Received October 5, 2021, accepted October 25, 2021, date of publication November 2, 2021, date of current version November 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3125014

Unit Module-Based Convergence Acceleration for Topology Optimization Using the Spatiotemporal Deep Neural Network

YOUNGHWAN JOO[1], YONGGYUN YU[2], AND IN GWUN JANG[3], (Member, IEEE)
[1]Korea Institute of Energy Research, Yuseong-gu, Daejeon 34129, Republic of Korea
[2]Korea Atomic Energy Research Institute, Yuseong-gu, Daejeon 34057, Republic of Korea
[3]Korea Advanced Institute of Science and Technology, Yuseong-gu, Daejeon 34141, Republic of Korea

Corresponding author: In Gwun Jang (igjang@kaist.edu)

This work was supported in part by the Korea Atomic Energy Research Institute (KAERI) Research and Development Program under Grant KAERI-524450-21, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant NRF-2018R1C1B6005157.

ABSTRACT This study proposes a unit module-based acceleration method for 2-D topology optimization. For the purpose, the first-stage topology optimization is performed until the predefined iteration. After a whole design domain is divided into a set of unit modules, information on the spatiotemporal characteristics of intermediate designs and a filtering radius is used to separately predict a near-optimal design of each unit module through a trained long short-term memory (convLSTM) network. Then, in the second-stage topology optimization, a combined near-optimal design of a whole design domain is used as an initial design to determine the optimized design in a more efficient way. To train a convLSTM network, a history of intermediate designs is obtained under a randomly generated boundary condition of a unit module. The filtering radius is also used as the training data to reflect the geometric features affected by a filtering process. For four examples with different design domains and boundary conditions, the proposed method successfully provides the accelerated convergence up to 6.09 with a negligible loss of accuracy less than 1.12% error. These numerical results also demonstrate that the proposed unit module-based approach achieves a scalable convergence acceleration at a design domain of an arbitrary size (or resolution).

INDEX TERMS Convergence acceleration, deep learning, finite element method, structural topology optimization.

I. INTRODUCTION

Topology optimization [1] is a design method that determines the optimal layout of a structure under a given boundary condition. It offers a conceptual design by iteratively updating a material distribution to extremize an objective function while satisfying the constraint functions. Because each iteration typically requires finite element (FE) analysis and design sensitivity calculation, the computing cost of topology optimization increases according to the number of finite elements and/or iterations. Thus, it is significantly important to reduce the computational cost of topology optimization for practical use.

Numerous researchers have attempted to solve this computing issue [2]. For instance, Jang and Kwak [3] proposed a design space adjustment technique that enables a change of design domain during topology optimization. Kim *et al.* [4] proposed an efficient convergence criterion to reduce the computing cost by gradually excluding the design variables that are determined to converge. Liao *et al.* [5] simultaneously implemented a multilevel mesh, an initial value-based preconditioned conjugate gradient, and a local update strategy to achieve faster convergence for large-scale problems. Zheng *et al.* [6] reduced the degrees of freedom from the FE equations, thereby accelerating the convergence. Another representative approach for improving computational efficiency is the multi-resolution topology optimization (MTOP) proposed by Nguyen *et al.* [7], [8]. Yoo *et al.* [9] also proposed an adaptive isosurface variable grouping (aIVG) method to enhance the computational efficiency of the MTOP.

Recent advances in machine learning (ML) techniques have brought an increasing attention to their engineering applications [10]–[12], and [13]. Several authors have

The associate editor coordinating the review of this manuscript and approving it for publication was Varuna De Silva.

applied ML techniques to reduce the computational cost of topology optimization [14], [15]. Aulig and Olhofer [16] developed a neuro-evolutionary topology optimization that substitutes the analytic design sensitivity with an artificial neural network. Liu *et al.* [17] applied an unsupervised machine learning algorithm to the clustering procedure for nonlinear multi-material topology optimization. Sosnovik and Oseledets [18] implemented an autoencoder network based on convolutional neural network (CNN) to predict a 0-1 binary image from an intermediate design during optimization. This strategy outperformed the conventional thresholding method where intermediate density values (neither 0 nor 1) are changed to 0 or 1 based on a predetermined threshold value. Yu *et al.* [19] proposed a deep learning-based method to predict a near-optimal topological design using a CNN-based encoder-decoder network and generative adversarial network. Instead of conducting finite element analysis, Sasaki and Igarashi [20] applied a CNN-based model to evaluate the performance of an intermediate design at each iteration. Qian and Ye [21] utilized artificial neural networks as surrogate models for sensitivity prediction.

As aforementioned, deep learning-based approaches typically uses a method of inferring a near-optimal solution to reduce the number of iterations during topology optimization. Therefore, they provide only an approximate solution rather than an exact solution. To tackle this issue, Kallioras *et al.* [22] recently applied an ML-based acceleration method that infers a near-optimal design from the optimization results obtained from previous iterations and then resumes the topology optimization from the inferred design. Although a local density variable is spatially influenced by neighboring density variables through the filtering process, only an iterative variation of each density variable was considered as time-series data. To fully investigate an iterative change of intermediate designs in topology optimization, both the spatial and time-series (i.e., spatiotemporal) characteristics of intermediate designs should be considered in a more rigorous way. For example, the convolutional long short-term memory (convLSTM) is a representative network to handle the spatiotemporal data [23]. This network incorporates the features of the CNN [24] and the long short-term memory (LSTM) [25]. The CNN and LSTM parts of the network capture the spatial and time-series characteristics, respectively, of data. The convLSTM has been actively used in the field where sequential image data are processed. Pfeuffer and Dietmayer [26] utilized the convLSTM for fast video segmentation required for self-driving cars which need to process a large amount of video frame data. Arbellel and Raviv [27] proposed microscopy cell segmentation method using the convLSTM. They successfully captured complicated spatial and temporal behaviors from live cell microscopy sequences. Because intermediate designs during topology optimization can be regarded as sequential image data like video frames (simply, the iteration number can be considered as time), it is possible to implement the convLSTM to investigate a design history during topology optimization and thereby predict a density distribution at a near optimum.

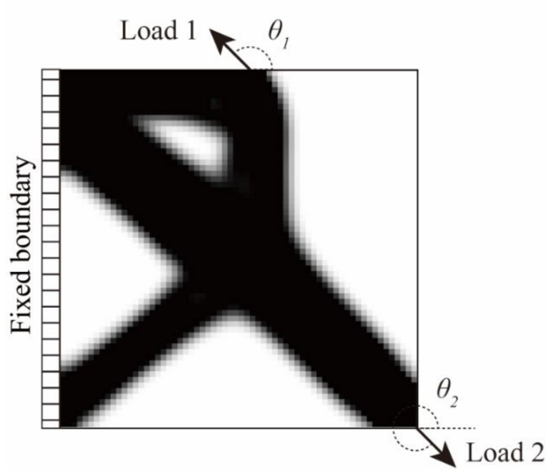


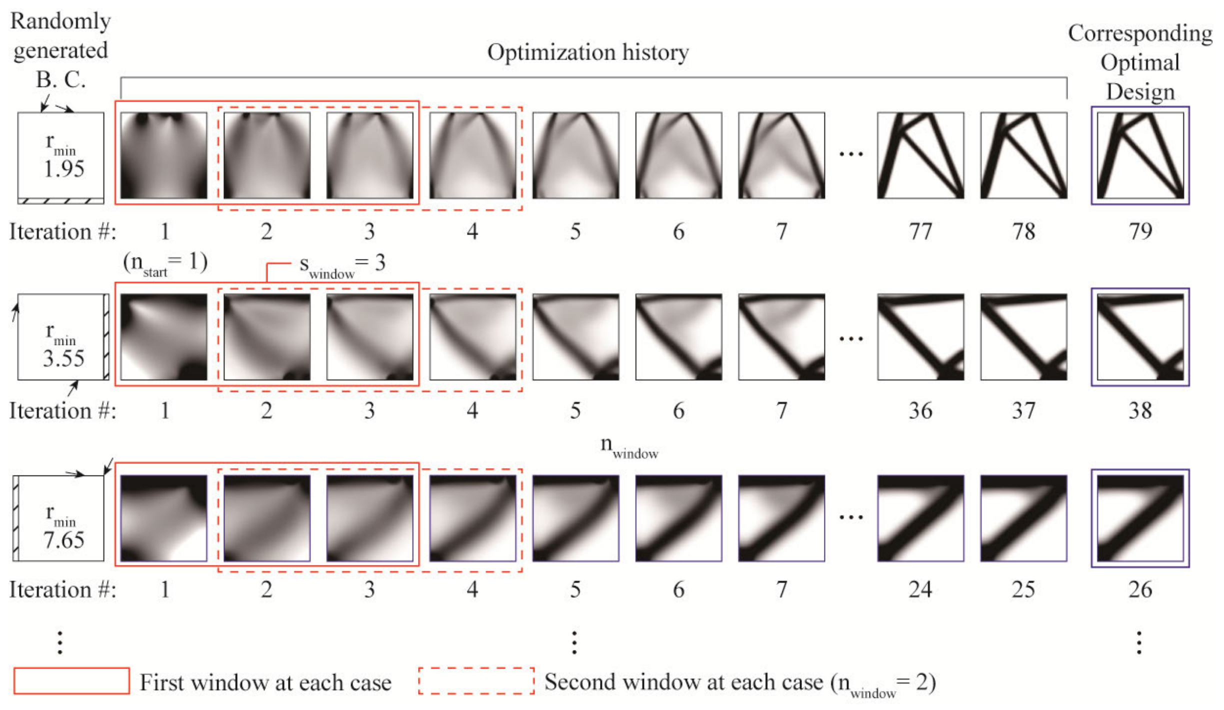
FIGURE 1. Design domain and boundary conditions of a unit module used in this study.

