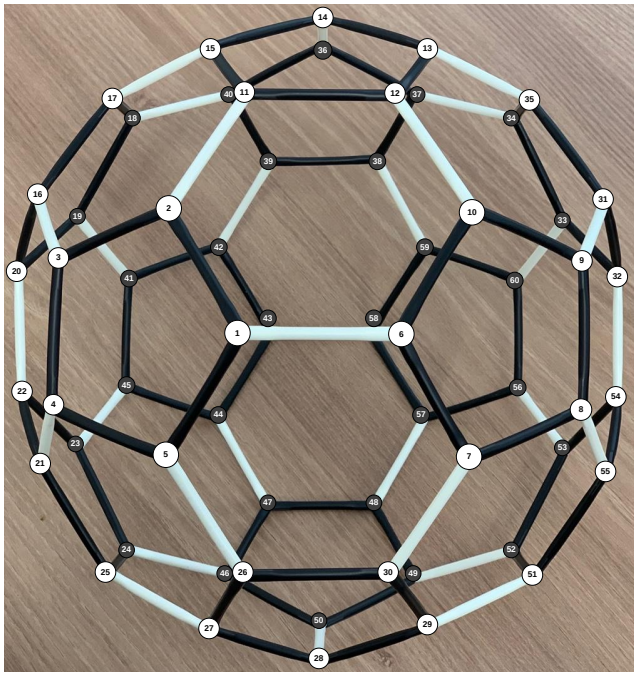


Figure 18. The nodal points of the 3D-truncated tetrahedron tensegrity are numbered on the model.

4.3.1. Data Preparation and Training Phase

The dataset of the 3D-truncated tetrahedron tensegrity system for training DNNs was 50,000. The three groups, namely edge cables q_e, vertical cables q_v and struts q_s, are used for generating the dataset. The dataset is uniformly distributed on the bounds.

Deciding the number of hidden layers and the number of neurons in each hidden layer is a challenging issue [41]. Figure 19 shows a comparison of the validation loss for various hidden layer sizes. In this figure, each hidden layer used 512 neurons and the ReLU activation function. The validation loss result of the case of three hidden layers is higher than those of other cases. Because the cases of two to five hidden layers are similar to each other, the case of three hidden layers is selected to train the DNN.

Currently, the most widely-used activation function is the rectified linear unit (ReLU). After the ReLu activation function has been introduced, various hand-designed alternatives, such as a leaky ReLU [42], an exponential linear unit (ELU) [43] and a scaled exponential linear unit (SELU) [44], to the ReLU are proposed. The leaky ReLU has the following form:

$$\text{lrelu}(x) = \begin{cases} \alpha x & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{27}$$

which has the gradient:

$$\frac{d}{dx}\text{lrelu}(x) = \begin{cases} \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{28}$$

where $\alpha = 0.01$, as proposed in the paper by [42].

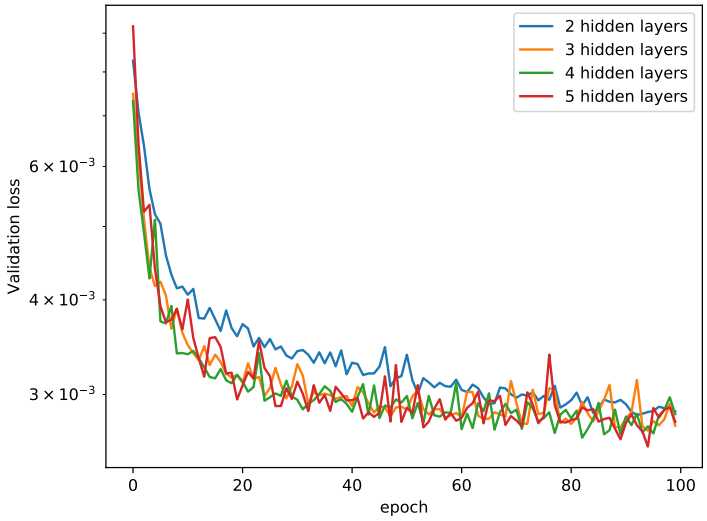


Figure 19. Comparison of the validation loss for various hidden layer sizes with log-scale. The number of neurons is 512 for the 3D-truncated icosahedron tensegrity.

The exponential linear unit (ELU) has the following form:

$$\text{ELU}(x) = \begin{cases} \alpha(\exp(x) - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{29}$$

which has the gradient:

$$\frac{d}{dx}\text{ELU}(x) = \begin{cases} \text{ELU}(x) + \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{30}$$

where $\alpha = 1$, as proposed in the paper by [43].

