applied sciences

Article

MLGen: Generative Design Framework Based on Machine Learning and Topology Optimization

Nikos Ath. Kallioras † and Nikos D. Lagaros *,†

Institute of Structural Analysis and Antiseismic Research, School of Civil Engineering, National Technical University of Athens, 9, Heroon Polytechniou Str., Zografou Campus, GR-15780 Athens, Greece; nkallio@mail.ntua.gr

* Correspondence: nlagaros@central.ntua.gr; Tel.: +30-210-772-2625

† These authors contributed equally to this work.

Abstract: Design and manufacturing processes are entering into a new era as novel methods and techniques are constantly introduced. Currently, 3D printing is already established in the production processes of several industries while more are continuously being added. At the same time, topology optimization has become part of the design procedure of various industries, such as automotive and aeronautical. Parametric design has been gaining ground in the architectural design literature in the past years. Generative design is introduced as the contemporary design process that relies on the utilization of algorithms for creating several forms that respect structural and architectural constraints imposed, among others, by the design codes and/or as defined by the designer. In this study, a novel generative design framework labeled as MLGen is presented. MLGen integrates machine learning into the generative design practice. MLGen is able to generate multiple optimized solutions which vary in shape but are equivalent in terms of performance criteria. The output of the proposed framework is exported in a format that can be handled by 3D printers. The ability of MLGen to efficiently handle different problems is validated via testing on several benchmark topology optimization problems frequently employed in the literature.

Keywords: generative design; machine learning; topology optimization; long short-term networks; ant colony optimization

check for updates

Citation: Kallioras, N.A.; Lagaros, N.D. MLGen: Generative Design Framework Based on Machine Learning and Topology Optimization. *Appl. Sci.* **2021**, *11*, 12044. https://doi.org/10.3390/app112412044

Academic Editor: Chiara Bedon

Received: 16 November 2021
Accepted: 14 December 2021
Published: 17 December 2021

1. Introduction

Generally speaking, structural optimization can be distinguished into three categories: topology, shape and sizing optimization [1,2]. Topology optimization refers to a mathematical procedure that aims to identify the optimal shape, in terms of structural performance, of a structural system when subjected to specific load and support conditions. This is achieved by optimizing the topological placement of a specific quantity of material into the design domain. Apart from structural performance, topology optimization can be used for optimization with respect to sizing and shape criteria as well. The application of such approaches helps, among others, to create structural systems that are very close to their optimal shape, and thus can be considered a supporting procedure in the conceptual design phase [3,4].

With the term generative design, a design exploration process is defined. The basic idea of this process is that engineers/designers in general introduce their design goals and constraints imposed by design codes, etc., into the generative design framework, along with other parameters, such as performance demands, material properties, etc. The generative design framework should be able to examine all or most of the possible solutions of the specific problem, and produce design alternatives of equivalent performance and criteria values.

During the last decade, research on modern soft computing methods drew significant attention [5], mainly due to the excessive amount of data generated. This led to the development of new methods, able to handle large amounts of data and extremely complex problem definitions. The scope of this study is to examine and combine modern machine learning methods with topology optimization procedures aiming to generate multiple forms in the concept of generative design. Recently, there is growing interest for applying deep learning techniques in topology optimization, mainly for accelerating its procedure [6]. MLGen, a generative design procedure based on machine learning integrated with topology optimization, is presented herein. More specifically, MLGen is formulated by combining the topology optimization solution approach of solid isotropic material with penalization [7–9], long short-term memory networks [10], image filtering techniques and derivative-free search algorithms [11].

The layout of this study begins with a short introduction on structural topology optimization together with some notes on the solid isotropic material with penalization (SIMP) approach in Section 2, followed by a description of long short-term memory (LSTM) networks in Section 3. Subsequently, in Section 4 the proposed MLGen (machine learning-based generative design) framework is described in detail. Section 5 presents the results of the numerical tests performed in order to assess its efficiency to generate multiple optimized solutions.

2. Topology Optimization

In this part of the study, a short introduction of structural topology optimization (STO) is presented: the general mathematical concept used to formulate the STO problem, together with theory and implementation issues of the solution process. STO is a mathematical formulation of an optimization problem that aims to optimally distribute material within a specific design domain, subjected to given series of loading, boundary conditions and constraints. The goal of STO is to maximize the structural performance of the system. This is achieved by optimizing the allocation of material within the finite element mesh of the design domain.

2.1. Problem Formulation

The definition of the structural optimization problem requires introducing: a function (known as objective function) that refers to the criterion to be optimized, design unknowns (known as design variables), and state quantities. The criterion F that is to be optimized represents an objective metric that could either be maximized or minimized. Such an objective could be the material volume or the stiffness of the structural system. Moreover, the structural design domain and the state quantities associated to the criterion need to be defined. The design variable x represents unknown parameters that are required in order to describe the design of the structure; it could represent, for example, the geometry. The state quantity y represents the structural response quantities, which may represent strain, stress or displacement.