To accelerate the overall convergence in topology optimization, this study proposes a spatiotemporal deep learning model (convLSTM-based network) that can consider the history of density distribution at each iteration as a type of video frame data. This acceleration model was applied to the conventional solid isotropic material with penalization (SIMP) method, in which a 64×64 resolution unit module was considered. Note that a design domain of any arbitrary resolution can be divided into a set of unit modules. To train a deep learning network, intermediate designs of a unit module were obtained under randomly generated boundary conditions. The radius of sensitivity filtering was also considered as the training data to cope with the geometric features of intermediate designs affected by the filtering process. Then, the proposed method was verified by performing an additional 200 cases of topology optimization which were not included in the training data set. Finally, the proposed method was applied to well-known 2-D topology optimization examples to evaluate the scalability and mesh independence of the proposed method.

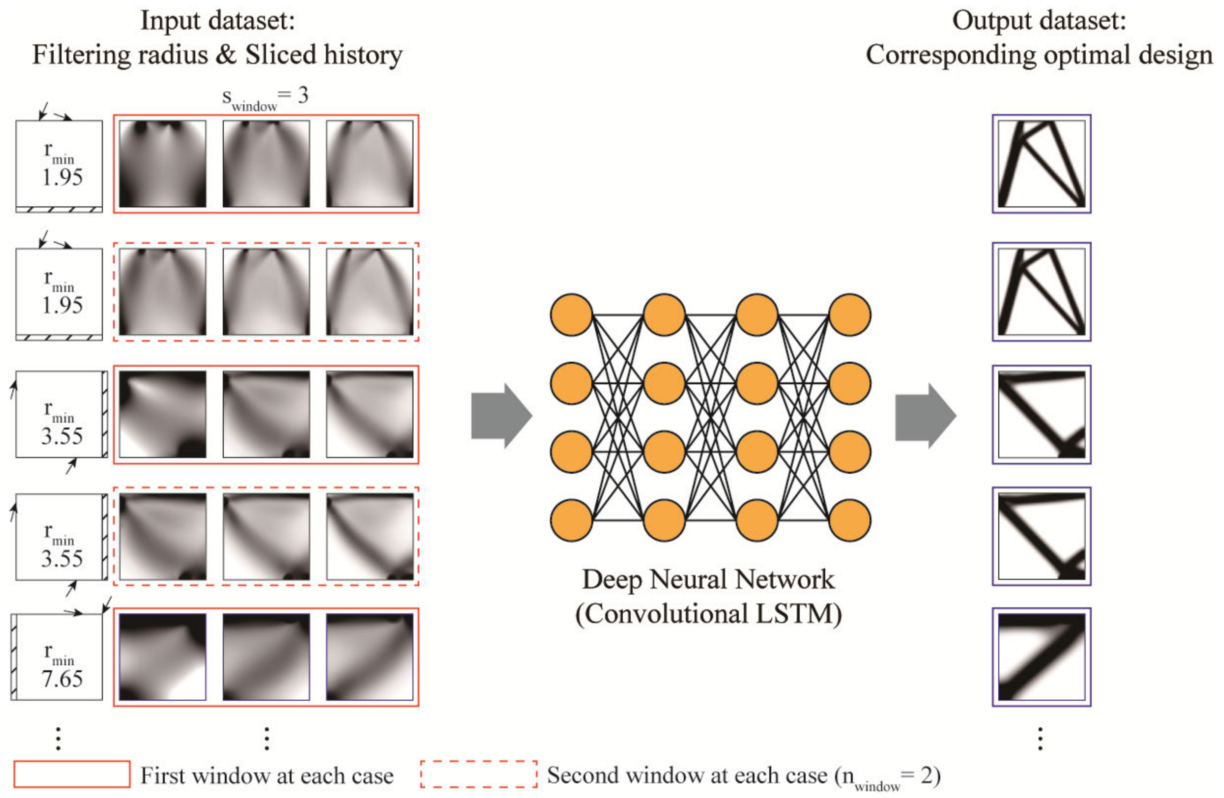
II. METHOD

A. PROBLEM DEFINITION

As the first step towards topology optimization through convergence acceleration, this study considered only 2-D compliance minimization among various types of topology optimization problems. To achieve a scalable convergence acceleration at a domain with an arbitrary resolution (or the number of finite elements), this study defines a unit module of 64×64 finite elements. Figure 1 shows an example of the optimized design of a unit module with its own boundary condition. A fixed boundary condition is imposed on one of the four sides, randomly selected. Two concentrated loads of magnitude 1 are applied at random points on the boundary except for the fixed side. The direction of these loads was randomly chosen between 0–360°. All random samples


Randomly generated B. C.
Optimization history
Corresponding Optimal Design
r_{min} 1.95
r_{min} 3.55
r_{min} 7.65
Iteration #: 1 2 3 4 5 6 7 ... 77 78 79
(n_{start} = 1)
s_{window} = 3
Iteration #: 1 2 3 4 5 6 7 ... 36 37 38
n_{window}
Iteration #: 1 2 3 4 5 6 7 ... 24 25 26
First window at each case
Second window at each case (n_{window} = 2)

(a)


Input dataset: Filtering radius & Sliced history
Output dataset: Corresponding optimal design
s_{window} = 3
r_{min} 1.95
r_{min} 1.95
r_{min} 3.55
r_{min} 3.55
r_{min} 7.65
Deep Neural Network (Convolutional LSTM)
First window at each case
Second window at each case (n_{window} = 2)

(b)

FIGURE 2. Conceptual diagram of (a) constructing training data from each optimization case (s_{window} = 3, n_{start} = 1, and n_{window} = 2) and (b) training a deep neural network.

were selected from a uniform distribution. Ideally, a random concentrated load can be considered at every node on the boundary of a unit module to include general designs that are encountered in topology optimization. This is conceptually

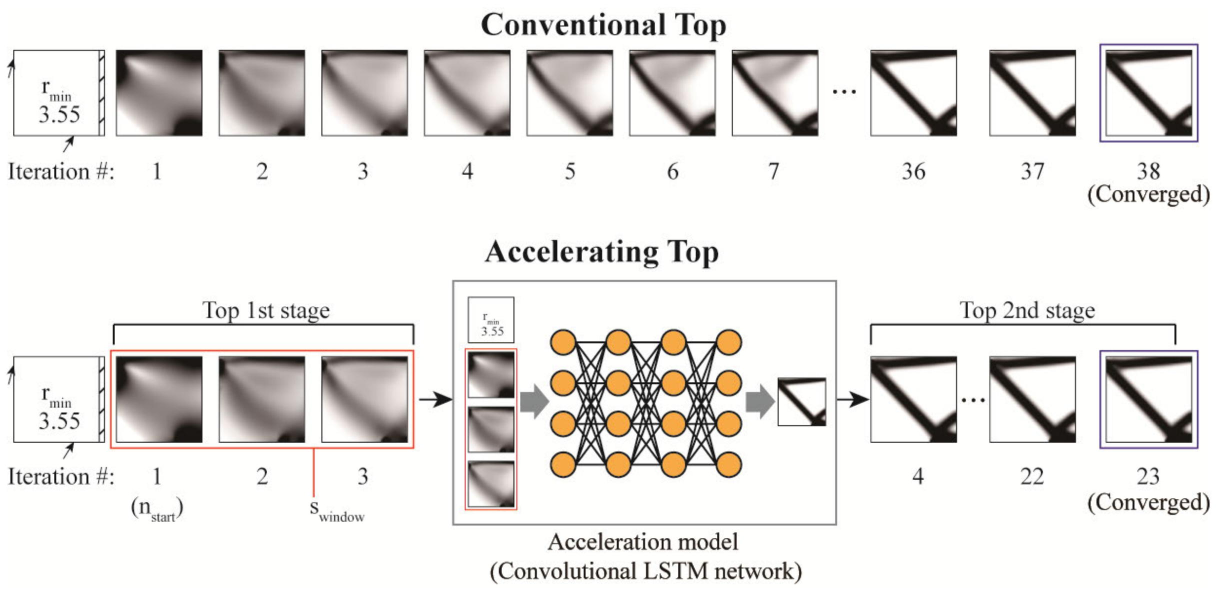

Conventional Top
r min 3.55
Iteration #: 1 2 3 4 5 6 7 … 36 37 38 (Converged)
Accelerating Top
Top 1st stage
Top 2nd stage
r min 3.55
Iteration #: 1 2 3 4 22 23 (Converged)
(n start)
s window
Acceleration model
(Convolutional LSTM network)

FIGURE 3. **Proposed concept of accelerating topology optimization incorporated with deep learning for the case of $s_{window} = 3$, $n_{start} = 1$, and $n_{window} = 1$.**

similar with the substructuring technique in the finite element analysis [28].

In this study, the 88-line MATLAB code [29] was modified to handle randomly generated boundary conditions and to obtain a history of design changes through the iterations. The detailed optimization formulation and the corresponding parameter setting are described in the reference [29]. The radius of sensitivity filtering was randomly selected between 1.0–8.0 (normalized distance metric divided by an element size) to reflect the effect of filtering radius on predicting a near-optimal design in the deep neural network. The volume fraction was also randomly selected between 0.1–0.5. The penalization exponent was set to 3.0.