The scaled exponential linear Unit (SELU) has the following form:

$$\text{SELU}(x) = \lambda \begin{cases} \alpha(\exp(x) - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{31}$$

which has the gradient:

$$\frac{d}{dx}\text{SELU}(x) = \begin{cases} \text{SELU}(x) + \lambda\alpha & \text{if } x \leq 0 \\ \lambda & \text{if } x > 0 \end{cases} \tag{32}$$

where $\alpha = 1.6733$ and $\lambda = 1.0507$, as proposed in the paper by [44].

Figure 20 shows the graphs of the different variants of the activation functions. Comparisons of the training loss and the validation loss for various activation functions are plotted in Figure 21. The architectures of the DNN have three hidden layers with 512 neurons. The cases of the ReLU and the leaky ReLU activation functions show better results than those of the ELU and SELU. For the 3D-truncated icosahedron tensegrity module,

the ReLU is selected as an activation function. As shown in Figure 22, the final trained DNN is then obtained to predict the element lengths.

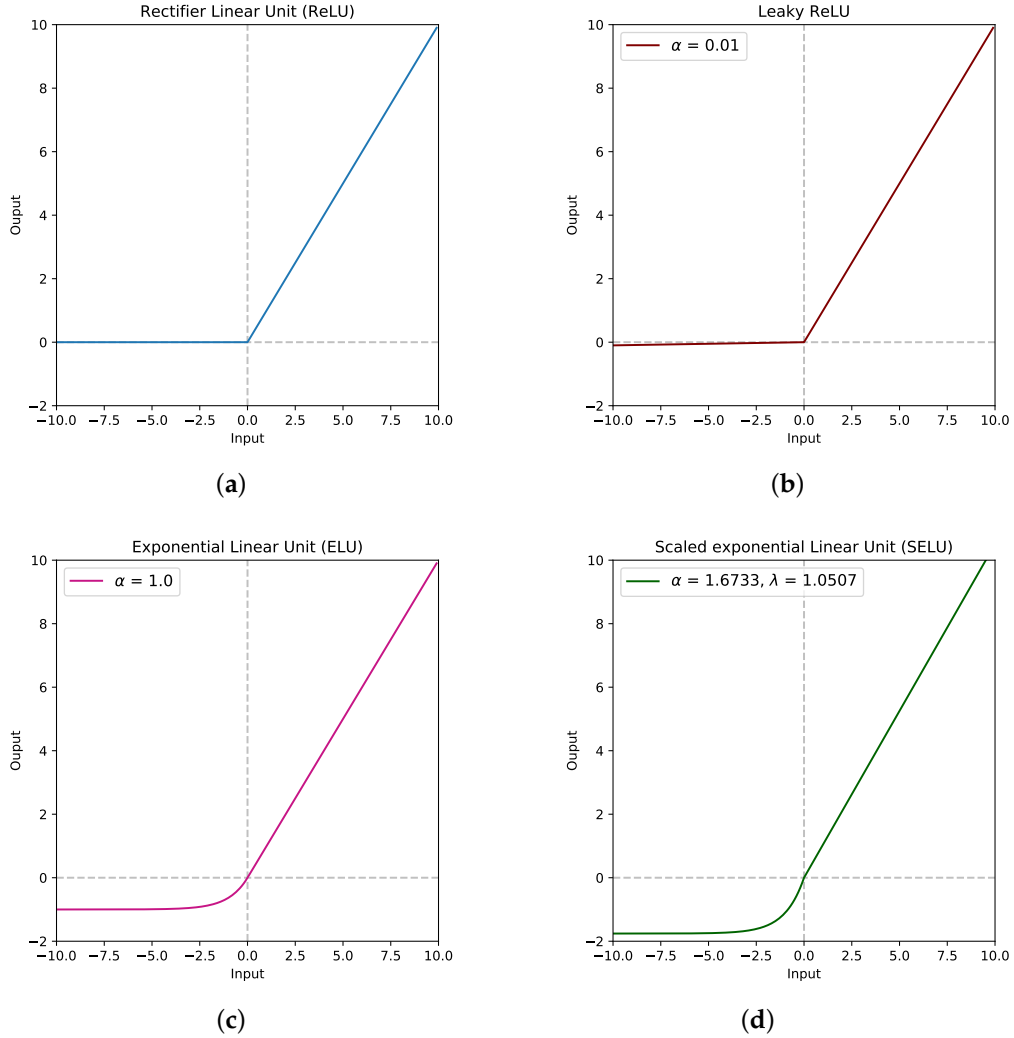


Figure 20. Variants of activation functions. (**a**) ReLU. (**b**) Leaky ReLU. (**c**) ELU. (**d**) SELU.