$$\begin{aligned} \min_{x \in S} F(x, y(x)) \\ subject \quad to: \\ state \quad constraints \quad on \quad y(x) \end{aligned} \tag{1}$$

where the objective function $F(x)$ (e.g., stiffness of the structure, and material volume) is to be optimized, and S is the design space of the unknowns' vector x. The state quantities depend on the design variable $y(x)$. The objective function is subjected to the design variables and state quantities constraints to lead the optimization process to the desired solution. A so-called state function $g(y)$ that corresponds to a certain state quantity can be devised, for example, as a deformation quantity along a certain degree of freedom (DoF). This state function $g(y)$ can be integrated as a constraint function to the structural optimization problem, where it is commonly expressed such that $g(y) \leq 0$, for example, the case where $g(y)$ is expressed by the displacement quantities vector $g(U(x))$ in a discrete finite element problem. In STO problems, the design variable x represents the presence or absence of material in a specific part of the design domain. It might involve features such as the size and/or the number of holes in the design domain. The general formulation of a STO problem is summarized below:

$$\begin{aligned} \min_{x\in[0,1]} F(x) \\ subject \quad to: \\ K(x)\cdot U(x) = P \\ g(x) \leq 0 \end{aligned} \tag{2}$$

where x refers the vector of unknowns, i.e., the density values of the finite elements, K denotes the global stiffness matrix of the structural system, vectors P and U contain the loads and displacements, respectively, and $g(x)$ refers to the vector of constraint functions (volume fraction, etc.).

2.2. *The Solid Isotropic Material with Penalization (SIMP) Approach*

In the past, several approaches have been proposed for solving the STO problem. The main ones are as follows [12]: (i) level-set method, (ii) density method, (iii) phase field method, (iv) topological derivative method, and (v) evolutionary method. SIMP, proposed in 90 s [7–9], is the most well-known representative of the density method and is commonly used for dealing with the STO problem. The structural system's compliance C is the most widely adopted performance indicator used in STO problem formulations. If n finite elements are used to discretize the design domain Ω, the distribution of material over Ω is denoted by x_i that are the density values of each finite element, $i \in [1,\ldots,n]$ and $x_i \in (0,1]$. When $x_i \approx 0$, it denotes no material to the ith finite element, while if $x_i = 1$, then the ith finite element is fully filled. Thus, Equation (2) becomes as follows:

$$\begin{aligned} \min_{x\in(0,1]} C(x) = U(x)^T \cdot P \\ subject \quad to: \\ K(x)\cdot U(x) = P \\ \frac{V(x)}{V_0} = V_t \end{aligned} \tag{3}$$

where $C(x)$ denotes the compliance of the structural systems for specific material distribution x, while $V(x)$, V_0 and V_t express the volume corresponding to density vector x, the initial volume for $x = x_0$ and the targeted volume of the optimized domain, respectively. According to SIMP, Young's modulus E is correlated based on a power law expression to the density value x_i of each element of the FE discretization as follows:

$$E_x(x_i) = x_i^p \times E_0 \iff K_x(x_i) = x_i^p \times K_{i,0} \tag{4}$$

where the penalization coefficient p is often set to $p = 3$, while $K_x(x_i)$ and $K_{i,0}$ denote the local stiffness matrix of the ith element when its density is equal to x_i and its original stiffness matrix, respectively. In the literature, various search algorithms were combined with SIMP for solving the problem of Equation (3); the method of moving asymptotes (MMA) and the optimality criteria (OC) algorithm are the most commonly used ones.

3. Long Short-Term Memory Networks

Long short-term memory (LSTM) networks were created as a variant of recurrent neural networks (RNNs) with a differentiated architecture. RNNs were firstly introduced in 1990 [13,14]. They were inspired by typical feedforward networks, but were tweaked in order to efficiently handle data that were in the form of time steps. In order to achieve that, RNNs are equipped with recurrent (feedback) connections as well. RNNs are able to operate not only on single data patterns (such as images), but also on sequences of data, such as video or speech. Due to this new architecture, RNNs presented the ability to efficiently handle time-series problems. A representation of an RNN is depicted in Figure 1 [15]. These networks were found to be difficult to train, as they were prone to numerical instability problems such as the exploding and/or vanishing gradient. This led to the development of the LSTM networks and the so-called bi-directional LSTMs [16] that are used in this study. LSTMs include an extra memory value that is trained on a specific number of time steps. This temporary memory acts as a hidden state that is cycled over the time steps.

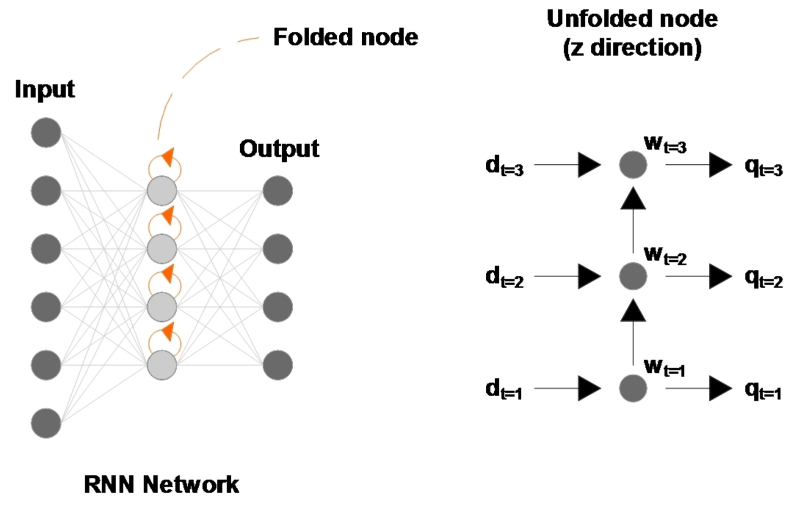

Figure 1. Recurrent neural network.

