Journal of Mechanical Science and Technology 36 (7) 2022

Original Article
DOI 10.1007/s12206-022-0628-2

Keywords:
· Topology optimization
· Deep learning
· BESO
· CNN
· Python

Correspondence to:
Cheol Kim
kimchul@knu.ac.kr

Citation:
Shin, J., Kim, C. (2022). Bi-directional evolutionary 3D topology optimization with a deep neural network. Journal of Mechanical Science and Technology 36 (7) (2022) 3509~3519.
http://doi.org/10.1007/s12206-022-0628-2

Received	August 18th, 2021
Revised	February 22nd, 2022
Accepted	March 1st, 2022

† Recommended by Editor Hyun-Gyu Kim

Bi-directional evolutionary 3D topology optimization with a deep neural network

Junseok Shin and Cheol Kim

Department of Mechanical Engineering, Kyungpook National University, Daegu 41566, Korea

Abstract The FEM-based topology optimization repeats usually finite element analyses many times to converge to the stopping criteria. If the near-optimal topology data are available in advance at the beginning of an optimization process, the iterative computation could be greatly reduced. In an effort to obtain swiftly optimum topology solutions, the deep learning and neural networks with a special segmentation scheme of digital images are combined with the BESO (bi-directional evolutionary structural optimization) topology method in this study. The pre-trained digital images of 3200 optimum topologies construct the design domain for the main topology optimization. Additionally, a new post-processor is developed in order to reconstruct the relative locations among finite elements in the raw outputs generated by the neural network. The proposed method has been demonstrated to be efficient in lowering the iterations with several 2D and 3D optimization examples. The iteration counts can be reduced 63 % for a 2D example and by 72.5 % for a 3D one, compared to BESO results alone.

1. Introduction

Breakthrough ideas using GPU for general purpose computing have resulted in highly cost-effective parallel computing power [1-3]. As the threshold for adopting deep learning has been lowered, the related studies have been conducted actively in various disciplines [4-9]. Many interesting applications of deep learning can be found in the area of topology optimization in recent years. The most time-consuming part of topology optimization is an iterative process that repeatedly executes finite element analyses to obtain solutions. Researches in the application of deep learning for topology optimization usually aim to reduce the number of iterations.

Ulu et al. [10] created a set of images representing the optimal 2D topologies acquired from the general topology optimization method. The feed-forward neural network is trained using loading configurations as an input and corresponding optimal topologies as an output. The neural network is able to predict the optimal topology for a new loading configuration. After this, many studies adopt mainly the convolutional neural network (CNN) which is effective for extracting the prominent feature of images. Sosnovik and Oseledets [11] introduced a neural network having a form of the convolutional autoencoder that is especially devised for image segmentation tasks. The intermediate values of design variables and their gradients were represented in the format of digital images. Those images were then used as an input for the neural network. Banga et al. [12] took the similar image segmentation scheme and extended it to the 3D topology optimization. The neural network by Zhang et al. [13] utilized the displacements and strains obtained from the first finite element analysis during topology optimization as an input. This approach gives a good generalization ability to the neural network. It is able to predict the optimum topology even for the boundaries that not included in the training dataset.

In the density-based topology optimization, the number of design variables is equal to the number of the elements in the design domain. Therefore, it would take a long time for convergence to an optimum solution, if the number of elements is too large. To solve this weakness, some methods based on a multi-resolution scheme were introduced. FEA and elements update are performed on a low resolution FE model and then the optimum solution for a high resolution model is obtained by interpolating the results of the low-resolution procedure [14, 15].

Xue et al. [16] incorporated CNN into the process of the multi-resolution topology optimization to reduce the time for the interpolation procedure.

Yu et al. [17] created both the low- and high-resolution 2D images of the optimum topologies obtained from traditional optimization. The low-resolution images of the optimum topologies are paired with the corresponding information such as forces, degree of freedom and mass fraction and then are used as a training set for a convolutional autoencoder. The generative adversarial network (GAN) interpolates the low-resolution optimal topology images predicted by the autoencoder to the high-resolution optimum topology images. Li et al. [18] adopted GANs for the topology optimization concerning thermal conduction. They used the GANs for both predicting a low-resolution optimal solution and interpolating the low-resolution optimum to a high-resolution one. Oh et al. [19] integrated the topology optimization into a deep generative design framework. The proposed design framework could generate more various and higher quality novel wheel designs satisfying both aesthetics and practical performances.

In general, the deep learning can be a solution for topology optimization of complex structures. Doi et al. [20] adopted the deep learning for a multi-objective topology optimization problem. They took two kinds of approaches. One was to apply a neural network, trained with the results of genetic algorithm (GA) using a small population, for the actual optimization using a large population. The other was to apply a neural network, trained with the results of torque performance optimization, for the torque screening of another kind of optimization. These two approaches reduced the number of unnecessary FEA executions. Tan et al. [21] introduced the combined CNN and GAN for the inverse design of microstructural materials. The neural networks took the desired effective compliance tensors as an input and was able to generate porous microstructures corresponding to the tensors.

Preceding studies concerning the deep learning applications for topology optimization have basically approached from the perspective of an image processing based on the CNN. Therefore, the input data of the neural network has the format of digital images. Digital images consist of pixels (or voxels in 3D) and channels that have the color information of pixels. Generally, the shape of each pixel is rectangular and voxels are cuboid. A good strategy to record the information of FE models according to the format of digital images is to match each pixel (or voxel) with a finite element at one-to-one. Most of previous studies used this one-to-one matching input data format. However, the one-to-one matching limits the application of various kinds of finite elements and complicated FE models.

In order to obtain a rapidly converging optimal topology solution, a new method combining the deep learning with a specially designed segmentation scheme for digital images and BESO (bi-directional evolutionary structural optimization) has been developed in this study. Additionally, we devised a unique method that can convert various types of 3D finite elements (FE) including the hexahedron and complex FE models, which

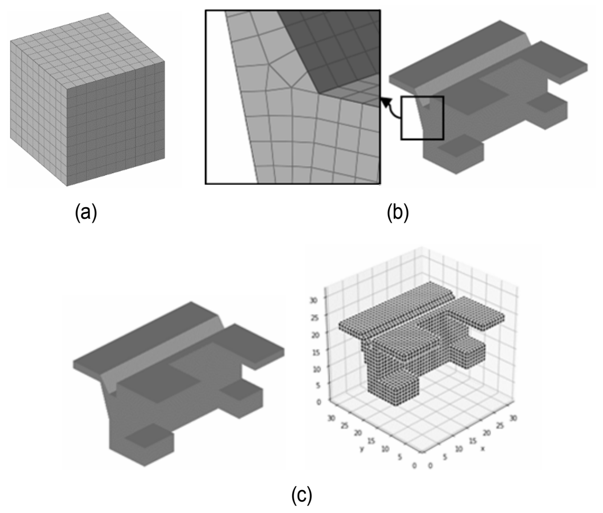

Fig. 1. (a) Voxel and finite elements matched one-to-one in the previous studies; (b) an example for the capability to accommodate both (c) the different types of elements and complicated geometrical shapes of FE models; (c) the input format and voxel (or pixel in 2D) image for the enhanced application of the deep learning for topology optimization.

could not be dealt with in previous studies where the pixels and finite elements of the digital image were matched simply one-on-one, into input/output data of neural networks (see Fig. 1). The details are described in Figs. 4-6 and Sec. 3.2 later in the text.