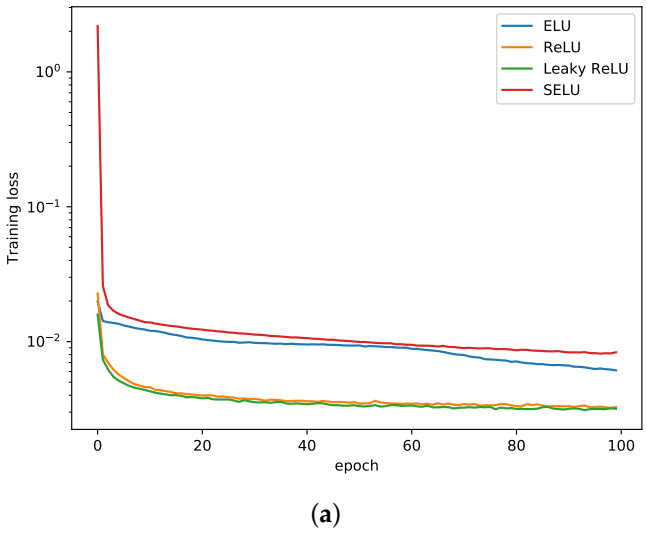
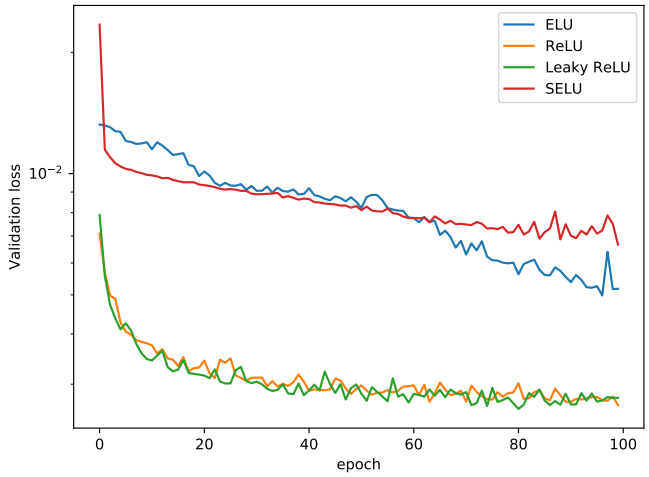


Figure 21. *Cont.*



(**b**)

Figure 21. Comparison of the training loss and validation loss for various activation functions with log scale. Three hidden layers and 512 neurons were used for the 3D-truncated icosahedron tensegrity example. (**a**) Training loss. (**b**) Validation loss.