B. TRAINING DATA GENERATION

The proposed acceleration model predicts a near-optimal design based on the spatiotemporal characteristics of intermediate designs at an early stage of topology optimization. In this study, the training data were generated by capturing a series of density distributions at each iteration and the corresponding optimized result. Figure 2 shows a conceptual diagram of constructing the training data from the optimization history. Under a randomly selected boundary condition, an intermediate density distribution was determined at each iteration. Then, a series of density distributions (red solid and dotted boxes in Fig. 2(a)) were captured as an input data set. A random filtering radius (r_{min}) was also provided as an input data set. The optimized result at the final iteration was used as the output dataset. To construct the training data, three parameters were defined in this study: window size (s_{window}), starting iteration number (n_{start}), and number of windows (n_{window}). In each optimization case, the window size determines the number of consecutive iterations that are considered for training and prediction. For example, the window size (s_{window}) was set to 3 in Fig. 2(a). The starting iteration number (n_{start}) is the lowest iteration number that is captured for use as the training data. If the starting iteration number is set to 5, intermediate designs until the fourth iteration are not used for training and prediction. The number of windows (n_{window}) indicates the number of training datasets generated in each optimization case. In Fig. 2(a), when the n_{window} is set to 2, the first three consecutive iterations (red solid boxes) and the next three consecutive iterations (red dotted boxes) are used to form the training data. As shown in Fig. 2(b), each captured history, that can be expressed as a tensor of $s_{window} \times 64 \times 64$, and the corresponding filtering radius form the input dataset. The corresponding optimized results becomes the output dataset. Common data augmentation techniques were used to create more training data from the existing dataset.

C. PROPOSED DEEP LEARNING-BASED FRAMEWORK OF ACCELERATING TOPOLOGY OPTIMIZATION

Figure 3 depicts a procedure of accelerating the topology optimization incorporated with deep learning. For a given problem, topology optimization is first performed until the predefined iteration (i.e., $n_{start} + n_{window} - 1$). After a whole design domain is divided into a set of unit modules, the input dataset obtained from topology optimization (i.e., iteration history and filtering radius) is used to separately predict a near-optimal design of each unit module through a trained network. Then, a combined near-optimal design of a whole design domain is used as a new initial design to perform the second-stage topology optimization. In this study, the final design of the proposed framework is considered as the optimized design. This result will be compared with the solution of conventional topology optimization in the subsequent sections.

As already explained, the proposed model handles a design domain of 64 × 64 finite elements (or resolution in images) as a unit module. Note that any larger design domain can be decomposed into a set of unit modules, each of which can deliver its own near-optimal design. Such domain decomposition enables the proposed method to be scalable to any arbitrary domain size (or resolution). The numbers of unit modules along the x and y-directions are defined as $n_{\text{module,x}}$ and $n_{\text{module,y}}$, respectively. Then, the entire domain is divided into total $n_{\text{module,x}} \times n_{\text{module,y}}$ unit modules. If a target design domain has n_{elx} and n_{ely} finite elements in the x- and y-directions, respectively, $n_{\text{module,x}}$ and $n_{\text{module,y}}$ can be determined by rounding the values of $2 \times (n_{\text{elx}} / 64) - 1$ and $2 \times (n_{\text{ely}} / 64) - 1$. Figure 4 shows the concept of dividing a target design domain into a set of unit modules and integrating them to predict a near-optimal design. For example, a design domain of 128 × 128 elements can be divided into nine unit modules (i.e., $n_{\text{module,x}} = n_{\text{module, y}} = 3$) with overlapped areas between the modules. Because each module has its own design history (i.e., spatiotemporal characteristics of intermediate designs), the proposed acceleration model can predict a near-optimal design for each module. Each near-optimal design is then integrated into an entire design of 128 × 128 elements. In the overlapped area, a higher density value is selected between two density values at the same position. Thus, the proposed acceleration model is applicable for a design domain of any arbitrary size.

D. DEEP NEURAL NETWORK FOR SPATIOTEMPORAL DATA USING CONVOLUTIONAL LONG SHORT-TERM MEMORY LAYERS

Using the information on iterative design changes, the proposed model infers a direction of shape change, which is conceptually similar to the design velocity in shape optimization [30]. These iterative design changes can also be regarded as video frames that change over time, as shown in Fig. 5. To effectively handle such a specific type of spatiotemporal data, this study implemented the convolutional long short-term memory (convLSTM) [23]. The convLSTM network, which is a representative recurrent neural network, incorporates the features of the CNN [24] and LSTM [25] and has been widely used for weather predictions [31]. In this network, the CNN contributes to capturing the spatial features of the time-series image data such as video frames, and the LSTM contributes to capturing the temporal variation of these time-series image data. For example, the convLSTM can infer the future movement of a walking person by training the video frame data shown in Fig. 5. The detailed architecture of the convLSTM is suggested in [23]. In this study, considering the iteration number as time, the convLSTM was used to predict the density distribution at a future iteration.

Figure 6 shows the overall structure of the proposed acceleration model which consists of the convLSTM layers. As explained in Sections 2.2 and 2.3, this model was designed to receive information on the total s_{window} intermediate designs with a 64 × 64 resolution and a filtering radius (r_{min}) as the input dataset and to return a single near-optimal design with a 64 × 64 resolution as the output dataset. A single value of r_{min} was expanded into the resolution of the convolutional filter through a fully connected layer. The overall structure of the model is based on the U-Net [32], in which the input data converge into the latent space (encoder) and then diverges to its original resolution (decoder). Another prominent feature of the proposed model is the direct connection between the layers with the same image resolution. These directly connected layers contribute to preserving the original geometric features and achieving better prediction performances. The number of convolutional

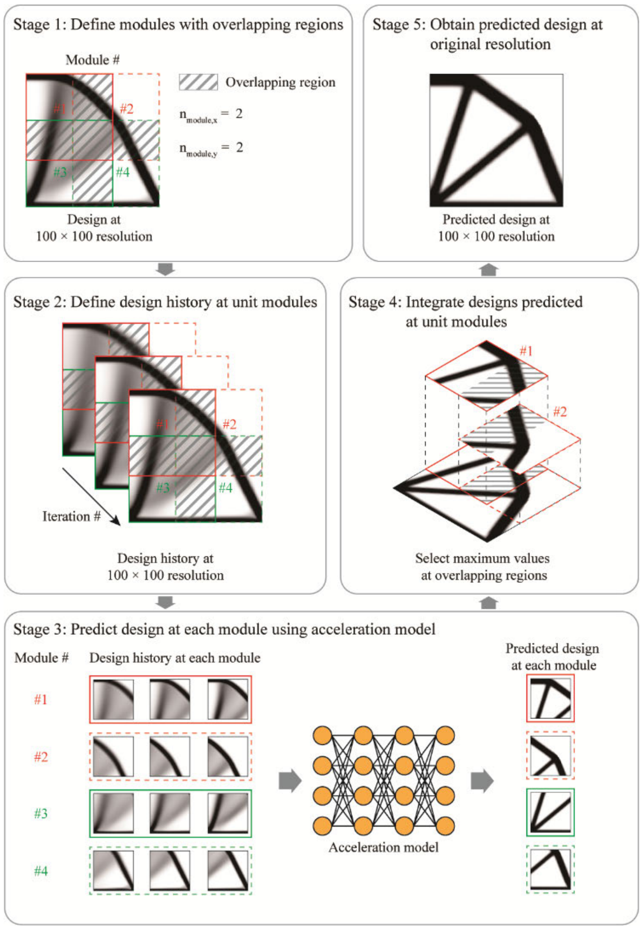

Stage 1: Define modules with overlapping regions
Module #
#1
#2
#3
#4
Overlapping region
$n_{\text{module,x}} = 2$
$n_{\text{module,y}} = 2$
Design at 100 × 100 resolution
Stage 5: Obtain predicted design at original resolution
Predicted design at 100 × 100 resolution
Stage 2: Define design history at unit modules
Iteration #
Design history at 100 × 100 resolution
Stage 4: Integrate designs predicted at unit modules
Select maximum values at overlapping regions
Stage 3: Predict design at each module using acceleration model
Module #
Design history at each module
Predicted design at each module
Acceleration model

FIGURE 4. Concept of predicting a near-optimal design of an arbitrary domain size, based on a unit module ($s_{\text{window}} = 3$, $n_{\text{module,x}} = 2$, $n_{\text{module,y}} = 2$).

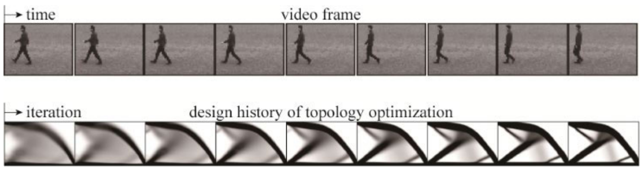


FIGURE 5. Comparison between video frame data [33] and iterative design changes in topology optimization.