Fig. 2 shows the flow chart of the deep learning-based topology optimization used in this study. The neural network is added from the early stage of topology optimization and the neural network already trained with 3200 known optimum topologies is loaded on the AI procedure. Along with some procedures related to the conversion of FE-model information to digital images, the neural network predicts the potential optimum topology candidate that is provided as an initial domain in the subsequent topology optimization procedure. This new combined approach of deep learning and topology optimization is proven to be effective in reducing the iteration numbers in the optimization with several examples.

2. Topology optimization

2.1 Bi-directional evolutionary structural optimization

The algorithm of topology optimization in this study is based on the BESO method with a “soft-kill” material interpolation scheme. BESO is a gradient-based optimization method with convergent and mesh-independent algorithm [22]. The basic idea is to remove materials in a low stress region and add materials in a high stress region. The concept of the evolutionary structural optimization (ESO) method, the predecessor of the BESO method, is to remove gradually the inefficient elements in a specified order in the design domain [23]. However, once elements are removed in the ESO, the removed elements are never restored in the subsequent optimization process. As a

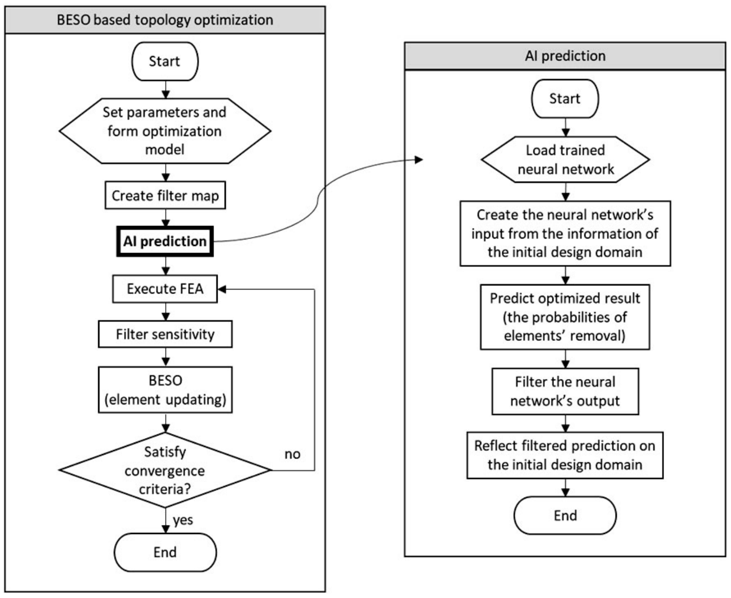
BESO based topology optimization
Start
Set parameters and form optimization model
Create filter map
AI prediction
Execute FEA
Filter sensitivity
BESO (element updating)
Satisfy convergence criteria?
no
yes
End
AI prediction
Start
Load trained neural network
Create the neural network's input from the information of the initial design domain
Predict optimized result (the probabilities of elements' removal)
Filter the neural network's output
Reflect filtered prediction on the initial design domain
End

Fig. 2. A full outline of the AI-based topology optimization. The AI phase provides information on the initial design domain necessary for next topology optimization such as finite elements and boundary conditions represented in voxel images.

result, the important changes in the removed elements in the structural model cannot be reflected in the subsequent optimization process. Therefore, the solution by the ESO method cannot ensure an absolute optimum.

The BESO method overcomes the deficiency in the ESO method by means of allowing elements to be removed and added simultaneously [24]. BESO method can be easily implemented as a post-processor to commercial finite element analysis software and the resulting optimal design gives a clear topology contour without grey area.

For the implementation of the BESO method, a Python code coupled with Nastran has been developed in this study. In the code, finite element analyses are executed by the Nastran. This was possible by adopting pyNastran, a Python library for handling both the input and output files of Nastran.

2.2 Problem formulation

The topology optimization problem here is stated as the compliance minimization of structures with a volume constraint. It can be described mathematically as follows [22, 25],

$$\underset{\mathbf{x}}{\text{minimize}} : C(\mathbf{X}) = \frac{1}{2}\mathbf{F}^T\mathbf{U} = \frac{1}{2}\mathbf{U}^T\mathbf{K}\mathbf{U} \tag{1}$$

$$\text{subject to} : \mathbf{X} = \{x_e\}, x_e = 1 \; or \; x_{\min} \quad \forall e = 1,\ldots,N \tag{2}$$

$$\mathbf{F} = \mathbf{K}\mathbf{U} \tag{3}$$

$$V(\mathbf{X}) = \sum_{\mathbf{x}} x_e v_e = V^* \tag{4}$$

where the compliance C is an objective function, $\boldsymbol{X}$ is the vector of relative elemental densities, x_e is the eth design variable with a value of either 1 for presence or xmin (0.001 in this paper) for absence, N is the total number of elements, $\boldsymbol{F}$ and $\boldsymbol{U}$ are the global force vector and displacement vector, respectively, $\boldsymbol{K}$ is the global stiffness matrix, $\boldsymbol{V}$ is the total volume of the structure with v_e being the elemental volume, $\boldsymbol{V^*}$ is the imposed value of the volume constraint. The constraint defined in Eq. (3) ensures the equilibrium of the structure.