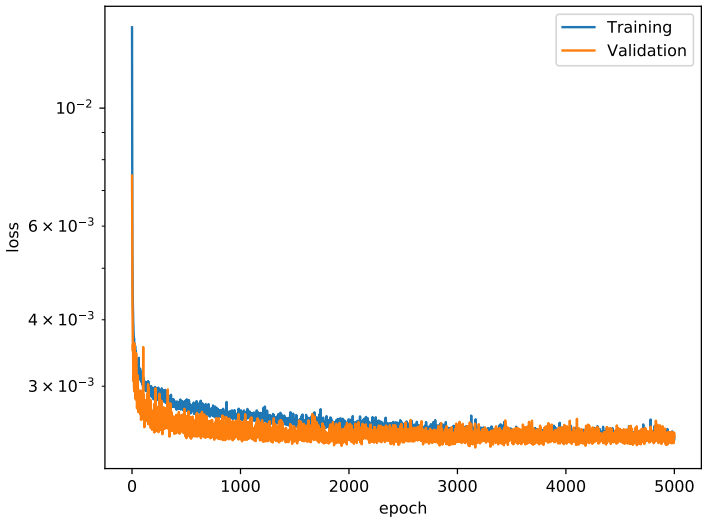
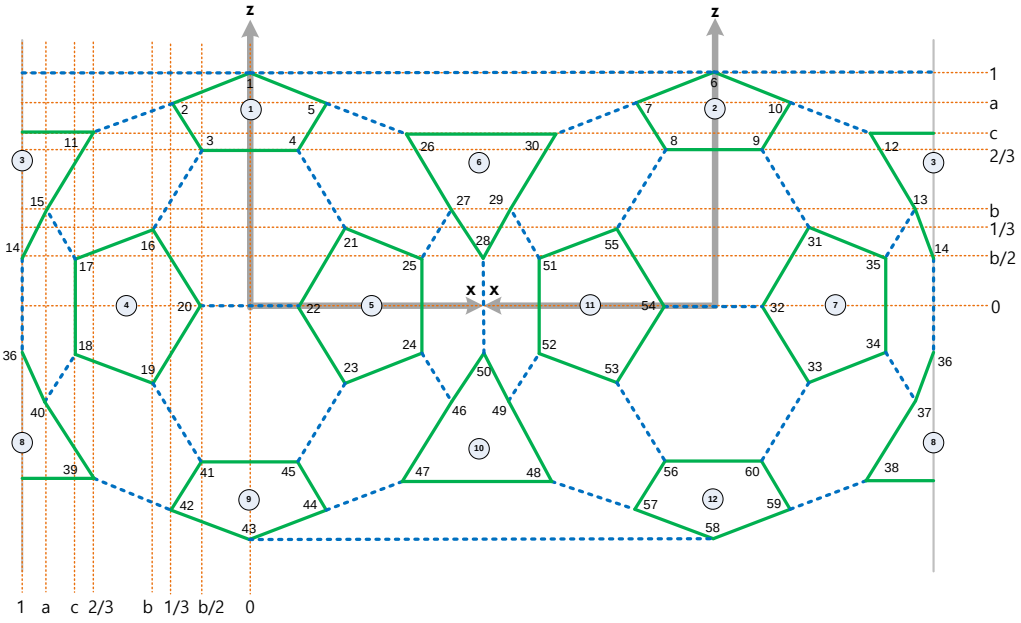
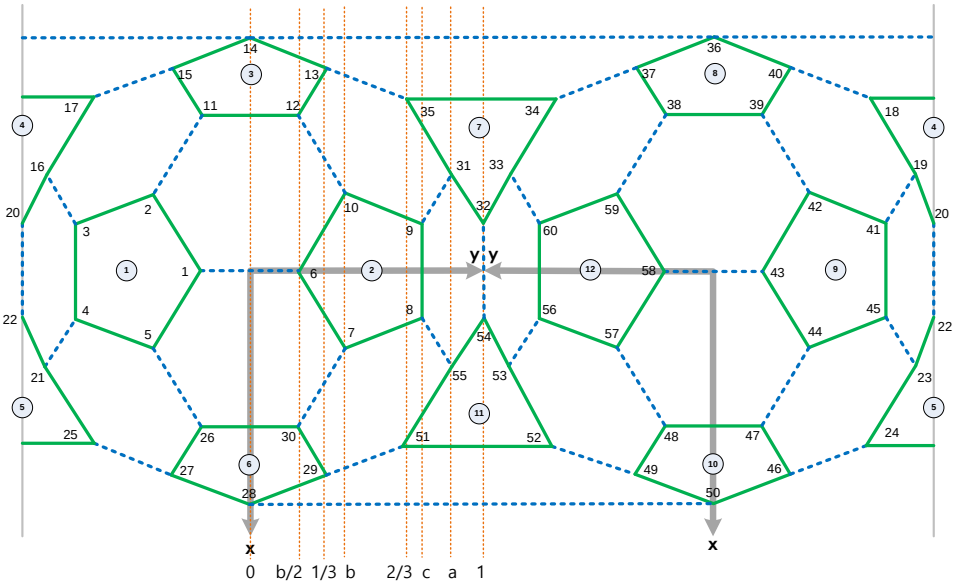


Figure 22. The training and validation loss curves for the 3D-truncated icosahedron tensegrity.

Figure 23 shows rectangular coordinates of the truncated icosahedron tensegrity in a $2 \times 2 \times 2$ cube. The $2 \times 2 \times 2$ cube is located in the 3D rectangular coordinate system so that the eight vertices occupy the position at $(\pm 1, \pm 1, \pm 1)$. In geometry, the truncated icosahedron tensegrity has 12 regular pentagonal faces, 20 regular hexagonal faces, 60 vertices and 90 edges. The truncated icosahedron tensegrity can be decomposed into 20 hexagonal pyramids and 12 pentagonal pyramids by drawing segments from all the vertices of the truncated icosahedron tensegrity toward the origin [45]. The analytical expression and numerical values of pin-joints of the truncated icosahedron tensegrity module are given in Table 7, where $a = (3 + \sqrt{5})/6$, $b = (\sqrt{5} - 1)/3$, and $c = \sqrt{5}/3$. Here, the notation b indicates the edge length of the pentagonal face.



(**a**)



(**b**)

Figure 23. Rectangular coordinates of the 3D-truncated icosahedron tensegrity in a $2 \times 2 \times 2$ cube, where $a = (3+\sqrt{5})/6$, $b = (\sqrt{5}-1)/3$, and $c = \sqrt{5}/3$. (**a**) Front view. (**b**) Top view.

Table 7. Cartesian coordinates of the 3D-truncated icosahedron tensegrity using the connectivity shown in Figure 17, where $a = (3+\sqrt{5})/6$, $b = (\sqrt{5}-1)/3$, and $c = \sqrt{5}/3$.