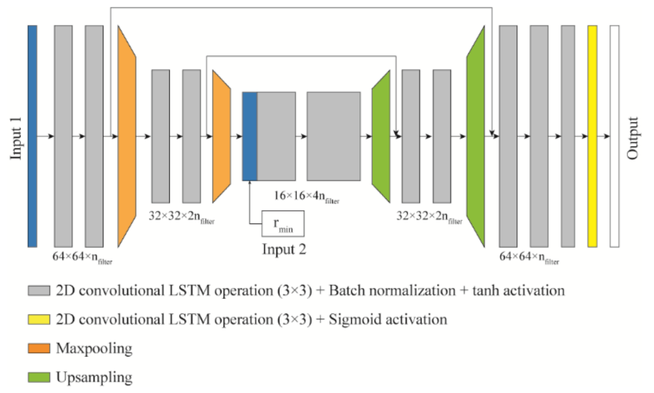

Input 1
64×64×n_filter
32×32×2n_filter
16×16×4n_filter
r_min
Input 2
32×32×2n_filter
64×64×n_filter
Output
2D convolutional LSTM operation (3×3) + Batch normalization + tanh activation
2D convolutional LSTM operation (3×3) + Sigmoid activation
Maxpooling
Upsampling

FIGURE 6. Architecture of the proposed deep neural network for accelerating topology optimization.

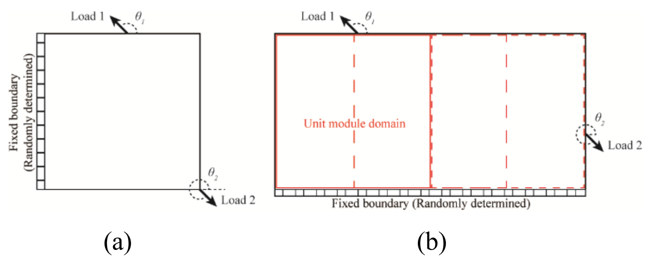

Load 1
θ1
Fixed boundary (Randomly determined)
θ2
Load 2
(a)
Load 1
θ1
Unit module domain
θ2
Load 2
Fixed boundary (Randomly determined)
(b)

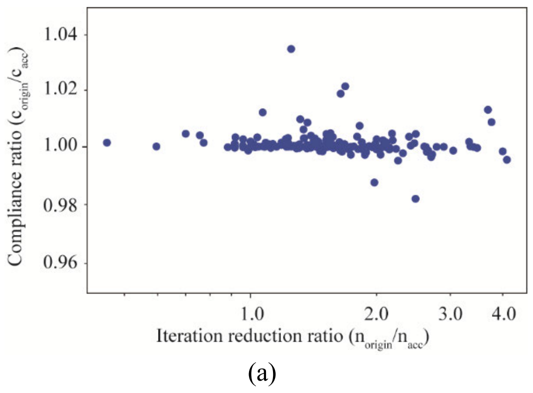
FIGURE 7. Design domains with a randomly selected fixed boundary and two-load conditions for model test: (a) a 64 × 64 domain with a single unit module and (b) a 128 × 64 domain with three unit modules (red solid, long-dashed, and short-dashed boxes).

filters (n_{filter}) was set to 32, and the recurrent activation function for the convLSTM layer was set to "hard sigmoid."

E. MODEL TRAINING AND VALIDATION

A total of 5,000 optimization cases were used for model training: 80% of them were used as the training data and the remaining as the validation data. The proposed acceleration model was trained by varying three hyperparameters (n_{start}, n_{window}, and s_{window}), which are related to training data generation and the model structure. The number of trainable parameters of the deep learning model is determined using s_{window}. For example, when s_{window} is set to 5, approximately 3,700,000 trainable parameters are used. For model training, the Adam optimizer [34] was used with a loss function of the mean absolute error between the target and predicted design. By performing a maximum of 500 epochs, the training process stopped when the validation loss did not decrease for the last 50 epochs.

For model test, topology optimization was performed with the trained network for an additional 200 optimization cases, as shown in Fig. 7. The conventional topology optimization was performed with the 88-line MATLAB code [29], and the proposed method was performed using the trained network with the 88-line MATLAB code. The design domain in Fig. 7(a) has the same resolution as that of a unit module, whereas the design domain in Fig. 7(b) has a 128 × 64 resolution to investigate the scalability of the proposed model. Fixed boundary and two-load conditions were randomly generated, as described in Section 2.1. The filtering radius (r_{min}) was randomly determined between 1.0–6.0.



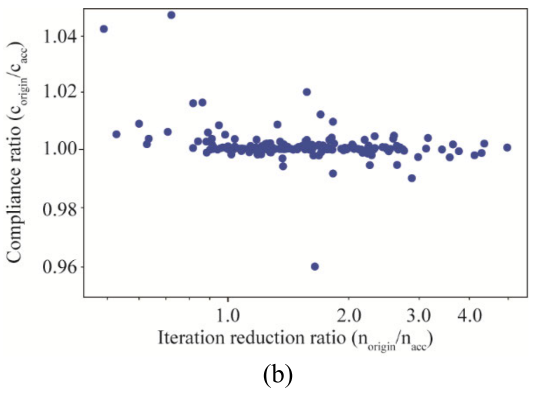


FIGURE 8. Distribution of iteration number ratio and compliance ratio in the test cases of (a) 64 × 64 resolution and (b) 128 × 64 resolution.

Table 1 shows the results of the model training with various combinations of hyperparameters (n_{start}, n_{window}, and s_{window}). The test values for the hyperparameters were determined so that possible combinations of hyperparameters could be examined in the range where the iteration number before the acceleration model do not exceed 30. A total of 42 sets were determined by considering 3 cases for n_{window} for each of the 14 cases obtained by the combination of n_{start} and s_{window}. To evaluate the performance of the proposed model, the failure rate and iteration reduction ratio were defined in this study. The failure rate was defined as the ratio of cases in which a difference in compliance between the proposed and conventional topology optimization is larger than 5%. The iteration reduction ratio was defined as the ratio of the total iteration number of conventional topology optimization (n_{origin}) to that of the proposed method (n_{acc}). The higher the iteration reduction ratio, the better the acceleration performance. On average, the proposed method shows a 1.66 iteration reduction ratio with a failure rate of 1.34%. To provide a reliable result with a reasonably boosted efficiency, this study selected the hyperparameters of Set 37 in Table 1 (i.e., $n_{\mathrm{start}} = 1$, $n_{\mathrm{window}} = 1$, and $s_{\mathrm{window}} = 20$). These hyperparameters will be used in the forthcoming numerical examples.

It is interesting to note that, with relatively small n_{start} and s_{window} values, the proposed acceleration model can use only a limited number of designs at an early stage to predict

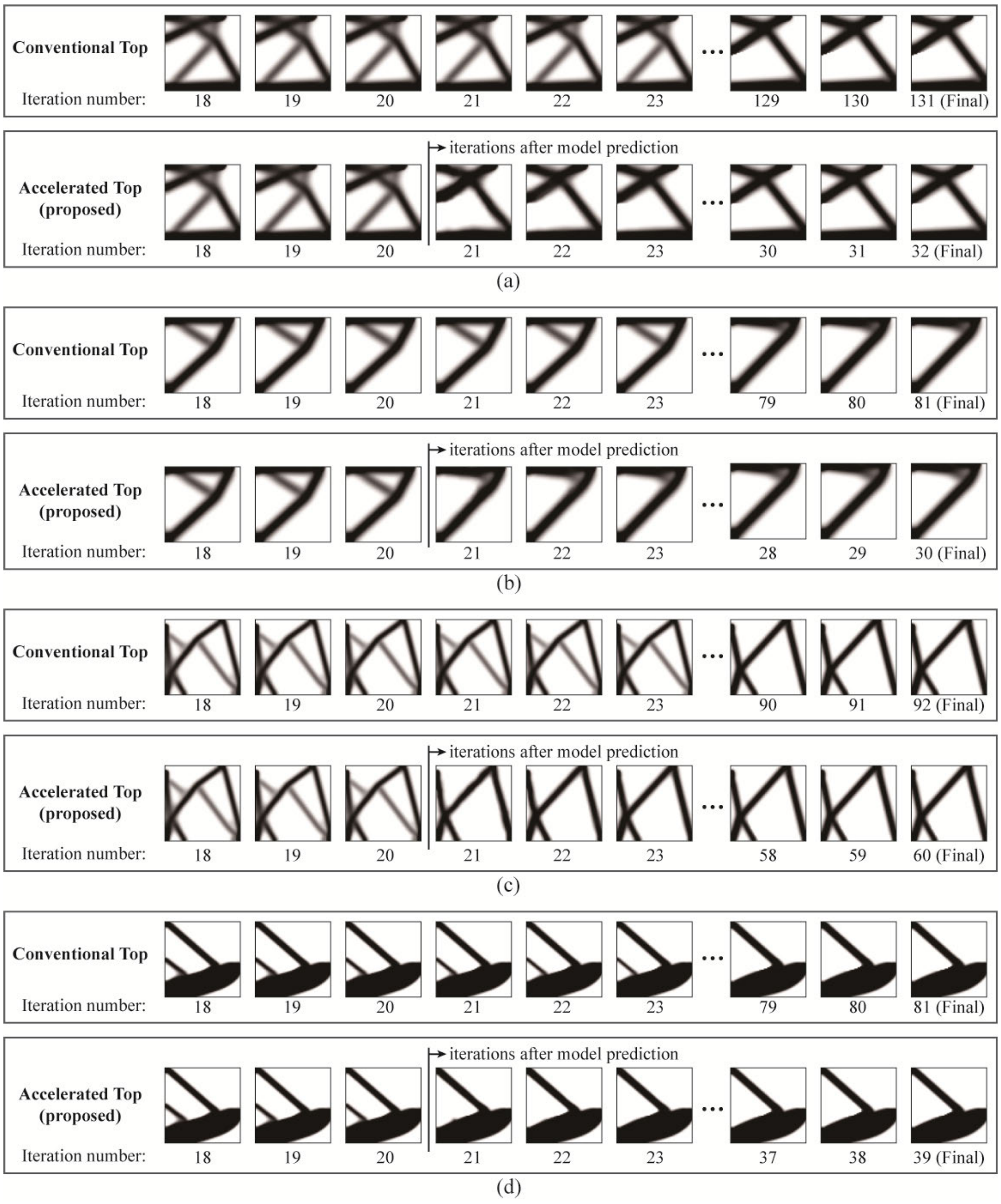

Conventional Top
Iteration number: 18 19 20 21 22 23 ... 129 130 131 (Final)
iterations after model prediction
Accelerated Top (proposed)
Iteration number: 18 19 20 21 22 23 ... 30 31 32 (Final)
(a)
Conventional Top
Iteration number: 18 19 20 21 22 23 ... 79 80 81 (Final)
Accelerated Top (proposed)
Iteration number: 18 19 20 21 22 23 ... 28 29 30 (Final)
(b)
Conventional Top
Iteration number: 18 19 20 21 22 23 ... 90 91 92 (Final)
Accelerated Top (proposed)
Iteration number: 18 19 20 21 22 23 ... 58 59 60 (Final)
(c)
Conventional Top
Iteration number: 18 19 20 21 22 23 ... 79 80 81 (Final)
Accelerated Top (proposed)
Iteration number: 18 19 20 21 22 23 ... 37 38 39 (Final)
(d)