2.3 Filter scheme and sensitivity with BESO

The BESO algorithm is used to solve the above optimization formulation. The gradient of the objective function C regarding to each element density is the sensitivity of each element and the density design variables are updated depending on the element sensitivity, a_e, which can be calculated based on linear elasticity and the SIMP material model and expressed as [22, 26],

$$\alpha_e = \frac{\partial C}{\partial x_e} = -\frac{1}{2} p x_e^{p-1} \mathbf{u}_e^T \mathbf{K}_0 \mathbf{u}_e = -\frac{p}{2x_e} x_e^p \mathbf{u}_e^T \mathbf{K}_0 \mathbf{u}_e = -\frac{p}{2}\frac{E_e}{x_e} \tag{5}$$

where p is the penalty exponent [26], u_e is the element displacement vector and k_0 is the element stiffness matrix of a survived element (i.e. $x_e = 1$). In this equation, the element strain energy, $x_e^p \mathbf{u}_e^T \mathbf{K}_0 \mathbf{u}_e$, is defined as E_e and can be directly calculated from finite element analysis during topology optimization. For solid-void designs, the element sensitivity is used to determine the presence or absence of elements constituting a structure during the element updating process.

The filtering scheme for the element sensitivity shown in Ref. [25] is applied here to avoid numerical instabilities such as checkerboard patterns and mesh-dependency, as follows,

$$\tilde{\alpha}_e = \frac{\sum_i^m w(r_{ei})\alpha_i}{\sum_i^m w(r_{ei})} = \sum_i^m \mu_i \alpha_i \tag{6}$$

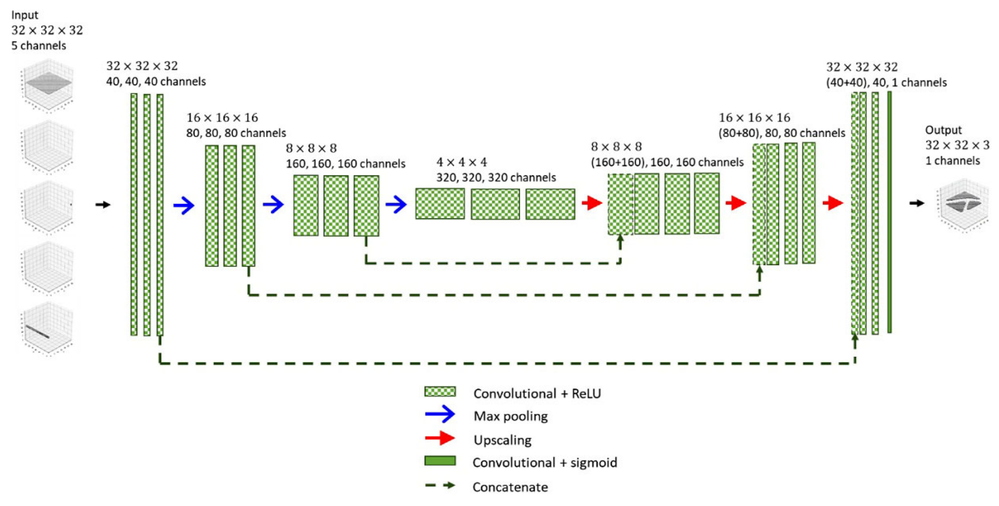

Fig. 3. The neural network used for this study.

$$w(r_{ei}) = \max(0, r_{\min} - r_{ei}) \tag{7}$$

$$\bar{\alpha}_e = \frac{\tilde{\alpha}_e^k + \tilde{\alpha}_e^{k-1}}{2} \tag{8}$$

where m is the total number of nodes in a sub-domain; $w(r_{ei})$ is a weight factor described in Eq. (7); rei is the distance between the centers of elements e and i; r_{min} is a filter radius. The weight (μ_i) is independent of the sensitivity values and computed in advance. Since the BESO method using only $\tilde{a}_e$ in Eq. (6) is difficult to converge, the averaged value of the sensitivity of the current and previous steps, as shown in Eq. (8), is used to secure a convergent solution. k in Eq. (8) denotes the current iteration number.

The sensitivity (*i.e.*, $\tilde{a}_e$) of solid and void elements is equal to the element strain energy and 0, respectively. Void elements are completely deleted from the design domain except where the removal causes the singularity problems in a global stiffness matrix. The BESO method starts from the full design domain and iteratively reduces the design volume by switching elements to solid or void. The next target volume, V^{k+1}, is updated iteratively using the current volume, V^k, and an evolutionary ratio (e_r, 2 % here), as follows:

$$V^{k+1} = V^k(1 \pm e_r) \tag{9}$$

when the element having a value greater than the reference sensitivity value is changed to void, and the element having a small value to soild. At the same time, when the target volume is achieved, the value is set as the reference sensitivity criterion by a bisection method.

3. Topology optimization with the deep learning

3.1 Neural network architecture

Artificial neural networks (ANNs) are a type of function constructed with layers of artificial neurons. In this study, the neural network, as described in Fig. 3, is a function that feeds the information on the initial design domain necessary for further topology optimization such as finite elements and boundary conditions represented in voxel images and then outputs the necessary images containing information of elements removal in the corresponding optimum topology.

The neural network architecture as in Fig. 3 was constructed for this study, based on U-Net [28] and the similar concept [11] in which topology optimization problems are considered as an image segmentation task. The U-Net architecture consists of two paths. One path is to capture the context along with gradual contraction of image size. The other is a symmetric expanding path that extract segmentation masks from the outputs of the contracting path. By concatenating each contracting path's convolutional layer with corresponding level of expanding path's convolutional layer, the U-Net architecture is able to execute more precise localization than any other neural network architectures in image segmentation.

In Fig. 3, the size of images and the number of channels are described on the top of each level of layers. Checkered pattern bars represent the convolutional layer having a 3×3×3 filter size with the ReLU activation function. The right end bar without the pattern is an output convolutional layer possessing a 1×1×1 filter size with the sigmoid activation function.

3.2 Input/output generation scheme

The neural network here takes necessary input and output data in the form of images. In this study, we devised a unique method that can convert various types of finite elements (FE) and complex FE shapes, which could not be dealt with in previous studies where the pixels and finite elements of the digital image were matched one-on-one, into input/output data of neural networks. The information of the FE model is assigned to each channel where RGB color information of digital images is stored originally. Digital images consist of 2D pixels (or 3D