A typical LSTM architecture contains the input layer, a series of hidden layers (according to the depth of the network) and finally, the output layer. The size of the input layer depends on the problem and, specifically, the size of the input vector. The hidden layers present the special architecture of LSTM networks, the memory cells. These cells are characterized by three nodes, known as gates. The first gate, named input, is responsible for introducing a new value to the cell. The second one, called the forget gate, decides which previous state will be dropped by the network. The last one, known as the output gate, decides on the cell's output value. In a time series problem, whenever a new step is introduced to the network, all gates are fed with the new input along with the output value of the previous step that was stored in the networks' memory. Finally, the size of the output layer also depends on the problem. For example, in a classification problem with two classes, the size of the output layer would be equal to two [17]. As LSTMs were gaining more research attention, a new architecture of LSTMs was introduced, named bi-directional LSTMs. The novelty in this approach was that they include two LSTMs functioning at the same time but in reverse directions (forward and backward) and are connected by a merging function. Due to this form, the hidden state repeatedly extracts knowledge from both past and future information fed to the network [18]. Bi-directional LSTMs are very efficient when dealing with problems of time series, as they can take advantage of past and future input values simultaneously.

4. The MLGen Framework Description

This section presents the proposed methodology MLGen, together with its components. The goal of MLGen is to support the designing engineer with a tool able to automatically create a large number of equivalent designs that can act as an inspiration trigger. The proposed shapes are created via algorithms and it is ensured that all respect the goals and restrictions set by the user/designer. As a result, the combination of a number of methods is necessary. The proposed framework incorporates topology optimization but it is worth pointing out that the goal of MLGen is not to identify the optimal shape but to be able to discover a large number of differentiated designs that present near-to-optimal behavior. The methods used and the workflow of MLGen are presented in the following sections.

4.1. Framework Architecture

MLGen is a generative design framework that incorporates several methods in order to deliver automatically generated shapes. The methods used are a topology optimization algorithm (SIMP), neural networks (LSTM), image filtering, metaheuristics (ACO) and 3D printing support. The first three are used for generating a plethora of shapes, metaheuristics are used for visualizing the final result (not as optimizers) and 3D printing is for experimentally producing the outputs of MLGen. The flowchart of MLGen is presented in Figure 2.

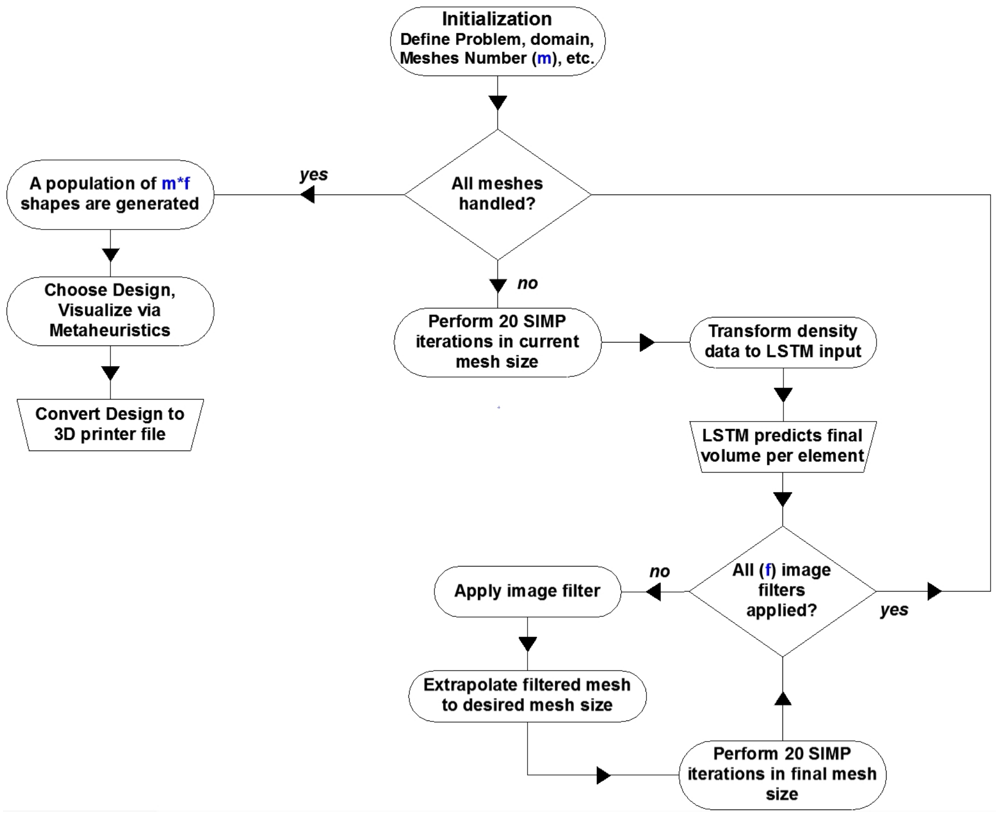
Initialization
Define Problem, domain, Meshes Number (m), etc.
All meshes handled?
yes
no
A population of m*f shapes are generated
Choose Design, Visualize via Metaheuristics
Convert Design to 3D printer file
Perform 20 SIMP iterations in current mesh size
Transform density data to LSTM input
LSTM predicts final volume per element
All (f) image filters applied?
no
yes
Apply image filter
Extrapolate filtered mesh to desired mesh size
Perform 20 SIMP iterations in final mesh size

Figure 2. MLGen flowchart.