FIGURE 9. Representative design histories of conventional and proposed topology optimization of a 64 × 64 design domain.

a near-optimal design, thereby resulting in a higher failure rate. However, $(n_{\text{start}} + s_{\text{window}}) > 15$ leads to a failure rate less than 1%. Note that $(n_{\text{start}} + s_{\text{window}} - 1)$ is the iteration number at which the proposed acceleration model is applied. Thus, the delayed application of the proposed model is advantageous in enhancing the accuracy, while disadvantageous in reducing the total number of iterations. This trend shows a trade-off between the model accuracy and computational efficiency.

Figure 8 shows the detailed distribution of the iteration reduction ratio and the compliance ratio for additional 200 optimization cases with the selected hyperparameters:

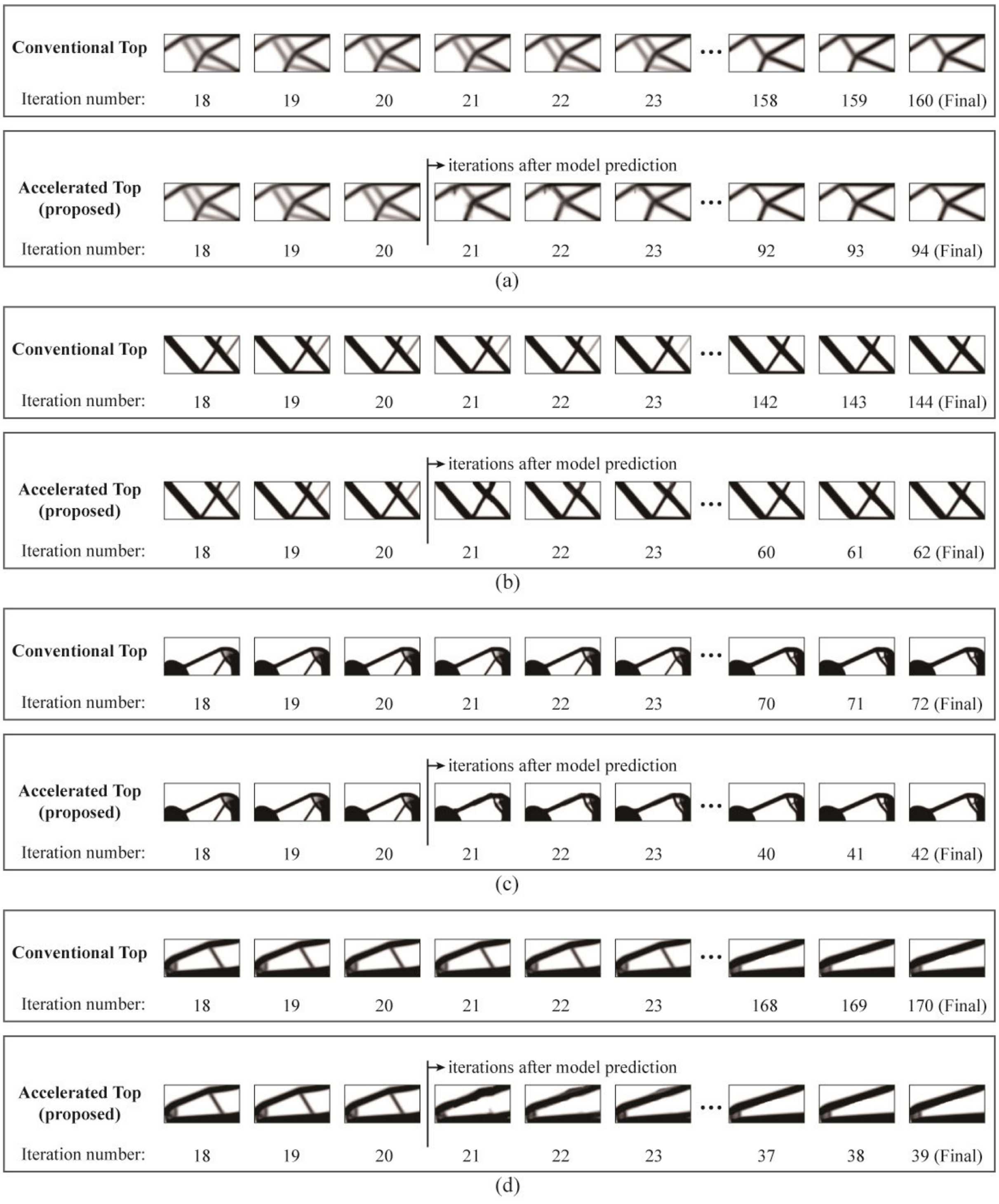

Conventional Top
Iteration number: 18 19 20 21 22 23 ... 158 159 160 (Final)
Accelerated Top (proposed)
iterations after model prediction
Iteration number: 18 19 20 21 22 23 ... 92 93 94 (Final)
(a)
Conventional Top
Iteration number: 18 19 20 21 22 23 ... 142 143 144 (Final)
Accelerated Top (proposed)
iterations after model prediction
Iteration number: 18 19 20 21 22 23 ... 60 61 62 (Final)
(b)
Conventional Top
Iteration number: 18 19 20 21 22 23 ... 70 71 72 (Final)
Accelerated Top (proposed)
iterations after model prediction
Iteration number: 18 19 20 21 22 23 ... 40 41 42 (Final)
(c)
Conventional Top
Iteration number: 18 19 20 21 22 23 ... 168 169 170 (Final)
Accelerated Top (proposed)
iterations after model prediction
Iteration number: 18 19 20 21 22 23 ... 37 38 39 (Final)
(d)

FIGURE 10. Representative design histories of conventional and proposed topology optimization of a 128 × 64 domain.

$n_{\text{start}} = 1$, $n_{\text{window}} = 1$, and $s_{\text{window}} = 20$. Here, compliance ratio was defined as the ratio of compliance of the conventional topology optimization to that of the accelerated topology optimization. In both design domains for model test, the compliance ratios were well distributed (i.e., close to 1.0).

Figures 9 and 10 show the representative design histories of the conventional and proposed topology optimization among the additional 200 optimization cases: four cases with a 64 × 64 resolution (Fig. 9) and another four cases with 128 × 64 resolution (Fig. 10). In the proposed topology optimization, the first three designs show the last three input data from

TABLE 1. Results of model test according to various hyper parameters.

Set	n_{start}	n_{window}	s_{window}	64 × 64 domain		128 × 64 domain	
				Failure rate (%)	Iteration reduction ratio	Failure rate (%)	Iteration reduction ratio
1	1	1	5	6.00	2.183	20.00	2.072
2	1	2	5	3.50	2.229	10.50	1.837
3	1	3	5	4.00	2.263	10.50	1.921
4	5	1	5	0.50	1.886	3.50	1.674
5	5	2	5	2.01	1.921	4.00	1.711
6	5	3	5	2.01	1.931	3.50	1.621
7	10	1	5	0.53	1.779	1.06	1.608
8	10	2	5	0.53	1.755	0.53	1.577
9	10	3	5	0.53	1.804	1.06	1.615
10	15	1	5	0.56	1.670	0.56	1.499
11	15	2	5	0.56	1.612	0.56	1.514
12	15	3	5	0.56	1.666	1.11	1.501
13	20	1	5	0.61	1.582	0.58	1.512
14	20	2	5	0.00	1.584	0.00	1.503
15	20	3	5	0.00	1.607	1.16	1.466
16	1	1	10	1.55	1.837	2.50	1.644
17	1	2	10	1.03	1.902	2.00	1.689
18	1	3	10	1.03	1.906	2.50	1.638
19	5	1	10	0.53	1.668	0.53	1.526
20	5	2	10	0.53	1.722	0.00	1.567
21	5	3	10	0.53	1.790	1.60	1.573
22	10	1	10	0.00	1.752	0.56	1.513
23	10	2	10	0.00	1.668	0.56	1.528
24	10	3	10	0.00	1.738	0.56	1.553
25	15	1	10	0.61	1.630	0.00	1.484
26	15	2	10	0.00	1.624	0.58	1.494
27	15	3	10	0.00	1.586	0.00	1.495
28	1	1	15	0.53	1.736	1.06	1.564
29	1	2	15	0.53	1.715	0.00	1.562
30	1	3	15	1.06	1.778	0.53	1.592
31	5	1	15	1.12	1.682	1.11	1.509
32	5	2	15	0.56	1.670	0.56	1.544
33	5	3	15	0.56	1.627	0.56	1.480
34	10	1	15	0.00	1.565	0.58	1.505
35	10	2	15	0.61	1.597	1.16	1.559
36	10	3	15	0.00	1.665	0.58	1.582
37	1	1	20	0.00	1.674	0.00	1.657
38	1	2	20	0.57	1.622	0.00	1.530
39	1	3	20	0.57	1.603	0.00	1.519
40	5	1	20	0.00	1.648	1.16	1.530
41	5	2	20	0.61	1.665	0.00	1.587
42	5	3	20	0.61	1.655	0.00	1.566

the s_{window}, which were used to predict a near-optimal design, and the next three designs show the optimization history after re-performing the topology optimization from the predicted near-optimal design. The last three designs show the final design changes at convergence.