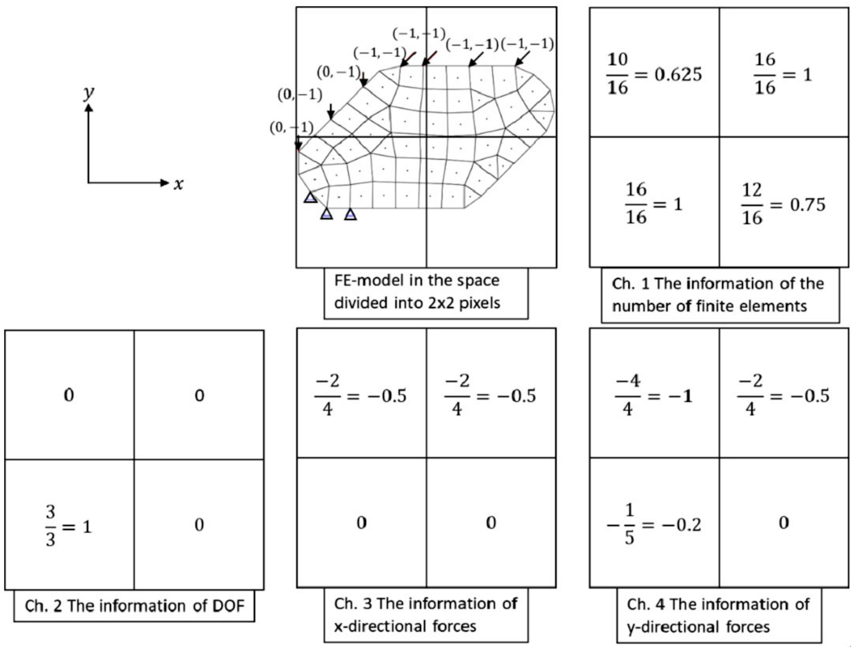

Fig. 4. An example of input data for the neural network is illustrated for 2D image data that has 2×2 pixels with four channels. The arrows and triangles on the FE-model represent force vectors and fixed DOFs, respectively.

voxels) and channels that have some information of pixels. In this study, FE-models information is stored into the channels of input image data. Specific information stored in the input data includes the number of finite elements, boundary conditions, and the force components in *x, y* and *z* directions in the corresponding voxel.

Although this study is focused on 3D, just for easy explanation of the new concept used in this study. Fig. 4 illustrates an example of input data that are represented as 2D images. First of all, a unit space in a square is created with the side length equal to the longest of *x* or *y* coordinate of the FE model and the FE-model is placed in the middle of the space. Secondly, the space is divided into 2×2 pixels. In the first channel of the input data, the number of the finite elements belonging to each pixel are stored. Whether a pixel includes a finite element or not depends on the location of the centroid of each element. In Fig. 4, the number of the elements in each pixel is divided by the largest number in the channel one. This is like a normalization procedure which is necessary to stabilize and make the training of the neural network faster. In the second channel, the number of fixed DOFs in each pixel is stored and the normalization is also applied. In the third and fourth channels, the magnitudes of *x*- and *y*-directional forces in each pixel are stored. The normalization for these channels is applied but in different way. In that case, the divisor is the largest absolute value in the third and fourth channels. Note that in the actual practice of the study with 3D cases, a space is divided into 32×32×32 voxels and the input image has one additional channel for storing the information of z-directional force component.

For simplicity, let's consider the case that a target volume fraction is 0.5; Poisson's ratio is 0.3; a filter radius is 1/20 of the longest of *x*, *y*, or *z* coordinate length of the design domain. There are already some previous studies considering the variation of these scalar factors. A simple way to store the scalar factors into an input image is adding more channels [13, 18]. However, filling up a channel with the same scalar value for a factor having no spatial information is computationally inefficient. More efficient way is to input a scalar value directly into the layer of latent variables [17]. Fig. 5 shows an example of output data represented in 2D for convenience. The procedure that locates the FE model in an image space is the same as that of input data. The output image has only one channel. In the channel, there is each pixel holds information about the number of elements removed in the corresponding optimum topology. After normalization, each pixel value in the channel can be interpreted as the probability of element removal in the pixel.

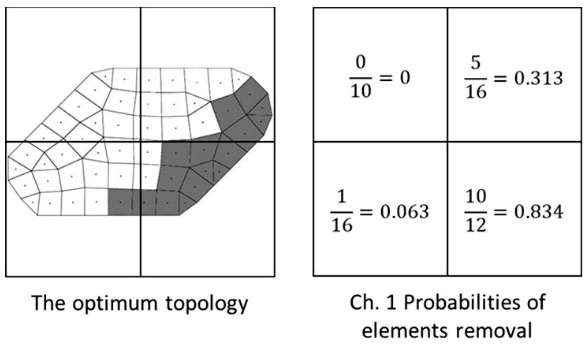

Fig. 5. An example of output data for the neural network for 2D image data that has 2×2 pixels with one channel. The removed FEs are represented in black.

It must be considered for the generation of output images that not only the creation but also the reflection of an image onto the design domain. Fig. 6 shows an example of the reflection procedure. Fig. 6(a) shows that a value of pixel is assigned to each FE. Some FEs are removed from the design domain in the order of higher assigned values closer to a preset target volume fraction and the results could be somewhat discontinuous.

For better results, a post-processing procedure is added after assigning pixel values to FEs. The procedure is inspired from the mesh-independent filtering of the density-based topology optimization [25, 27]. A filtering scheme depicted in Eqs. (6)-(8) is applied. Fig. 6(b) shows that FEs in the design domain have more continuously varying pixel values after post-processing and the isolated FEs that are removable can be relocated. Fig. 7 shows that filtering enhances the resolution of the reflection result.

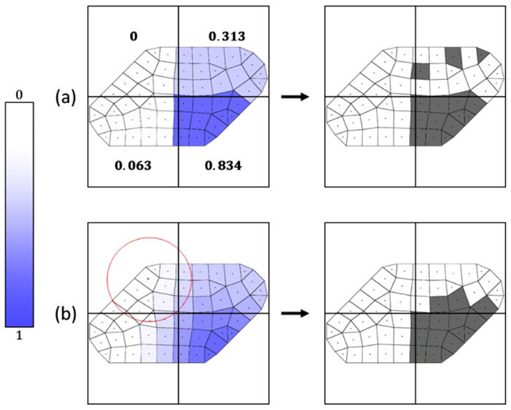

Fig. 6. Example of the procedure that reflects the output image onto the design domain. The shade gradients represent the value assigned to each FE: (a) FEs have assigned values of pixel in their specific region without filtering; (b) assigned values are filtered. The dark black FEs would be removed from the design domain.

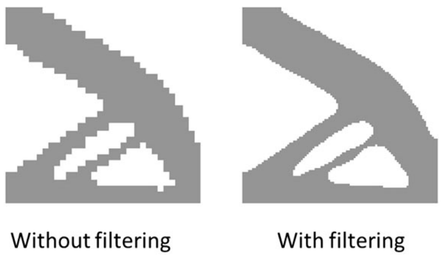

Fig. 7. Comparison of design changes obtained with and without filtering. The post-processing for the raw output values enhances the resolution of the reflection result.