The first step of executing MLGen is the initialization step. The user is called to define the problem by choosing the original mesh dimensions, FE population, loading and support conditions. Additionally the number $m-1$ and the FE population of less dense domains need to be decided as well. Once this step is completed, SIMP performs 20 iterations on all defined domains. Via this step, density data are exported for all FEs in the m domains and these data are then transformed according to Equation (7). The trained LSTM is then used for predicting the outcome of each FE of all domains used in the application. The output of the LSTM is an estimation of the final density distribution in all the m domains. Each of the m outputs is then filtered with the use of f number of image filters, such as Gaussian, edge detection, etc. Following that, each filtered domain is then extrapolated according to the FE center weight coordinates' proximity to the FE population of the desired output. Finally, the extrapolated domains are then passed into SIMP, which performs a maximum of 20 iterations for fine-tuning of the output. Once this procedure is finished, a population of $m \times f$ shapes is generated. Once all shapes are created and the designer chooses the one that is most preferable, a metaheuristic algorithm is used for visualizing the result design. Lastly, an algorithm is used for transforming the SIMP output into a type of file that can be used as an input for 3D printing. This formulation can be applied in 2D and 3D problems as well. As the volume classification is performed per finite element, it can be performed on both cases. What differentiates in the application in 2D and 3D models is the image filtering process. While for 2D cases, it is only performed once, in 3D cases it is executed z times, where z is the dimension of the mesh on the 3rd dimension. The z outputs of the filtering part are then joined to formulate the 3D shapes that are fine-tuned by SIMP.

4.2. Design and Training of the LSTM Network

In this part of the work, a detailed description of the network architecture, the training procedure and the database creation is presented. As previously stated, in the literature, LSTM networks are used successfully in several time-series classification problems. In this study, the scope for using the LSTM networks is to derive the density class of every element of the FE discretization. This class is defined according to the final one that the SIMP approach would have calculated, after multiple iterations, at convergence. It is worth mentioning that the prediction does not use information regarding the support conditions, the location of the element in the mesh of the design domain, the loading conditions or the desired volume fraction. The only information used is the density value of each element in the first 20 iterations of the SIMP approach in the original problem. This strategy is chosen in order to assure that the same trained network can offer high quality results without needing to be retrained regardless of the problem definition.

LSTMs present the ability to handle inputs that, apart from a single value per time-step, also include additional values (features) per time step. Assuming that SIMP is applied on a test example with a population of finite elements equal to $nx \times ny = m$ finite elements and a number of k iterations are performed until convergence is achieved, then a matrix D containing all densities $d_{i,j}$ per FE is formulated as follows [19]:

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & . & . & d_{1,k} \\ d_{2,1} & d_{2,2} & . & . & d_{2,k} \\ . & . & . & . & . \\ . & . & . & . & . \\ d_{m,1} & d_{m,2} & . & . & d_{m,k} \end{bmatrix} \tag{5}$$

It is noted that in the remaining part of this work, the density of an FE is defined as d rather than x, which is used in Section 2. In an attempt to improve the performance of the network, a decision is made to increase the features of the input instead of just using the density of one FE. The additional features that are used as inputs are the density values of neighboring finite elements of the element to be classified. In detail, apart from the density of the FE whose final density is to be classified, the densities of extra 24 FEs that are the closest to it in terms of center weights coordinates are also used as features of the networks' inputs. For example, in case of an element $e(i,j)$, the density recording input at the tth iteration is presented in Equation (6).

$$D^t_{i,j} = \begin{bmatrix} d_{i-2,j-2} \\ d_{i-2,j-1} \\ . \\ . \\ d_{i,j} \\ . \\ . \\ d_{i+2,j+1} \\ d_{i+2,j+2} \end{bmatrix} \tag{6}$$

The formulation of the database for m finite elements containing their density values over the optimization process until convergence, where the first t iterations of the SIMP approach out of the T are needed for convergence, is presented in Equation (7).

$$\begin{array}{c}
\left[\begin{array}{cccc|ccc|c}
D_1^1 & D_1^2 & \dots & D_1^t & D_1^{t+1} & \dots & D_1^{T-1} & d_1^T \\
D_2^1 & D_2^2 & \dots & D_2^t & D_2^{t+1} & \dots & D_2^{T-1} & d_2^T \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
D_{m-1}^1 & D_{m-1}^2 & \dots & D_{m-1}^t & D_{m-1}^{\prime t+1} & \dots & D_{m-1}^{T-1} & d_{m-1}^T \\
D_m^1 & D_m^2 & \dots & D_m^t & D_m^{t+1} & \dots & D_m^{T-1} & d_{m,T}
\end{array}\right] \\
m \quad InputVectors \qquad\qquad\qquad\qquad Output
\end{array} \tag{7}$$

A schematic representation of the features used can also be seen in Figure 3. It is worth pointing out that in the case of finite elements located close to the boundaries of the design domain in which there are no neighboring elements in one or more directions, padding of the necessary rows and/or columns is performed. The volume fractions of the generated FEs are set equal to the volume of their existing neighboring element.