No.	x	y	z	No.	x	y	z
1	0	$-b/2$	1	31	$-b$	a	1/3
2	$-1/3$	$-b$	a	32	$-b/2$	1	0
3	$-b/2$	$-c$	2/3	33	$-b$	a	$-1/3$
4	$b/2$	$-c$	2/3	34	$-c$	2/3	$-b/2$
5	1/3	$-b$	a	35	$-c$	2/3	$b/2$
6	0	$b/2$	1	36	-1	0	$-b/2$

Table 7. *Cont.*

No.	x	y	z	No.	x	y	z
7	$1/3$	b	a	37	$-a$	$1/3$	$-b$
8	$b/2$	c	$2/3$	38	$-2/3$	$b/2$	$-c$
9	$-b/2$	c	$2/3$	39	$-2/3$	$-b/2$	$-c$
10	$-1/3$	b	a	40	$-a$	$-1/3$	$-b$
11	$-2/3$	$-b/2$	c	41	$-b/2$	$-c$	$-2/3$
12	$-2/3$	$b/2$	c	42	$-1/3$	$-b$	$-a$
13	$-a$	$1/3$	b	43	0	$-b/2$	-1
14	-1	0	$b/2$	44	$1/3$	$-b$	−a
15	$-a$	$-1/3$	b	45	$b/2$	$-c$	$-2/3$
16	$-b$	$-a$	$1/3$	46	a	$-1/3$	$-b$
17	$-c$	$-2/3$	$b/2$	47	$2/3$	$-b/2$	$-c$
18	$-c$	$-2/3$	$-b/2$	48	$2/3$	$b/2$	$-c$
19	$-b$	$-a$	$-1/3$	49	a	$1/3$	$-b$
20	$-b/2$	-1	0	50	1	0	$-b/2$
21	b	$-a$	$1/3$	51	c	$2/3$	$b/2$
22	$b/2$	-1	0	52	c	$2/3$	$-b/2$
23	b	$-a$	$-1/3$	53	b	a	$-1/3$
24	c	$-2/3$	$-b/2$	54	$b/2$	1	0
25	c	$-2/3$	$b/2$	55	b	a	$1/3$
26	$2/3$	$-b/2$	c	56	$b/2$	c	$-2/3$
27	a	$-1/3$	b	57	$1/3$	b	$-a$
28	1	0	$b/2$	58	0	$b/2$	-1
29	a	$1/3$	b	59	$-1/3$	b	$-a$
30	$2/3$	$b/2$	c	60	$-b/2$	c	$-2/3$

Because the Cartesian coordinates described in Table 7 are for a $2 \times 2 \times 2$ cube, a scale factor is needed to obtain the nodal coordinates of the final tensegrity shape. If a notation l_e denotes the edge cable of the truncated icosahedron tensegrity, the scale factor is defined as follows:

$$\alpha = \frac{l_e}{b} \tag{33}$$

where $b = (\sqrt{5} - 1)/3 = 0.41202266$.

After obtaining the set of element lengths by using the trained DNN, the nodal coordinates can be obtained by the scale factor.

4.3.2. Optimization Algorithm

A total number of eight iterations is implemented to obtain the final set of force densities. Table 8 shows the final results of the 3D-truncated icosahedron tensegrity using the FDM and the proposed DNN-based form-finding model. Figure 24 illustrates a comparison of the force densities of the truncated icosahedron tensegrity using an FDM, a DNN-based form-finding model with a DE algorithm and an analytical solution [6]. The result of the proposed method, $[1.00, 0.80, -0.35]$, is in complete agreement with the analytical solution while that of the FDM, $[1.00, 1.00, -1.00]$, does not satisfy the super-stable line. Figure 25 shows relationship between the iteration and error of the two methods. Again, the proposed method shows a much better convergence than a conventional FDM.

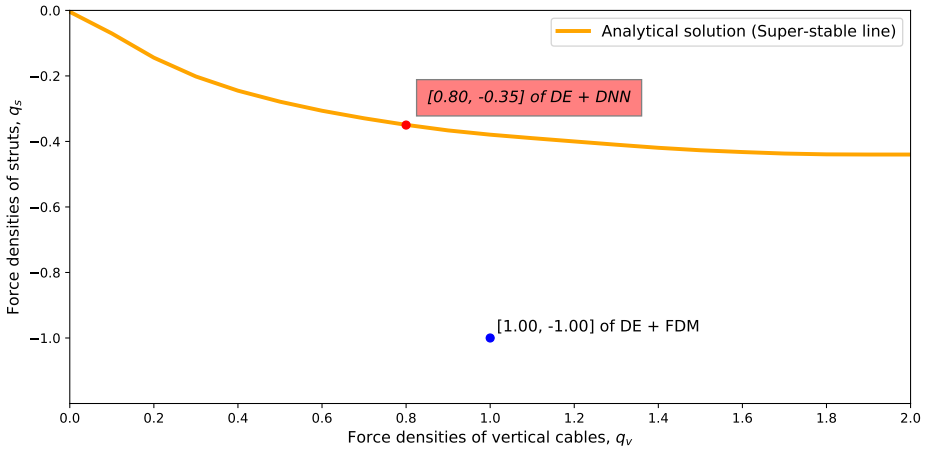


Figure 24. Comparison of the force densities of the 3D truncated icosahedron tensegrity using FDM, DNN-based form-finding model with DE algorithm and analytical solution [6].