3.3 Training dataset

Three different types of the design domain are studied for the training dataset as shown in Fig. 8. There are 1700 datasets for a 2D cantilevered square plate with 100×100×1 shell elements, 900 datasets for a 3D cantilevered beam with 40×25×25 hexahedral elements, and 600 datasets for a 3D cantilevered one with 60×25×25 hexahedral elements. For each training data related with the design domain of the 2D square plate, a concentrated force is applied in any direction on the top edge. For each cantilever data in the two types of 3D design domains, a randomly-oriented concentrated force is applied on any surface. The BESO-based topology optimization is conducted for these design domains with fixed boundary conditions. The total number of samples in the obtained training dataset is 3200 and only 320 (i.e., 10 %) samples are utilized as a validation dataset.

In order to increase the diversity of the acquired training dataset, two kinds of data augmentations are applied. One is the random rotation augmentation, as shown in Fig. 9. The images in the training dataset are randomly rotated about x, y or z axis through 0, 90, 180 or 270 degrees. The other is the random flip augmentation. Images can be randomly flipped

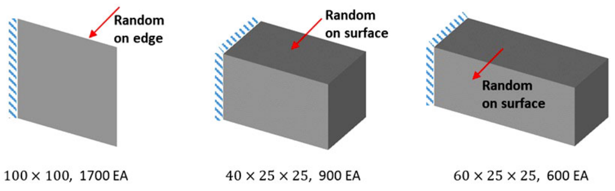

Fig. 8. Three types of cantilevered FE-models used for generating a training dataset for validation study subjected to a randomly oriented concentrated force.

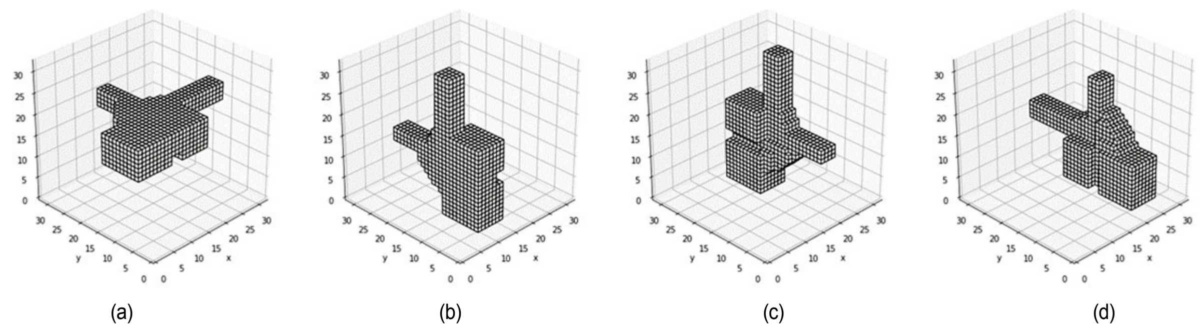

Fig. 9. Examples of random rotation augmentation. Each image in the training dataset randomly rotates for x, y or z axis while the neural network is trained: (a) initial position; (b) 90° rotation about the y-axis from (a); (c) 180° rotation about the z-axis from (b); (d) 90° rotation about the x-axis from (c).

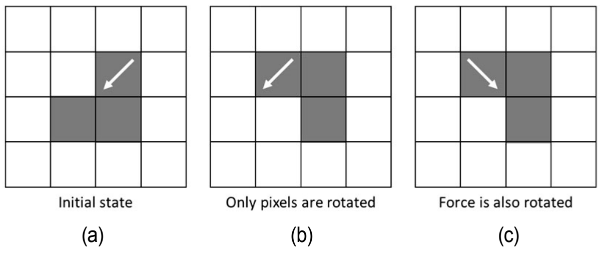

Fig. 10. An example of 90° rotation of an input image in 2D. When the input image rotates or flips, force as well as pixel move together and are recorded in the pixel channels.

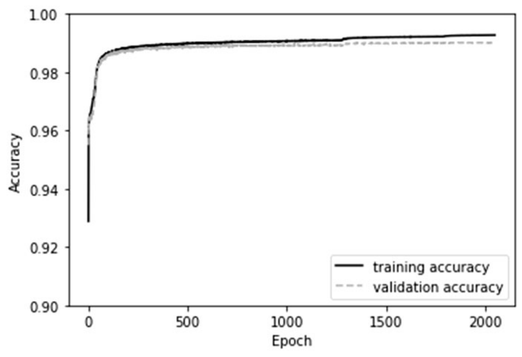

Fig. 11. Graph shows the accuracy change in voxel-wise prediction as training progresses.

through about x, y, or z axis.

It is emphasized in Fig. 10 that the loading should be rotated together with the image pixels. Rotating only pixels is insufficient to realize the rotation of the input image. Because the spin of forces is also required along with pixels, the additional rotation of forces is necessary separately.

In order to train the neural network, the ADAM optimizer is applied to minimize the loss that is a binary cross-entropy. The batch size was 4 and the initial learning rate was 0.0002. The learning rate became half at 1280 epoch and 1/4 at 1920 epoch. The training has finished after 2048 epoch. This hyperparameter setting was chosen because it showed the highest voxel-wise prediction accuracy for both training and validation datasets among some candidate settings. Fig. 11 shows the history of training and validation accuracy in the given setting. At the final epoch, the training accuracy is 0.9925 and the validation accuracy is 0.9902. The training may be thought to be accurate because the gap between two accuracies is small in Fig. 11.

In this suggested neural network, predictions are performed for every voxel value separately. It means total number of predictions executed is 32×32×32. The predictions are usually considered to be correct if the predicted voxel values are within an acceptable limit, approximately ±6×10^{-6} %, from the real voxel values. As an evaluation metric for the training of neural networks, the accuracy is defined as follow,

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{10}$$

4. Results and discussions

To validate the proposed method, two types of structural examples having different finite elements and shapes are demonstrated. One is two 2D square plates modeled with 100×100 shall elements. The other is a 3D cantilever modeled with 40×20×20 hexahedral elements without using the prepared training dataset. The convergence criteria of the BESO method [22] is expressed in Eq. (11) and the same parameters are used in the both 2D and 3D structures.

$$error = \frac{\left| \sum_{i=1}^{N} C_{k-i+1} - \sum_{i=1}^{N} C_{k-N-i+1} \right|}{\sum_{i=1}^{N} C_{k-i+1}} \le \tau \tag{11}$$

where k is the current iteration number, τ is an allowable convergence error, and N is an integer. In this study, τ is 10^{-5} and N is 5. Note that τ used here is relatively smaller than usual 10-3 to 10^{-4}. Even if a small τ requires more iterations to converge, τ = 10^{-5} has been chosen in this study in order to ensure that every automatically created samples of datasets converge to optimum points.