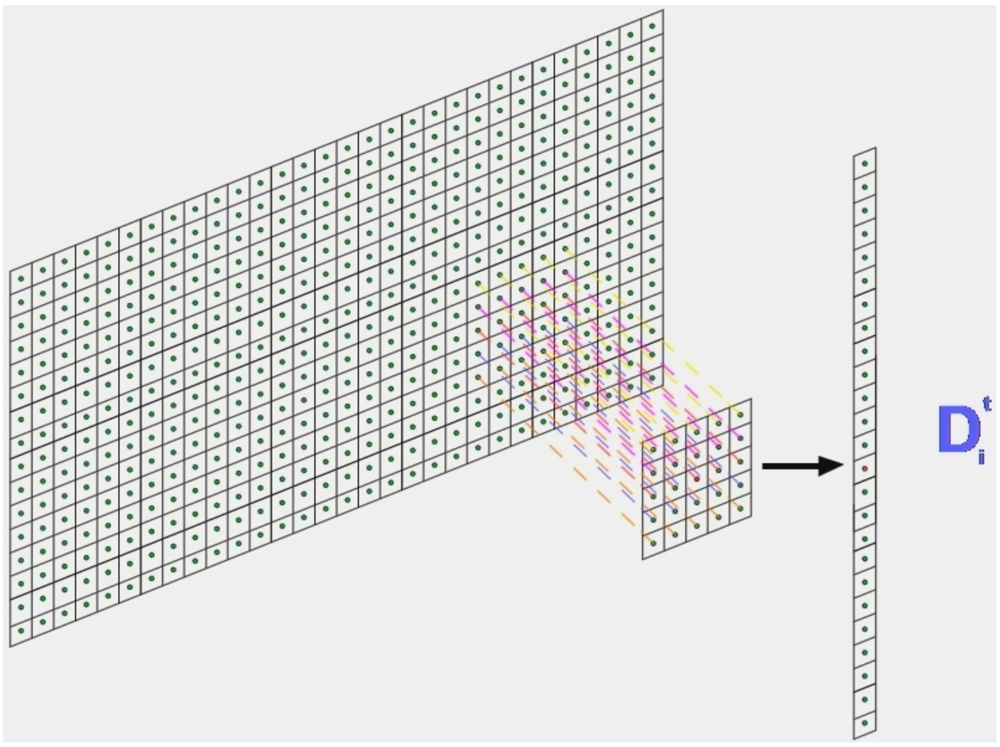

Figure 3. Creating density input vector.

As previously stated, the LSTM network is used for classifying the time series of the volume fluctuation of each finite element in the design domain. In order for the LSTM to be to able to perform this classification, the network has to be trained on a number of samples of finite elements. The training database is formulated by performing topology optimization via the SIMP approach on a specific problem with varying the FE mesh discretization in terms of its denseness. In detail, the test example chosen is the cantilever beam shown in Figure 4.

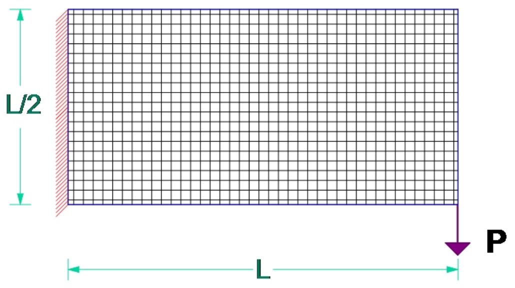

Figure 4. Creating density input vector.

The support condition can be described as fully fixed boundary conditions at the left part of the design domain along the vertical y axis, covering the height of the domain, while the loading conditions are a single concentrated load at the bottom right corner of the domain, facing towards the negative part of the y axis. The desired volume fraction is set equal to $V_t = 40\%$.

A population of 20 different FE mesh discretizations varying drastically on the number of the finite elements is chosen equal to [100, 500, 1000, 3000, 6000, 10,000, 20,000, 40,000, 60,000, 100,000] elements, and each mesh is generated with $ne_y/ne_x = 1/2$ and $ne_y/ne_x = 1/3$ height to length ratios. For each FE mesh discretization, a separate topology optimization problem is defined. After all problems are optimized by the SIMP approach, the density time series of all finite elements for all problems are saved and are divided into classes according to their final density value. The population of finite elements time series created is equal to 510,000 samples. The separation bounds of the classes are presented in the following Equation (8).

$$d_{i,k} \in \begin{cases} [0, 0.1] \Rightarrow d_{i,k} = 0 \\ (0.1, 0.9] \Rightarrow d_{i,k} = 0.50 \\ (0.9, 1] \Rightarrow d_{i,k} = 1 \end{cases} \tag{8}$$

The populations of the time-series samples in each class are equal to [240,000, 132,000, 138,000], respectively. The database is divided into a training and testing sample with an analogy of 80–20%. The first 20 steps of SIMP are used as the length of the network input. The network used in this study is a bi-directional LSTM network. The complete architecture is defined by five different layers: the input layer, the bi-directional LSTM, a fully connected layer, a soft-max layer and the output class layer. The network is built in Matlab and training is implemented both in CPU and GPU environments. In the training process, 10 epochs are run with 2000 iterations per epoch. The CPU and the GPU that are used for performing the training part are an Intel i9-9920x and an NVidia Titan RTX; in both cases, RAM is equal to 64 GB. In the case of CPU usage, the training needs 5500 s to complete, while in the case of GPU usage, the training time is reduced by a factor of more than 10, as it only needs 460 s. The accuracy achieved by both training procedures is equal to 97.3%.