Particularly, Figures 9(a)–(c) show optimization cases where blurry geometries (or gray zone) disappeared just after the application of the proposed acceleration model. In Fig. 9(d), even a clear local geometry disappeared by predicting a near-optimal design. It should be noted that,

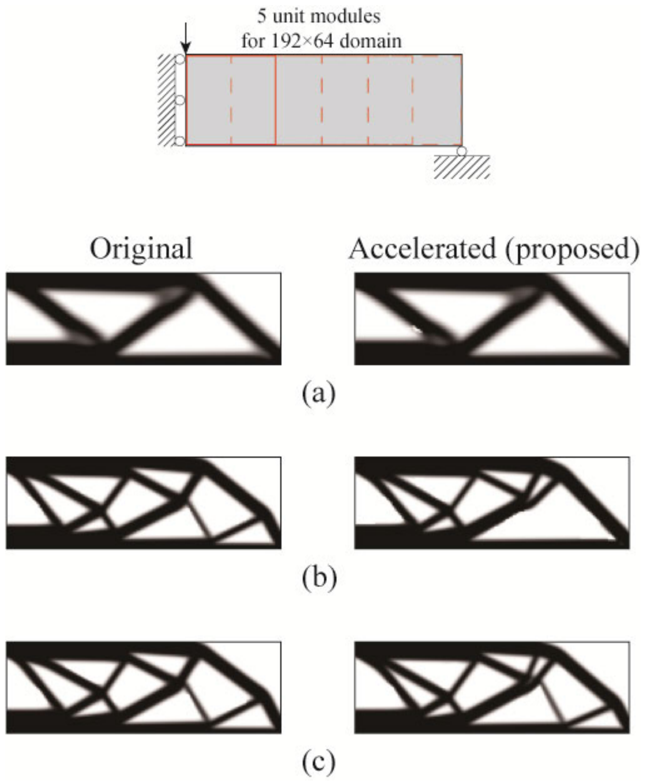


FIGURE 11. Design domain with five unit modules (red solid and dashed boxes) and optimization results for the MBB beam.

only after a limited number of iterations (20 iterations in this study), the proposed model successfully predicted a topologically optimal design, whereas conventional topology optimization failed to achieve it. These results show that the proposed acceleration model can capture a spatiotemporal design change that may be less visible (or even invisible) at the early stage of topology optimization. Such a drastic change to a near-optimal design can boost the overall convergence in topology optimization.

The overall features of the optimization results in a larger design domain (Fig. 10) are similar to those shown in Fig. 9. In the cases of Figs. 10(a), (b), and (d), a abrupt disappearance of blurry geometries boosted the overall convergence of the topology optimization. In Fig. 10(c), a clear local geometry disappeared after the application of the proposed acceleration model. Table 2 lists the iteration reduction ratios and compliance errors for model test.

III. NUMERICAL EXAMPLES

In this section, the proposed acceleration model will be applied to well-known examples such as the MBB beam and Michell truss. Although the proposed model was trained only under two-load and single side-fixed boundary condition randomly generated in a unit module, each numerical example has a design domain of a different size (no 64 × 64 resolution) under a different boundary condition (no two-load and single side-fixed condition) to check the applicability of the proposed method to general problems. Each example investigates a different combination of resolutions and filtering radii: Case (a) with the original resolution and filtering radius, Case (b) with the same resolution and a half $r_{\min}$, and Case (c) with a doubled resolution and the same radius. Note that Cases (b) and (c) should have the same optimized results to guarantee mesh independence. For the proposed method, the hyperparameters of Set 37 in Table 1 were used for the trained neural network. The detailed network structure is the same as that suggested in Fig. 6. For each case, CPU time required for the iterative computation was measured using a PC with Intel Core i5 CPU.

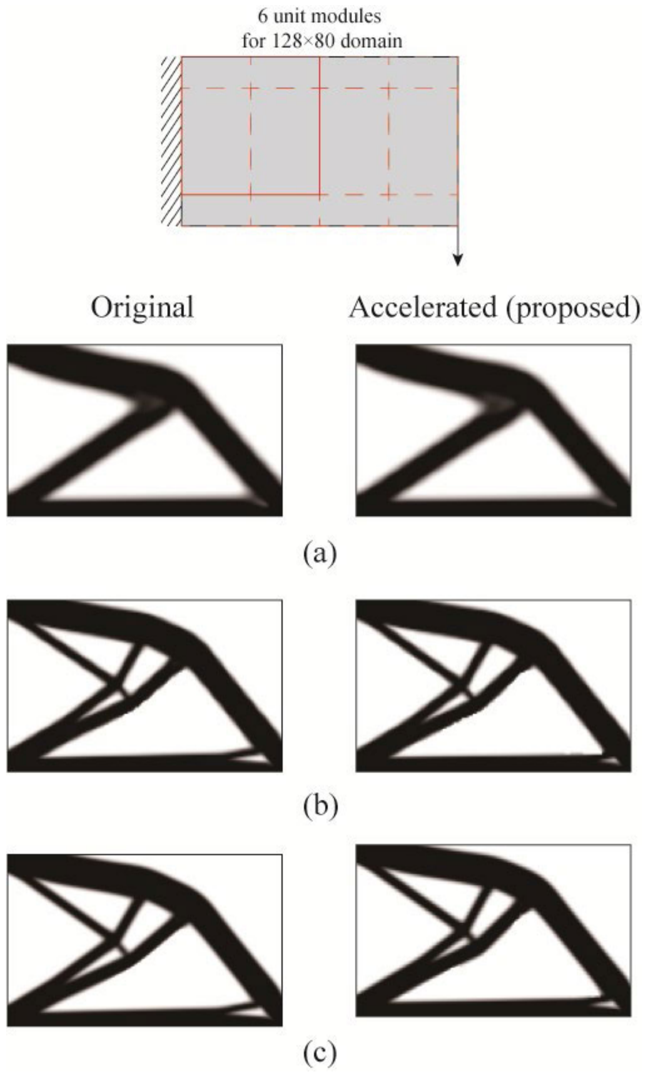


FIGURE 12. Design domain with six unit modules (red solid and dashed boxes) and optimization results for the cantilever example.

A. MBB BEAM

Figure 11 shows the design domain and the corresponding optimization results for the MBB beam example. This study used the same conditions as those described in the 88-lines topology optimization code [29]. Table 3 presents the number of iterations, CPU time, and compliance values of the conventional and proposed topology optimization. In Case (a),

TABLE 2. Comparison of iterations and compliance values between the conventional and proposed topology optimization techniques shown in Fig. 9 and 10.

Figure		Number of iterations			Compliance		
		Conventional	Proposed	Ratio (n_{origin}/n_{acc})	Conventional	Proposed	Error (%)
9	(a)	131	32	4.09	31.58	31.72	0.45
	(b)	81	30	2.70	30.93	31.04	0.36
	(c)	92	60	1.53	60.67	60.54	0.21
	(d)	81	39	2.08	9.69	9.69	0.02
10	(a)	160	94	1.70	124.34	122.84	1.20
	(b)	144	62	2.32	21.81	21.71	0.45
	(c)	72	42	1.71	9.62	9.62	0.05
	(d)	170	39	4.36	26.41	26.36	0.20

TABLE 3. Comparison of iterations and compliance values between the conventional and proposed topology optimization for the MBB beam.

Case	Domain	r_{min}	Number of iterations			CPU time (sec)			Compliance		
			Conventional (n_{origin})	Proposed (n_{acc})	Ratio (n_{origin}/n_{acc})	Conventional	Proposed	Ratio	Conventional	Proposed	Error (%)
(a)	192×64	5.6	100	79	1.27	12.53	9.94	1.26	211.32	211.18	0.07
(b)	192×64	2.8	283	102	2.77	36.01	12.72	2.83	195.98	196.14	0.08
(c)	384×128	5.6	536	88	6.09	281.08	46.25	6.08	197.96	200.19	1.12

TABLE 4. Comparison of iterations and compliance values between the conventional and proposed topology optimization for the cantilever example.