4.1 2D square plate

In the case of the 2D cantilevered plate, many samples of the training dataset can be acquired without difficulty due to its simple geometry and consequently more accurate results were obtained. The convergence performance of the BESO-based topology optimization only is compared to that of the combined deep learning and BESO optimization for the 2D square plate with two different loading points.

The initial design domain of the cantilevered square plate consists of 100×100×1 finite shell elements. All DOFs at one edge are fixed and a concentrated force is applied to the center of the free end, as shown in Fig. 12(a). A target volume fraction is set to be 0.5 and a filter radius is 5. When only BESO topology optimization was performed, the convergence reached after 68 iterations, as shown in Fig. 12(c). On the other hand, when topology optimization was started with initial design domain (Fig. 13(a)) predicted first by the neural network, the optimization converged to the solution after 28 iterations (see Fig. 13(c)). The neural network predicts all details clearly, as shown in Fig. 13(a). In terms of the number of iterations, which is proportional to CPU run time, it is reduced by 59 % compared to when only BESO was used. The objective function, which is structural strain energy, converges to 0.003533 in this case, when optimum design is reached.

As the second example, this time the same concentrated vertical force shown in Fig. 12(a) is applied at the bottom corner of the cantilever plate, as in Fig. 14(a). The same target volume fraction of 0.5 and filter radius of 5 are also set. When topology optimization with BESO only is performed in a normal way, it requires 67 iterations to reach convergence (see Fig.

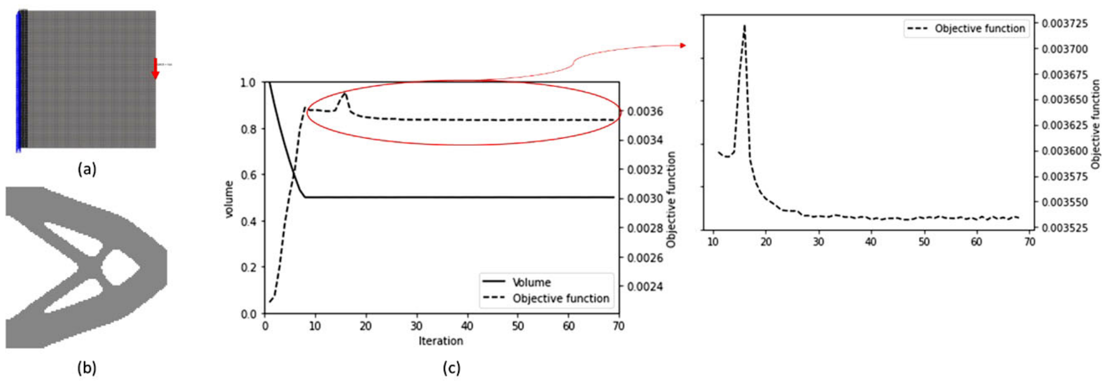

Fig. 12. The optimum solution of the normal BESO-based topology optimization for a 2D cantilevered square plate. The iteration count was 68 for the convergence: (a) an initial design domain with boundary and load conditions; (b) a final optimum topology; (c) plots for convergence history.

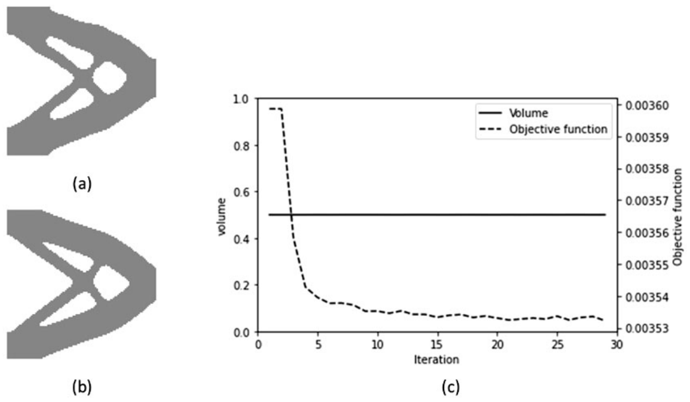

Fig. 13. The optimum solution of the combined deep learning and BESO-based topology optimization for a 2D cantilevered square plate. The iteration count was 28 for the convergence: (a) an initial design domain obtained from the neural network predictions; (b) a final optimum topology; (c) plots for convergence history.

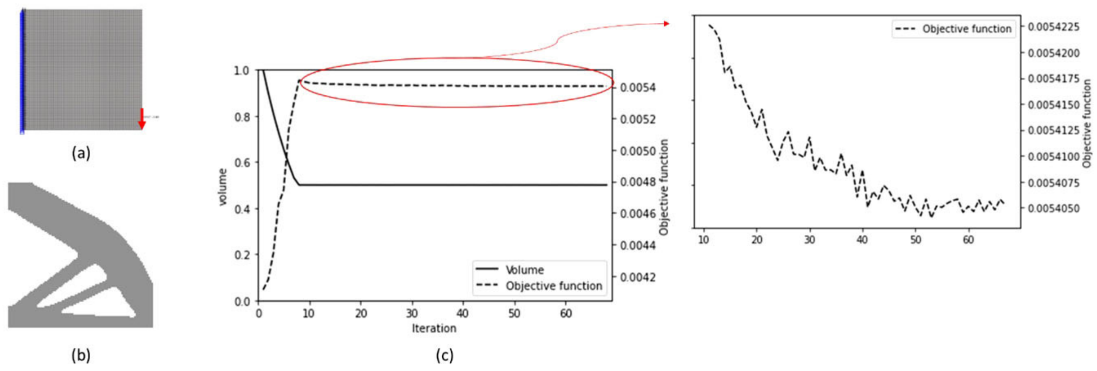

Fig. 14. Conventional BESO-based topology optimization results of the 2D square plate. It takes 67 iterations for the convergence: (a) an initial design domain and boundary conditions; (b) a final optimum topology; (c) plots for convergence history.

14). On the other hand, when it is started with the initial design domain provided by the neural network, it takes only 25 iterations to reach convergence (see Fig. 15). The iteration is reduced by 63% compared to when only BESO was used. The objective function converges to 0.00541 in this case, when optimum design is reached. It can be seen that some errors in the learning process, shown in Fig. 15(a), have been completely corrected.