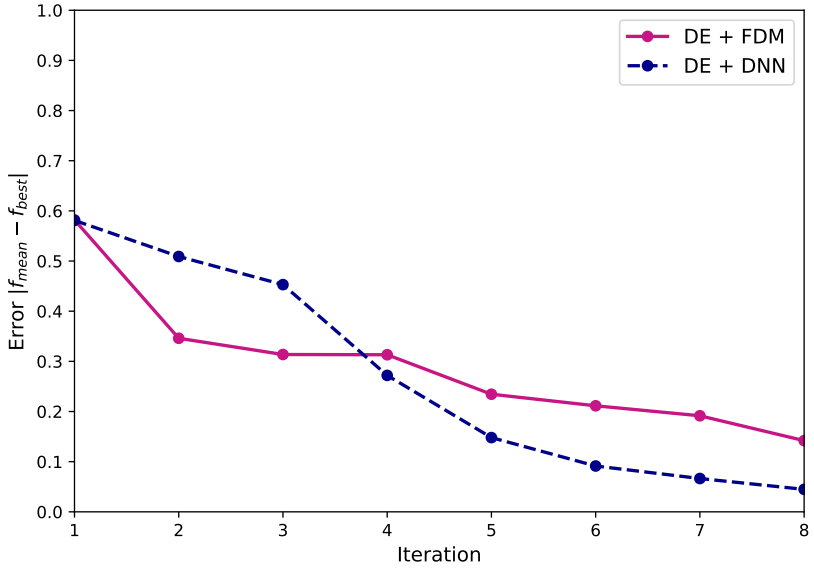


Figure 25. Relationship between the iteration and error of the 3D truncated icosahedron tensegrity using an FDM and a DNN-based form-finding model with a DE algorithm.

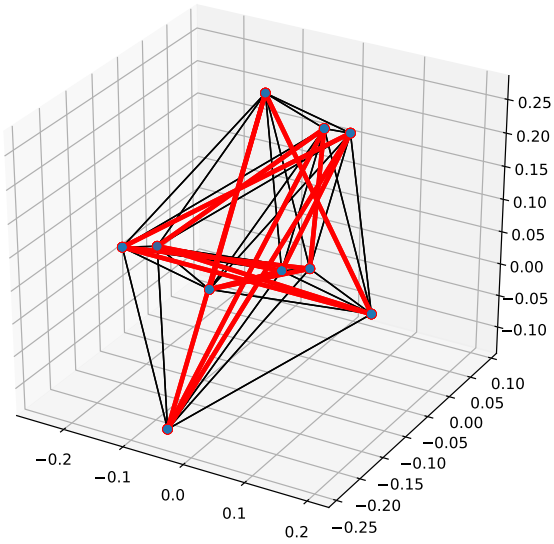
Table 8. The final results of the 3D-truncated icosahedron tensegrity using an FDM and a DNN-based form-finding model with a DE algorithm.

	DE + FDM	**DE + DNN**
Iteration	8	8
$\lvert f_{mean} - f_{best} \rvert$	0.14	0.04
Solution $[q_e, q_v, q_s]$	[1.00, 1.00, −1.00]	[1.00, 0.80, −0.35]
Force density * $[q_e, q_v, q_s]$	[1.00, 1.00, −1.00]	[1.00, 0.80, −0.35]

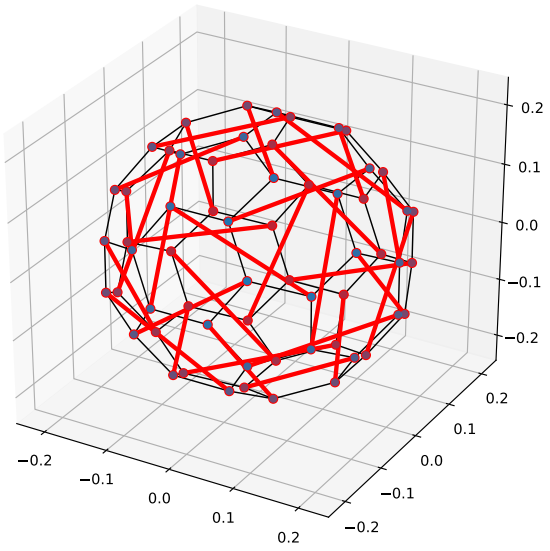
* Normalized with respect to the solution of element 1.

4.3.3. Prediction for Lengths and Designation of Coordinates

The final geometries of eight iterations of form-finding processes for the 3D-truncated icosahedron tensegrity module obtained by using the conventional FDM and the proposed DNN-based form-finding model with a DE algorithm are shown in Figure 26. The case of the proposed model shows a much better tensegrity shape compared with a conventional one.