Case	Domain	r_{min}	Number of iterations			CPU time (sec)			Compliance		
			Conventional (n_{origin})	Proposed (n_{acc})	Ratio (n_{origin}/n_{acc})	Conventional	Proposed	Ratio	Conventional	Proposed	Error (%)
(a)	128×80	3.84	67	56	1.20	7.12	5.96	1.19	59.43	59.43	0.01
(b)	128×80	2.1	178	81	2.20	18.62	8.59	2.17	56.57	56.47	0.18
(c)	256×160	4.2	332	80	4.15	152.24	35.11	4.34	58.16	58.22	0.10

the proposed method determined the optimized design, which is identical to that of conventional topology optimization in terms of topological layout and compliance (only 0.07% compliance error). However, in Cases (b) and (c), the optimized design was topologically different, albeit with a negligible compliance error (maximum 1.12%). Furthermore, the proposed method failed to achieve mesh independence in some local areas. This is because the proposed acceleration model causes a "bifurcation" in structural designs, which is frequently observed during topology optimization. However, the proposed method provided accelerated convergence up to 6.09 in Case (c). This means that only 16.4% of the original computing time is required to determine the same design. It is interesting to mention that a more significant convergence acceleration was observed at a higher resolution.

B. CANTILEVER

Figure 12 shows the design domain and the optimization results for the cantilever beam. Note that these design domains of 128 × 80 and 256 × 160 cannot be constructed as multiples of 64 (i.e., single side of a unit module). See Section 2.3 to check how to handle the overlapped areas between the unit modules. Table 4 presents the number of iterations, CPU time, and compliance values of the conventional and proposed topology optimization. In all cases, the proposed method successfully determined the optimized designs which

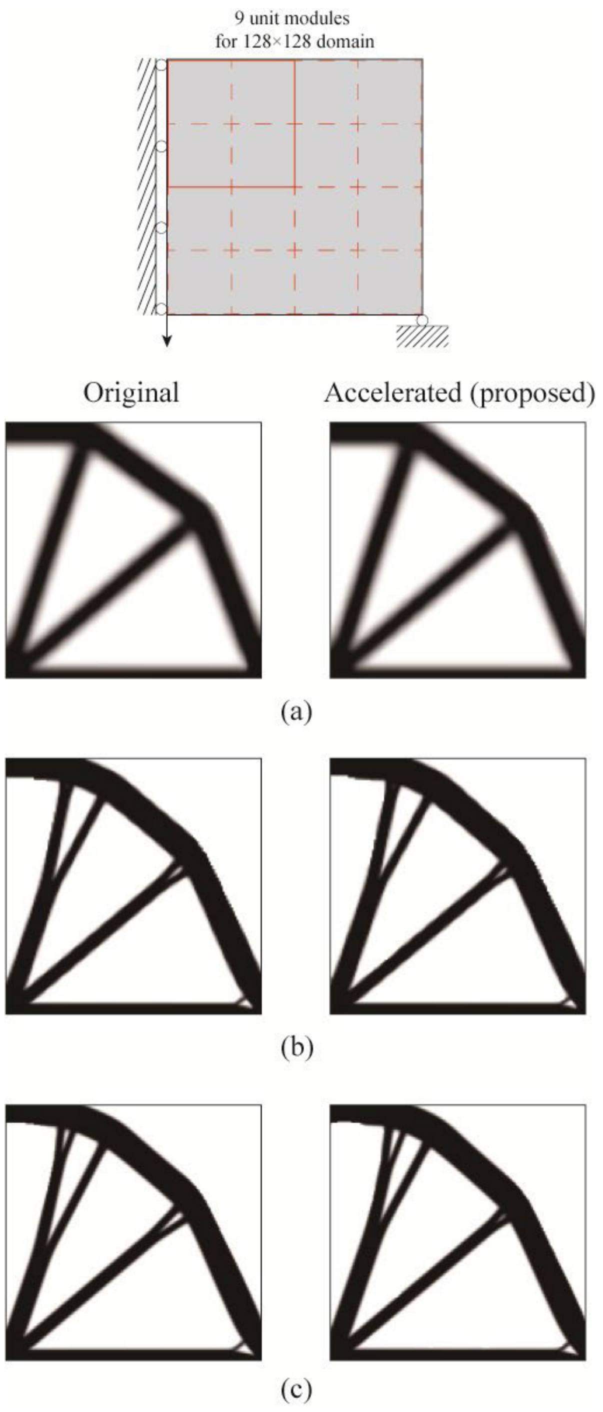


FIGURE 13. **Design domain with nine unit modules (red solid and dashed boxes) and optimization results for Michell truss.**

are identical to those of conventional topology optimization in terms of topological layout and compliance (maximum 0.18% compliance error). Moreover, the proposed method shows mesh independence in Cases (b) and (c), as would be expected. The proposed method also provided an accelerated convergence of up to 4.15. This means that only 24% of the original computing time was required to determine the same optimized design.

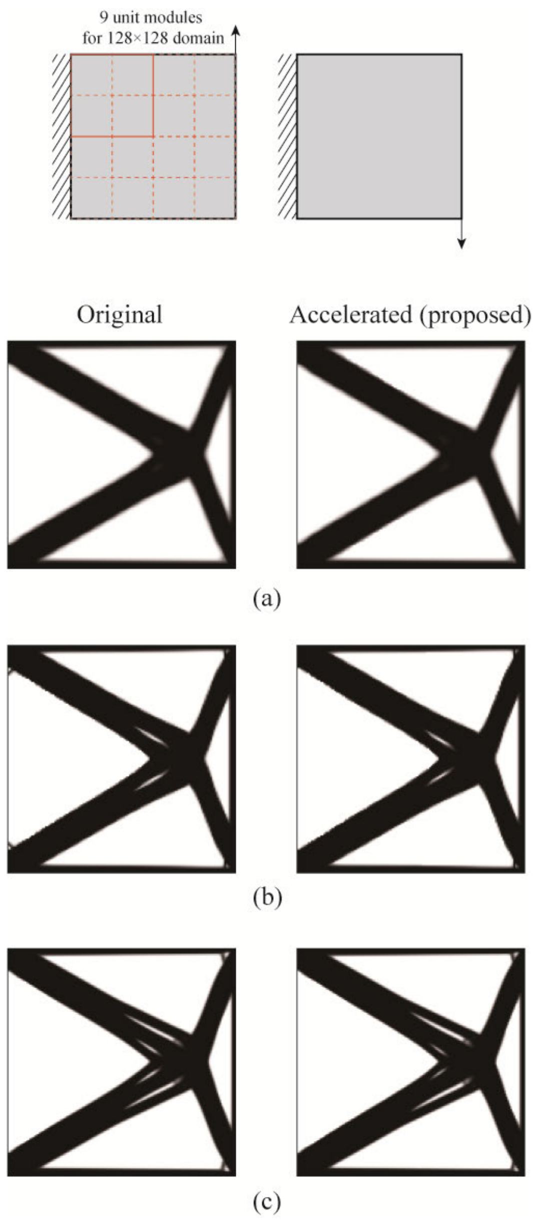


FIGURE 14. **Design domain with nine unit modules (red solid and dashed boxes) and optimization results for a multiple load case.**

C. MICHELL TRUSS

Figure 13 shows the design domain and optimization results for the Michell truss. Table 5 presents the number of iterations, CPU time, and compliance values of the conventional and proposed topology optimization. In similarity with the cantilever example, the proposed method determined the same optimized designs which are topologically and structurally identical to those of conventional topology optimization. The proposed method also shows mesh independence in Cases (b) and (c). In this example, the proposed method provided an accelerated convergence up to 4.39, which means that only 23% of the original computing time was required to obtain the optimized design.

TABLE 5. Comparison of iterations and compliance values between the conventional and proposed topology optimization techniques for Michell truss.

Case	Domain	$r_{\min}$	Number of iterations			CPU time (sec)			Compliance		
			Conventional (n_{origin})	Proposed (n_{acc})	Ratio ($n_{\text{origin}}/n_{\text{acc}}$)	Conventional	Proposed	Ratio	Conventional	Proposed	Error (%)
(a)	128×128	5.5	140	82	1.71	23.91	14.14	1.69	52.18	51.97	0.41
(b)	128×128	1.5	244	120	2.03	41.08	20.39	2.01	45.56	45.70	0.30
(c)	256×256	3	404	92	4.39	308.57	68.34	4.52	47.81	47.97	0.33

TABLE 6. Comparison of iterations and compliance values between the conventional and proposed topology optimization techniques for a multiple load case.

Case	Domain	$r_{\min}$	Number of iterations			CPU time (sec)			Compliance		
			Conventional (n_{origin})	Proposed (n_{acc})	Ratio ($n_{\text{origin}}/n_{\text{acc}}$)	Conventional	Proposed	Ratio	Conventional	Proposed	Error (%)
(a)	128×128	2.5	50	43	1.16	8.87	7.66	1.16	63.95	63.84	0.17
(b)	128×128	1.2	194	124	1.56	36.69	23.67	1.55	61.96	61.84	0.20
(c)	256×256	2.4	231	93	2.48	185.15	71.85	2.58	65.02	64.93	0.13

D. MULTIPLE LOAD CASE

Figure 14 shows the design domain and optimization results for the cantilever with multiple loads. This example is a replication based on a multiple load case described in [29]. Table 6 presents the number of iterations, CPU time, and compliance values of the conventional and proposed topology optimization. In all cases, the proposed method determined the same optimized designs, compared with those of conventional topology optimization (maximum 0.20% compliance error). The proposed method also shows mesh independence in Cases (b) and (c). In this example, the proposed method provided a relatively lower convergence acceleration: 1.16 in Case (a), 1.56 in Case (b), and 2.48 in Case (c).

In summary, although the proposed acceleration model was trained with a 64 × 64 resolution unit module under two-load and single side-fixed boundary condition, it could accelerate the convergence of topology optimization at any arbitrary resolution under different boundary conditions. The acceleration performance tended to increase at a higher resolution. The proposed method shows mesh independence in most cases, but cannot guarantee it in every case.