As the third example, the MBB (Messerschmitt-Bölkow-Blohm) beam problem with 3-1 aspect ratio is solved, as in Fig. 16(a). The same target volume fraction of 0.5 and filter radius of 5 are set. The MBB beam consists of 300×100×1 shell elements. When topology optimization with BESO only is performed in a normal way, it requires 71 iterations to reach convergence. When it is started with the initial design domain provided by the neural network, as in Fig. 16(b), it takes only 20

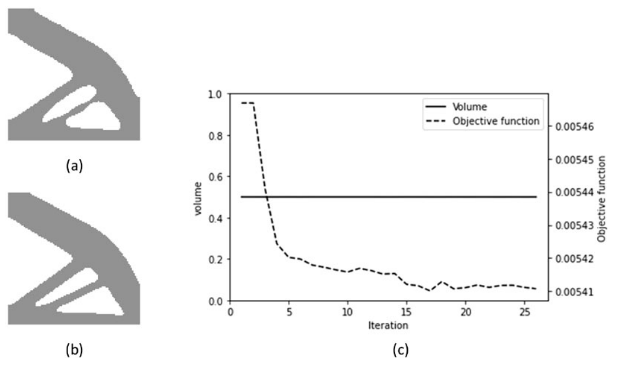

Fig. 15. AI-assisted BESO-based topology optimization results of the same 2D square plate. It takes 25 iterations for the convergence: (a) an initial design domain obtained from the neural network predictions; (b) a final optimum topology; (c) plots for convergence history.

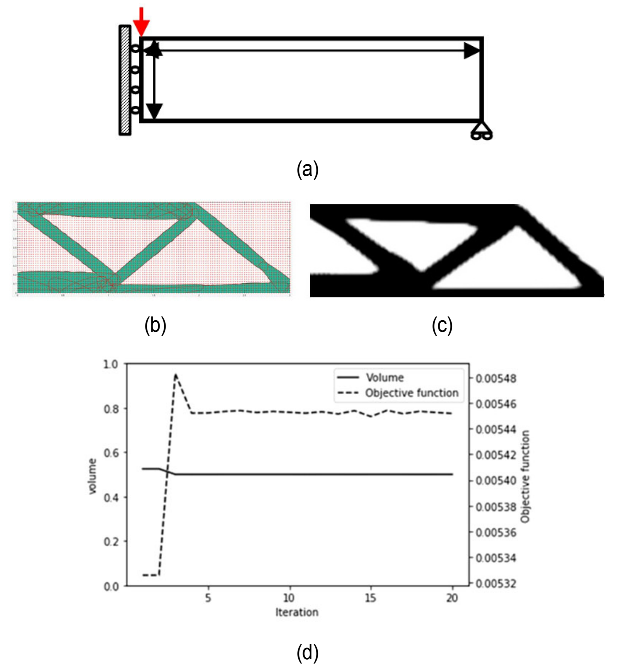

Fig. 16. AI-assisted BESO-based topology optimization results of the MBB beam: (a) it takes 20 iterations for the convergence; (b) an initial design domain obtained from the neural network predictions; (c) a final optimum topology; (d) plots for convergence history.

iterations to reach convergence (see Figs. 16(c) and (d)).

4.2 3D cantilever

The 3D cantilever beam, which consists of 40×20×20 solid finite elements, is illustrated here. This case with 40×20×20 elements is not included in the training dataset. A vertical force is applied to the centroid of the free end. A target volume fraction is set to be 0.5 and a filter radius is 4. Note that this filter radius is twice larger than the filter radius of samples in training dataset. Usually, a larger filter radius results in the optimum topology in a simple structure. When topology optimization was performed with only BESO, it requires 98 iterations to reach

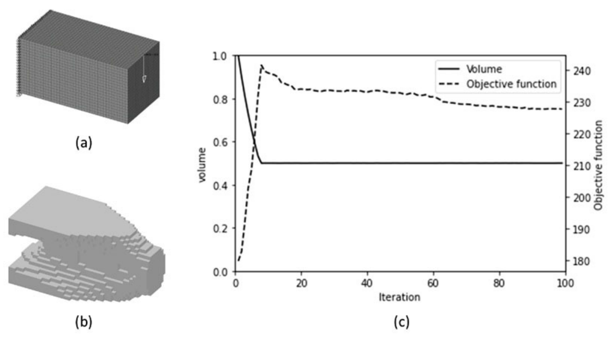

Fig. 17. Conventional BESO-based topology optimization results of a 3D cantilever. It takes 98 iterations for the convergence: (a) an initial design domain with loading; (b) a final optimum topology; (c) a convergence history.

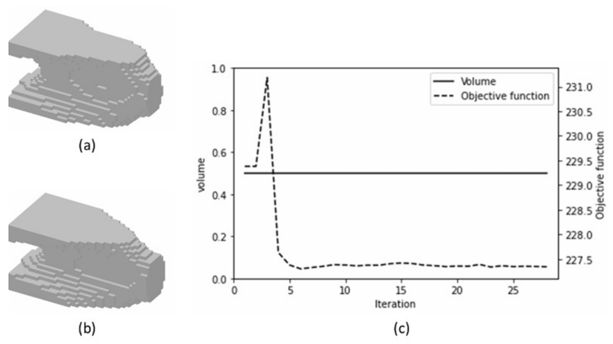

Fig. 18. AI accelerated BESO-based topology optimization results of a 3D cantilever. It takes 27 iterations for the convergence: (a) an initial design domain obtained from neural network predictions; (b) a final optimum topology; (c) a convergence history.

convergence (see Fig. 17). On the other hand, when it was started with the initial design domain provided by deep learning even with excluded beam shape data, it takes only 27 iterations to reach convergence (see Fig. 18). The iteration is reduced by 72.5 % compared to when only BESO was used. As in all other studies, the time consumed by the deep learning process is not considered in the count. With the advancement of neural network algorithms, the time for learning will be significantly reduced.

Since the current neural network can still give many incorrect topology solutions during the learning process, it indicates the need for the subsequent optimization presented in this study. In the proposed method, if the results from the learning process that can be independently performed are prepared in advance, topology optimization can be achieved quickly even if they are not in the dataset. The results of the examples show that the present method which combines the topology optimization and neural networks can predict the optimum solutions well even for new untrained cases in some range without loss of generality.

This study presents a fundamental and common deep learning-based topology optimization methodology. 3200 topologies learned for this paper are included in the training dataset. However, if trained for other 3D structures with different load and boundary conditions such as simply supported beams, torsional problems, etc., this method is applicable and can predict the correct topologies due to its 3D optimization ability with 3D filtering and FE elements.