4.2.1. Metaheuristics

In the past, metaheuristic algorithms were used as optimizers in several problems presented in the literature or in real life. One group of these algorithms are mainly inspired by nature mimicking procedures, such as ant colony optimization (ACO) [20], particle swarm optimization (PSO) [21], river formation dynamics (RFD) [22], pity beetle algorithm (PBA) [23], etc. In this work, the goal of using a metaheuristic algorithm is not to perform the actual optimization part, as this is handled by the SIMP approach supported by either OC [24] or MMA [25] algorithms. The goal of the metaheuristic algorithm is to assist in the visualization of the shape proposed by MLGen. In detail, information of the density of each finite element in the domain is used by the metaheuristic algorithm for deciding on which finite element to go to, moving from the current FE. For example, in the ACO algorithm, a number of ants are randomly placed inside the search space. On each step of the algorithm, each ant decides to move from a current node to another one according to the amount of pheromones on the trail connecting the two nodes. The amount of pheromones on the trail depends on the distance between the two nodes and the likelihood of this trail to be chosen by other ants as well [20].

In the MLGen implementation, particles (ants) are not randomly placed inside the search space (domain). They are placed on FEs that should definitely have the presence of particles. Such FEs are the ones where the loads and supports are placed. Instead of using pheromones, the attraction of a node depends on the density it presents according to SIMP and the likelihood of a connection to be chosen depends on the density value of the end node, divided by the normalized distance between the start and the end node. As a result, ACO is transformed into a design visualization method.

4.2.2. 3D Printing of MLGen Output

In order to produce the output of MLGen methodology, connecting the result exported by MLGen with a 3D printer is needed. For this reason, a bibliographic survey of methods for translating topology optimization results into files used by 3D printers is performed. The most common file types proposed are $*.stl$ and $*.obj$ files. Such methods were introduced for use with SIMP, BESO and level-set [3,26–28]. In the current work, the algorithm proposed by Liu [28] for exporting $*.stl$ files out of the results of the SIMP approach, named Top3DSTL, is selected, as it is found to be very efficient, while the methods used are more similar to the ones implemented in the MLGEn framework. Top3DSTL uses cubic representation and the export is in binary file format.

5. Numerical Tests

In order to validate the performance of the MLGen framework, five, known from the literature, benchmark test examples of topology optimization problems are selected. The selected test examples employ different loading and support conditions, while the volume fraction is also differentiated in each of them. In all of the test examples, seven models are used with the number of the FEs being equal to [1000, 5000, 7000, 10,000, 20,000, 25,000, 75,000] respectively. The iterations of SIMP in the original models are set equal to 20, while 23 different image filters are used. As a result, a population of 161 different shapes are generated for each test example with the population of the FEs being equal to 75,000. The final (fine-tuning) iterations of SIMP implemented for every design proposed out of the previous image filtering step is set equal to 20.

The definitions of dimensions along with the loading and support conditions of the five selected test examples are presented below. In addition, out of the different designs obtained 24 variants of the optimized designs are provided for each test example. For all five test examples, a sensitivity filter with radius equal to 3 is used for the SIMP implementation.

5.1. Test Example A

The design domain for the case of test example A can be seen in Figure 5 that is of rectangular shape with dimensions $L \times L/2$; it is supported by two fixed joints with the first one located in the bottom left corner on the x axis and the second one also placed on the bottom edge along the x axis at a distance of $0.6 \times L$ from the other support. The loading condition refers to a load distributed along the top edge of the x axis applied toward the vertical axis y. For this test example, the desired volume fraction is set equal to $V_t = 40\%$, while the structured FE discretization ratio is equal to $ne_y/ne_x = 1/2$. Out of the 161 optimized designs that are generated by means of the MLGen methodology for this test example, a randomly chosen sample of 24 optimized designs can be seen in Figure 6. As it can be observed, the MLGen methodology generates multiple different, but equivalent in terms of performance, optimized designs.

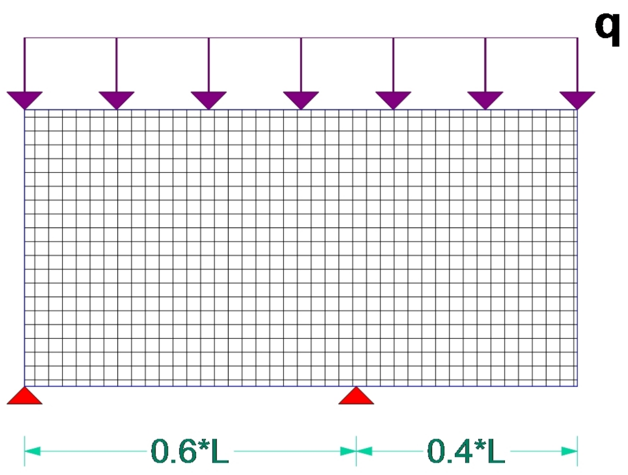

Figure 5. Definition of test example A.