(**a**)



(**b**)

Figure 26. Final geometry of eight iterations of the 3D-truncated icosahedron tensegrity module using an FDM and a DNN-based form-finding model with DE algorithm. (**a**) FDM. (**b**) DNN-based form-finding model.

5. Conclusions

In this study, the machine-learning-based form-finding method for tensegrity structures is proposed. The new form-finding method used state-of-the-art deep learning techniques to train the DNN. The trained DNNs can predict the feasible set of coordinates. The eigenvalue decomposition process of the force density matrix and the singular value decomposition process of the equilibrium matrix in the conventional approach are not needed to find the nodal point information. The proposed DNN-based form-finding model is based on the differential evolution algorithm, which is performed to find the global minimum of the constrained optimization problems. Three well-known tensegrity examples, the 2D two-strut, the 3D-truncated tetrahedron and the 3D icosahedron tensegrity are presented for numerical verifications. Variants of the activation functions such as the rectified linear unit (ReLU), the leaky ReLU, the exponential linear unit (ELU) and the

scaled exponential linear unit (SELU) are investigated. The cases of the ReLU and the leaky ReLU activation functions show better results than those of the ELU and SELU. Moreover, this study confirms that three of hidden layers and 512 neuron size are the best combination for the DNN structures. The results obtained by the proposed method are in complete agreement with those of analytical solution, stable line, or previous studies. It is concluded that the set of force densities obtained by the proposed DNN-based form-finding model shows more accurate results and their geometries have more uniform shape than those of the conventional form-finding process using the force density method. Future studies will be required to apply various regression methods such as kernel ridge regression, SVM (Support Vector Machines), decision tree, and ensemble methods to form-finding of tensegrity structures.

Author Contributions: Data curation, S.L.; Funding acquisition, J.L.; Methodology, S.L.; Software, Q.X.L.; Supervision, J.L.; Validation, S.L.; Visualization, S.L.; Writing—review & editing, S.L. and T.P.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by a grant (NRF-2021R1A2B5B03002410) from NRF (National Research Foundation of Korea) funded by MEST (Ministry of Education and Science Technology) of Korean government.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pugh, A. *An Introduction to Tensegrity*; University of California Press: Barkley, CA, USA, 1976.
2. Snelson, K. Continuous Tension, Discontinuous Compression Structures. U.S. Patent 3,169,611, 16 February 1965.
3. Zhang, J.Y.; Ohsaki, M. Adaptive force density method for form-finding problem of tensegrity structures. *Int. J. Solids Struct.* **2006**, *43*, 5658–5673. [CrossRef]
4. Skelton, R.E.; Adhikari, R.; Pinaud, J.P.; Chan, W.; Helton, J.W. An introduction to the mechanics of tensegrity structures. In Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228), Orlando, FL, USA, 4–7 December 2001; Volume 5, pp. 4254–4259.
5. Pellegrino, S.; Calladine, C.R. Matrix analysis of statically and kinematically indeterminate frameworks. *Int. J. Solids Struct.* **1986**, *22*, 409–428. [CrossRef]
6. Tibert, A.G.; Pellegrino, S. Review of form-finding methods for tensegrity structures. *Int. J. Space Struct.* **2003**, *18*, 209–223. [CrossRef]
7. Shi, M.; Wang, C.; Li, X.Z.; Li, M.Q.; Wang, L.; Xie, N.G. EEG signal classification based on SVM with improved squirrel search algorithm. *Biomed. Eng.* **2021**, *66*, 137–152. [CrossRef] [PubMed]
8. Shi, M.; Wang, C.; Zhao, W.; Zhang, X.; Ye, Y.; Xie, N. Removal of ocular artifacts from electroencephalo-graph by improving variational mode decomposition. *China Commun.* **2022**, *19*, 47–61. [CrossRef]
9. Wang, B.; Wang, C.; Wang, L.; Xie, N.; Wei, W. Recognition of semg hand actions based on cloud adaptive quantum chaos ions motion algorithm optimized SVM. *J. Mech. Med. Biol.* **2019**, *19*, 1950047. [CrossRef]
10. Domer, B.; Fest, E.; Lalit, V.; Smith, I.F. Combining dynamic relaxation method with artificial neural networks to enhance simulation of tensegrity structures. *J. Struct. Eng.* **2003**, *129*, 672–681. [CrossRef]
11. Panigrahi, R.; Gupta, A.; Bhalla, S.; Arora, K. Application of Artificial Neural Network for Form Finding of Tensegrity Structures. In Proceedings of the 2nd Indian International Conference on Artificial Intelligence, Pune, India, 20–22 December 2005; pp. 1950–1962.
12. Zalyaev, E.; Savin, S.; Vorochaeva, L. Machine Learning Approach for Tensegrity Form Finding: Feature Extraction Problem. In Proceedings of the 2020 4th Scientific School on Dynamics of Complex Networks and their Application in Intellectual Robotics (DCNAIR), Innopolis, Russia, 7–9 September 2020; pp. 265–268.
13. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef]
14. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
15. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.