5. Conclusions

In an effort to reduce the number of iterations in the topology optimization, the deep learning is combined with the BESO method. By dividing the design domain into voxels based on the longest length among the lengths in each axial direction of the design model and storing the information of the FE model in an averaged value, it is possible to convert even a complex shape model into an input for deep learning. Besides, a new post-processing technique has been also developed in order to restore the information on the relative locations among finite elements, that may be lost in large design domains with many finite elements, in the raw outputs generated by the neural network, because the neural network applied here uses a fixed 40×25×25 resolution voxel image as input and output. Since the 2D and 3D examples show that satisfactory optimal topologies can be obtained from as few as 3200 topologies in the training dataset, the novel combined approach has proven effective in reducing the number of iterations by 72.5 % in the 3D example, compared to only BESO result.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2020R1A2C1102742).

References

[1] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn and T. J. Purcell, A survey of general-purpose computation on graphics hardware, *Comput. Graph. Forum*, 26 (1) (2007) 80-113.

[2] A. Coates, B. Huval, T. Wang, D. J. Wu, A. Y. Ng and B. Catanzaro, Deep learning with COTS HPC system, *Proc. the 30th International Conference on Machine Learning*, Georgia, June 17-19, PMLR, USA, 28 (3) (2013) 1337-1345.

[3] WIRED, *Now You Can Build Google's $1M Artificial Brain on the Cheap*, https://www.wired.com/2013/06/andrew-ng/ (2013).

[4] X. Lu, D. G. Giovanis, J. Yvonnet, V. Papadopoulos, F. Deterz and J. Bai, A data-driven computational homogenization method based on neural networks for the nonlinear anisotropic electrical response of graphene/polymer nanocomposites, *Comput. Mech*, 64 (2) (2019) 307-321.

[5] D. Z. Huang, K. Xu, X. Farhat and E. Darve, Learing constitutive relations from indirect observations using deep neural networks, *J. Comput. Phys.*, 416 (2020) 109491.

[6] A. Khan, D. K. Ko, C. S. Lim and H. S. Kim, Structural vibration-based classification and prediction of delamination in smart composite laminate using deep learning neural network, *Composites Part B: Eng.*, 161 (2019) 586-594.

[7] G. Toh and J. Park, Review of vibration-based structural health monitoring using deep learning, *Appl. Sci.*, 10 (5) (2020) 1680.

[8] J. Francis and L. Bian, Deep learning for distortion prediction in laser-based additive manufacturing using big data, *Manuf. Lett.*, 20 (2019) 10-14.

[9] B. Zhang, S. Liu and Y. C. Shin, In-process monitoring of porosity during laser additive manufacturing process, *Addit. Manuf.*, 28 (2019) 497-505.

[10] E. Ulu, R. Zhang and L. B. Kara, A data-driven investigation and estimation of optimal topologies under variable loading configurations, *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.*, 4 (2) (2016) 61-72.

[11] I. Sosnovik and I. Oseledets, Neural network for topology optimization, *Russ. J. Numer. Anal. M.*, 34 (4) (2019) 215-223.

[12] S. Banga, H. Gehani, S. Bhilare, S. J. Patel and L. B. Kara, 3D topology optimization using convolutional neural networks, *arXiv:1808.07440* (2018).

[13] Y. Zhang, B. Peng, X. Zhou, C. Xiang and D. Wang, A deep convolutional neural network for topology optimization with strong generalization ability, *arXiv: 1901.07761* (2019).

[14] J. H. Song and C. Kim, 3-D topology optimization based on nodal density of divided sub-elements for enhanced surface representation, *Int. J. Precis. Eng. Man.*, 13 (4) (2012) 557-563.

[15] Q. X. Lieu and J. Lee, A multi-resolution approach for multi-material topology optimization based on isogeometric analysis, *Comput. Methods Appl. Mech Eng.*, 323 (2017) 272-302.

[16] L. Xue, J. Liu, G. Wen and H. Wang, An efficient and high-resolution topology optimization method based on convolutional neural networks, *arXiv:2001.04350* (2019).

[17] Y. Yu, T. Hur, J. Jung and I. G. Jang, Deep learning for determining a near-optimal topological design without any iteration, *Struct. Multidisc. Opti.*, 59 (3) (2019) 787-799.

[18] B. Li, C. Huang, X. Li, S. Zheng and J. Hong, Non-iterative structural topology optimization using deep learning, *Comput. Aided Design*, 115 (2019) 172-180.

[19] S. Oh, Y. Jung, S. Kim, I. Lee and N. Kang, Deep generative design: Integration of topology optimization and generative models, *J. Mech. Design*, 141 (11) (2019).

[20] S. Doi, H. Sasaki and H. Igarashi, Multi-objective topology optimization of rotating machines using deep learning, *IEEE Transactions on Magnetics*, 55 (6) (2019) 1-5.

[21] R. K. Tan, N. L. Zhang and W. Ye, A deep learning based method for the design of microstructural materials, *Struct. Multidisc. Opti.*, 61 (2020) 1417-1438.

[22] X. Huang and Y. M. Xie, Bi-directional evolutionary topology optimization of continuum structures with one or multiple materials, *Comput. Mech.*, 43 (3) (2009) 393-401.

[23] Y. M. Xie and G. P. Steven, A simple evolutionary procedure for structural optimization, *Comput. Struct.*, 49 (5) (1993) 885-896.

[24] X. Huang and Y. M. Xie, *Evolutionary Topology Optimization of Continuum Structures: Methods and Application*, John Wiley & Sons, UK (2010).

[25] Z. H. Zuo and Y. M. Xie, A simple and compact Python code

for complex 3D topology optimization, *Adv. Eng. Soft.*, 85 (2015) 1-11.

[26] M. P. Bendsøe and O. Sigmund, *Topology Optimization: Theory, methods, and Applications*, Springer-Verlag, Berlin, Germany (2003).

[27] O. Sigmund and J. Petersson, Numerical instabilities in topology optimization: a survey on procedures dealing with checkboards, mesh-dependencies and local minima, *Structural Optimization*, 16 (1) (1998) 68-75.

[28] O. Ronneberger, P. Fischer and T. Brox, U-Net: Convolutional networks for biomedical image segmentation, In: N. Navab, J. Hornegger, W. Wells and A. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015, MICCAI 2015, Lecture Notes in Computer Science*, 9351 (2015).

Cheol Kim's research interests include design optimization, mechanics of composite materials and Li-ion battery materials. He received Ph.D. from the University of Illinois at Urbana-Champaign.