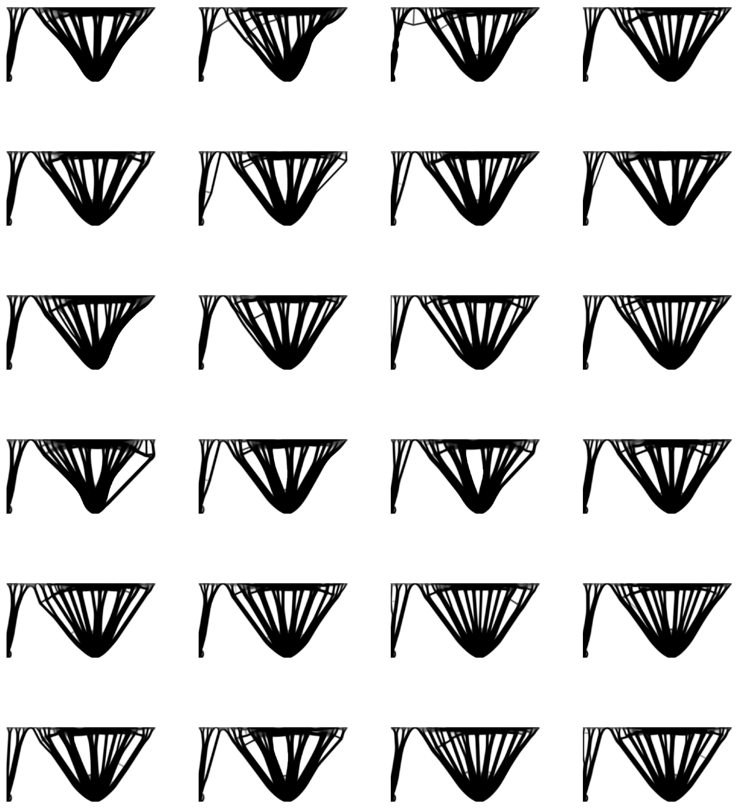

Figure 6. Results for test example A.

5.2. Test Example B

In test example B—the design domain that can be seen in Figure 7 that is of *lamda* shape with dimensions $L \times L/2$—fully fixed boundary conditions at the top left edge along the x axis are implemented, while a concentrated point load P at the right edge of the design domain is applied along the vertical y axis, defining the loading conditions. With respect to the structured FE discretization ratio, $ne_y/ne_x = 1/2$ while the target volume percentage is set equal to $V_t = 35\%$. A set of 24 randomly chosen images of the optimized results of this example as produced by MLGen can be seen in Figure 8. Similar to the previous test example, these are of equivalent value with respect to the design criteria chosen during the optimization problem formulation.

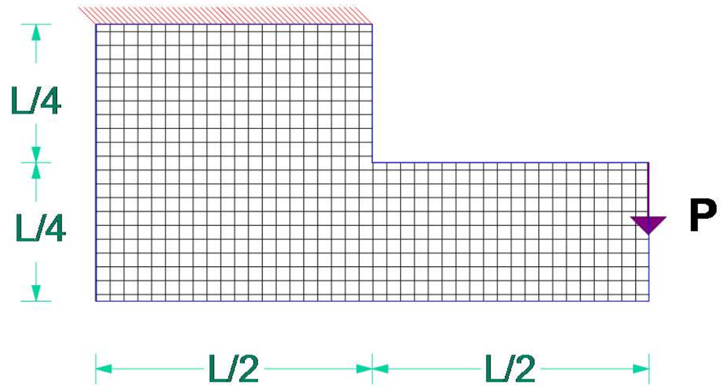

Figure 7. Definition of test example B.

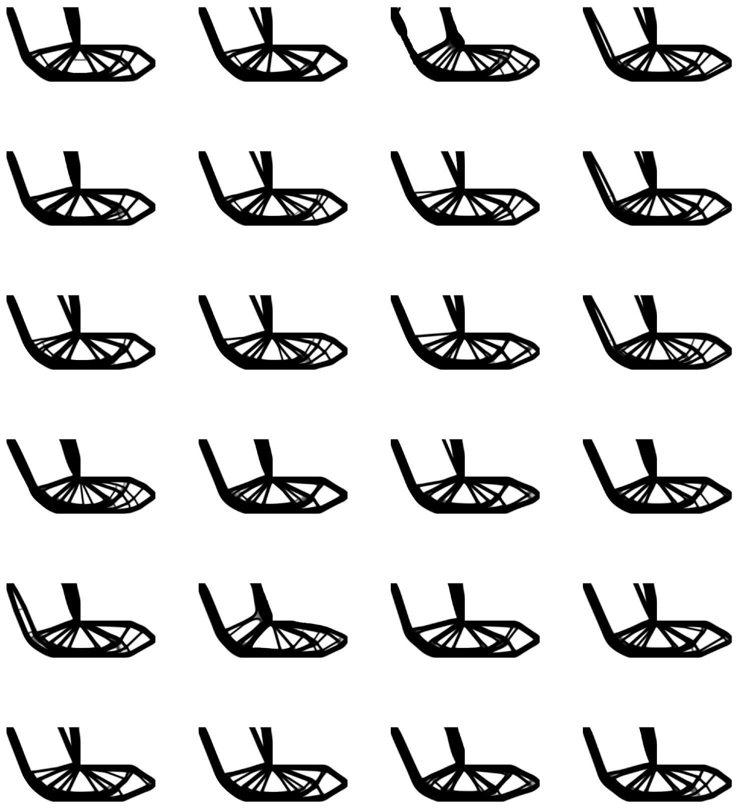

Figure 8. Results for test example B.