IV. CONCLUSION

In this study, a deep learning-based framework was proposed to accelerate the 2-D structural topology optimization for general problems. A convolutional LSTM network was applied to the proposed acceleration model to effectively investigate spatiotemporal characteristics in the design history. The deep neural network was trained to predict a near-optimal design from the intermediate density distributions at the early stage of topology optimization. To demonstrate the performance and potential of the proposed model, four numerical examples were investigated under various boundary conditions in the design domains of various sizes. Although the proposed model was trained only under two-load and single side-fixed boundary condition in a unit module, the proposed method successfully accelerated the convergence of topology optimization while providing the same (or nearly identical) optimized design in terms of topological shape and compliance. However, the current framework could not guarantee the mesh independence in all optimization cases owing to limited training with two-load and single side-fixed conditions. To overcome this issue, it would be necessary to train a deep learning model by considering a higher number of concentrated loads (ideally, local loads at every node on the boundary) under a force equilibrium condition. In further work, the proposed method would be extended to cover more practical problems such as nonlinear and/or 3D topology optimization cases.

ACKNOWLEDGMENT

Replication of results: Compiled codes and relevant materials are provided through a repository. *(Younghwan Joo and Yonggyun Yu are co-first authors.)*

REFERENCES

[1] M. P. Bendsoe and O. Sigmund, *Topology Optimization: Theory, Methods, and Applications*. Springer, 2013.

[2] S. Mukherjee, D. Lu, B. Raghavan, P. Breitkopf, S. Dutta, M. Xiao, and W. Zhang, "Accelerating large-scale topology optimization: State-of-the-art and challenges," *Arch. Comput. Methods Eng.*, vol. 1, pp. 1–23, Jan. 2021.

[3] I. G. Jang and B. M. Kwak, "Evolutionary topology optimization using design space adjustment based on fixed grid," *Int. J. Numer. Methods Eng.*, vol. 66, no. 11, pp. 1817–1840, 2006, doi: 10.1002/nme.1607.

[4] S. Y. Kim, I. Y. Kim, and C. K. Mechefske, "A new efficient convergence criterion for reducing computational expense in topology optimization: Reducible design variable method," *Int. J. Numer. Methods Eng.*, vol. 90, no. 6, pp. 752–783, May 2012.

[5] Z. Liao, Y. Zhang, Y. Wang, and W. Li, "A triple acceleration method for topology optimization," *Struct. Multidisciplinary Optim.*, vol. 60, no. 2, pp. 727–744, Aug. 2019, doi: 10.1007/s00158-019-02234-6.

[6] W. Zheng, Y. Wang, Y. Zheng, and D. Da, "Efficient topology optimization based on DOF reduction and convergence acceleration methods," *Adv. Eng. Softw.*, vol. 149, Nov. 2020, Art. no. 102890, doi: 10.1016/j.advengsoft.2020.102890.

[7] T. H. Nguyen, G. H. Paulino, J. Song, and C. H. Le, "A computational paradigm for multiresolution topology optimization (MTOP)," *Struct. Multidisciplinary Optim.*, vol. 41, no. 4, pp. 525–539, Apr. 2010, doi: 10.1007/s00158-009-0443-8.

[8] T. H. Nguyen, G. H. Paulino, J. Song, and C. H. Le, "Improving multiresolution topology optimization via multiple discretizations," *Int. J. Numer. Methods Eng.*, vol. 92, no. 6, pp. 507–530, Nov. 2012.

[9] J. Yoo, I. G. Jang, and I. Lee, "Multi-resolution topology optimization using adaptive isosurface variable grouping (MTOP-aIVG) for enhanced computational efficiency," *Struct. Multidisciplinary Optim.*, vol. 63, no. 4, pp. 1743–1766, Apr. 2021.

[10] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," *Science*, vol. 355, pp. 602–606, Feb. 2017, doi: 10.1126/science.aag2302.

[11] K. Mills, M. Spanner, and I. Tamblyn, "Deep learning and the Schrödinger equation," *Phys. Rev. A, Gen. Phys.*, vol. 96, no. 4, Oct. 2017, Art. no. 042113.

[12] A. P. Singh, S. Medida, and K. Duraisamy, "Machine-Learning-Augmented predictive modeling of turbulent separated flows over airfoils," *AIAA J.*, vol. 55, no. 7, pp. 2215–2227, Jul. 2017, doi: 10.2514/1.J055595.

[13] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, "Accelerating Eulerian fluid simulation with convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3424–3433.

[14] A. Chandrasekhar and K. Suresh, "TOuNN: Topology optimization using neural networks," *Struct. Multidisciplinary Optim.*, vol. 63, no. 3, pp. 1135–1149, Mar. 2021, doi: 10.1007/s00158-020-02748-4.

[15] 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., Imag. Visualizat.*, vol. 4, no. 2, pp. 61–72, Mar. 2016.

[16] N. Aulig and M. Olhofer, "Topology optimization by predicting sensitivities based on local state features," in *Proc. 5th Eur. Conf. Comput. Mech. (ECCM V)*, Barcelona, Spain, 2014, pp. 1–13.

[17] K. Liu, A. Tovar, E. Nutwell, and D. Detwiler, "Towards nonlinear multimaterial topology optimization using unsupervised machine learning and metamodel-based optimization," in *Proc. 41st Design Automat. Conf.*, Aug. 2015.

[18] I. Sosnovik and I. Oseledets, "Neural networks for topology optimization," *Russian J. Numer. Anal. Math. Model.*, vol. 34, no. 4, pp. 215–223, Aug. 2019, doi: 10.1515/rnam-2019-0018.

[19] Y. Yu, T. Hur, J. Jung, and I. G. Jang, "Deep learning for determining a near-optimal topological design without any iteration," *Struct. Multidisciplinary Optim.*, vol. 59, no. 3, pp. 787–799, Mar. 2019.

[20] H. Sasaki and H. Igarashi, "Topology optimization accelerated by deep learning," *IEEE Trans. Magn.*, vol. 55, no. 6, pp. 1–5, Jun. 2019.

[21] C. Qian and W. Ye, "Accelerating gradient-based topology optimization design with dual-model artificial neural networks," *Struct. Multidisciplinary Optim.*, vol. 63, no. 4, pp. 1687–1707, Apr. 2021, doi: 10.1007/s00158-020-02770-6.

[22] N. A. Kallioras, G. Kazakis, and N. D. Lagaros, "Accelerated topology optimization by means of deep learning," *Struct. Multidisciplinary Optim.*, vol. 62, no. 3, pp. 1185–1212, Sep. 2020, doi: 10.1007/s00158-020-02545-z.

[23] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 802–810.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[26] A. Pfeuffer and K. Dietmayer, "Separable convolutional LSTMs for faster video segmentation," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1072–1078.

[27] A. Arbelle and T. R. Raviv, "Microscopy cell segmentation via convolutional LSTM networks," in *Proc. IEEE 16th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2019, pp. 1008–1012.

[28] R. D. Cook, *Concepts and Applications of Finite Element Analysis*. Hoboken, NJ, USA: Wiley, 2007.

[29] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, and O. Sigmund, "Efficient topology optimization in MATLAB using 88 lines of code," *Struct. Multidiscipl. Optim.*, vol. 43, no. 1, pp. 1–16, Jan. 2010, doi: 10.1007/s00158-010-0594-7.

[30] K. K. Choi and N.-H. Kim, *Structural Sensitivity Analysis and optimization 1: Linear Systems*. Springer, 2004.

[31] X. Qing and Y. Niu, "Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM," *Energy*, vol. 148, pp. 461–468, Apr. 2018, doi: 10.1016/j.energy.2018.01.177.

[32] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput.-Assist. Intervent.*, 2015, pp. 234–241, doi: 10.1007/978-3-319-24574-4_28.

[33] C. Schuldt, "Recognizing human actions: A local SVM approach," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2004, pp. 32–36.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

FURTHER READING

I. G. Jang and B. M. Kwak, "Design space optimization using design space adjustment and refinement," *Struct. Multidisciplinary Optim.*, vol. 35, no. 1, pp. 41–54, 2008.

YOUNGHWAN JOO received the B.S. and Ph.D. degrees in mechanical engineering from KAIST, in 2012 and 2017, respectively. He then worked as a Senior Researcher at the Korea Atomic Energy Research Institute. He is currently a Senior Researcher with the Energy Management System Laboratory, Korea Institute of Energy Research, Daejeon, South Korea. His expertise covers design optimization of heat transfer devices for industrial applications.

YONGGYUN YU received the B.S. and Ph.D. degrees from the Department of Mechanical Engineering, KAIST, in 2001 and 2010, respectively. He then worked as a Postdoctoral Fellow and a Research Assistant Professor at the Mobile Harbor Center, KAIST. He is currently a Principal Researcher and a Research Director with the Artificial Intelligence Application and Strategy Team, Korea Atomic Energy Research Institute, Daejeon, South Korea. His expertise covers design optimization of musical instruments and nuclear reactor components. His current research interest includes industrial application of deep learning.

IN GWUN JANG (Member, IEEE) received the B.S. and Ph.D. degrees from the Department of Mechanical Engineering, KAIST, Daejeon, South Korea, in 1999 and 2006, respectively. He then worked as a Postdoctoral Fellow at Queen's University, Kingston, ON, Canada. He is currently an Associate Professor with the Cho Chun Shik Graduate School of Green Transportation, KAIST. His expertise covers design optimization ranging from the component-level to system-level, based on multidisciplinary computational analysis. His current research interest includes develop a new topology optimization algorithm which is incorporated with deep learning.