16. Cera, B.; Agogino, A.M. Multi-cable rolling locomotion with spherical tensegrities using model predictive control and deep learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–9.
17. Zhang, M.; Geng, X.; Bruce, J.; Caluwaerts, K.; Vespignani, M.; SunSpiral, V.; Abbeel, P.; Levine, S. Deep reinforcement learning for tensegrity robot locomotion. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 634–641.
18. Luo, J.; Edmunds, R.; Rice, F.; Agogino, A.M. Tensegrity robot locomotion under limited sensory inputs via deep reinforcement learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6260–6267.
19. Do, D.T.; Lee, S.; Lee, J. A modified differential evolution algorithm for tensegrity structures. *Compos. Struct.* **2016**, *158*, 11–19. [CrossRef]
20. Lee, S.; Jeong, J.; Ahn, S.; Lieu, Q.X.; Lee, J. Performance of quadruplex module tensegrities using new pin-jointed connections. *J. Constr. Steel Res.* **2020**, *172*, 105763 . [CrossRef]
21. Lee, S.; Lee, J. Form-finding of tensegrity structures with arbitrary strut and cable members. *Int. J. Mech. Sci.* **2014**, *85*, 55–62. [CrossRef]
22. Pellegrino, S. Structural computations with the singular value decomposition of the equilibrium matrix. *Int. J. Solids Struct.* **1993**, *30*, 3025–3035. [CrossRef]
23. Schek, H.-J. The force density method for form finding and computation of general networks. *Comput. Methods Appl. Mech. Eng.* **1974**, *3*, 115–134. [CrossRef]
24. Connelly, R. Rigidity and energy. *Invent. Math.* **1982**, *66*, 11–33. [CrossRef]
25. Abdi, H. The eigen-decomposition: Eigenvalues and eigenvectors. In *Encyclopedia of Measurement and Statistics*; SAGE Publications, Inc.: New York, NY, USA, 2007; pp. 304–308.
26. Estrada, G.G.; Bungartz, H.J.; Mohrdieck, C. Numerical form-finding of tensegrity structures. *Int. J. Solids Struct.* **2006**, *43*, 6855–6868. [CrossRef]
27. Tran, H.C.; Lee, J. Advanced form-finding of tensegrity structures. *Comput. Struct.* **2010**, *88*, 237–246. [CrossRef]
28. Ze, H.; Senior, A.; Schuster, M. Statistical parametric speech synthesis using deep neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 7962–7966.
29. Oishi, A.; Yagawa, G. Computational mechanics enhanced by deep learning. *Comput. Methods Appl. Mech. Eng.* **2017**, *327*, 327–351. [CrossRef]
30. Hinton, G.E. A practical guide to training restricted Boltzmann machines. In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 599–619.
31. Huqqani, A.A.; Schikuta, E.; Ye, S.; Chen, P. Multicore and gpu parallelization of neural networks for face recognition. *Procedia Comput. Sci.* **2013**, *18*, 349–358. [CrossRef]
32. NVIDIA. NVIDIA Tesla P100 GPU Architecture. 2016. Available online: http://www.nvidia.com/object/pascal-architecture-whitepaper.html (accessed on 24 April 2022).
33. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
34. Pedamonti, D. Comparison of non-linear activation functions for deep neural networks on MNIST classification task. *arXiv* **2018**, arXiv:1804.02763.
35. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
36. Koohestani, K. Form-finding of tensegrity structures via genetic algorithm. *Int. J. Solids Struct.* **2012**, *49*, 739–747. [CrossRef]
37. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
38. Chollet, F. *Keras: The Python Deep Learning Library*; Astrophysics Source Code Library: Houghton, MI, USA, 2018.
39. Lee, S.; Lee, J. Advanced automatic grouping for form-finding of tensegrity structures. *Struct. Multidiscip. Optim.* **2017**, *55*, 959–968. [CrossRef]
40. Fuller, R.B. Tensile-Integrity Structures. U.S. Patent 3,063,521, 13 November 1962 .
41. Karsoliya, S. Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. *Int. J. Eng. Trends Technol.* **2012**, *3*, 714–717.
42. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 14–16 June 2013; Volume 30, p. 3.
43. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv* **2015**, arXiv:1511.07289.
44. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-normalizing neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 971–980.
45. Hosoya, H.; Maruyama, Y. Efficient generation of the cartesian coordinates of truncated icosahedron and related polyhedra. *J. Mol. Graph. Model.* **2001**, *19*, 205–209. [CrossRef]