5.3. Test Example C

In Figure 9, the design domain that is also of rectangular shape with dimensions $L \times L/2$ and the corresponding loading and support conditions for the third test example C are presented. In detail, five fixed joints are located at the bottom edge of the design domain and they are equally distributed along the x axis. A load distributed along the horizontal x axis at the top edge of the design domain is applied toward the vertical y axis.The target volume percentage is set equal to $V_t = 25\%$, and the structured FE discretization ratio is equal to $ne_y/ne_x = 1/2$. The randomly chosen set of 24 optimized solution for this test example out of the 161 ones generated through the implementation of MLGen methodology is presented in Figure 10.

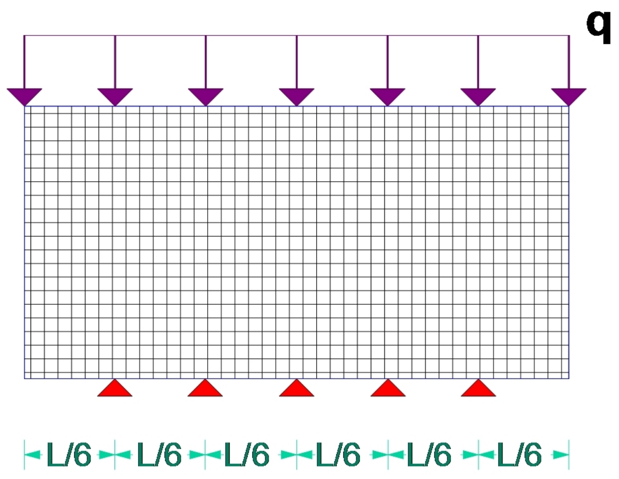

Figure 9. Definition of test example C.

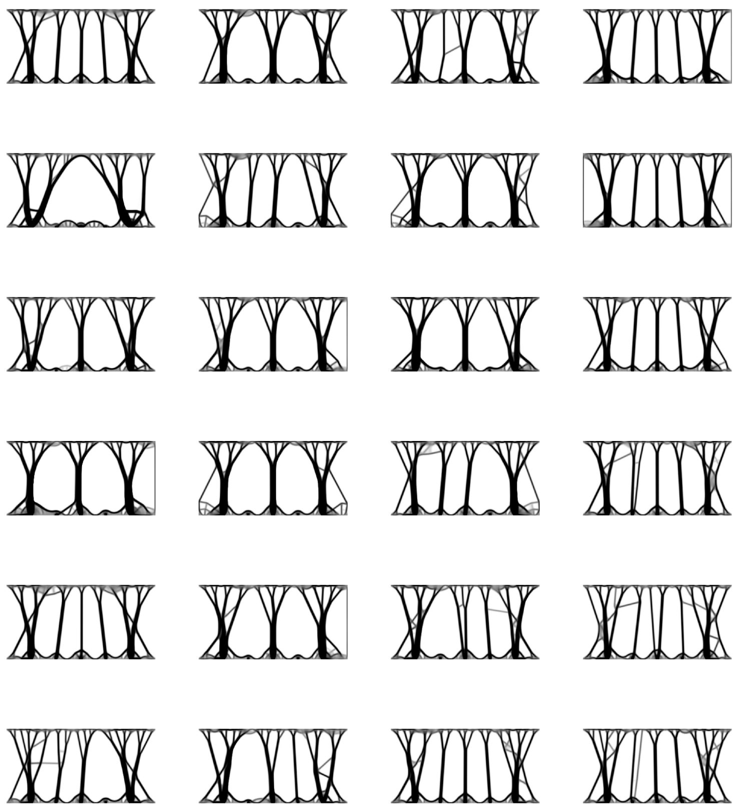

Figure 10. Results for test example C.

5.4. Test Example D

The next example is labeled test example D, and the corresponding design domain is presented in Figure 11 that is also of rectangular shape with dimensions $L \times L/2$. The support conditions correspond to two fixed points at the bottom edges located at the two opposite corners of the design domain, while the loading conditions refer to a concentrated load at the middle of the bottom edge applied along the vertical y axis. The desired volume fraction is equal to $V_t = 45\%$, and the structured FE discretization ratio is equal to $ne_y/ne_x = 1/2$. The corresponding set of randomly chosen optimized shapes by means of the MLGen methodology is presented in Figure 12.

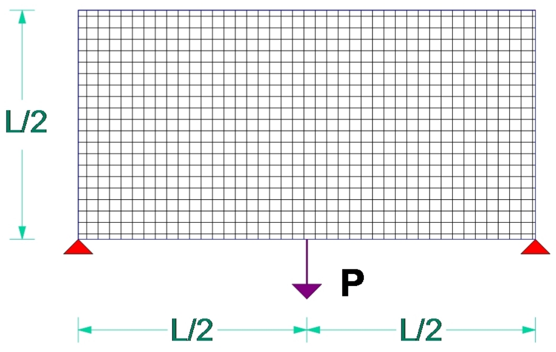

Figure 11. Definition of test example D.

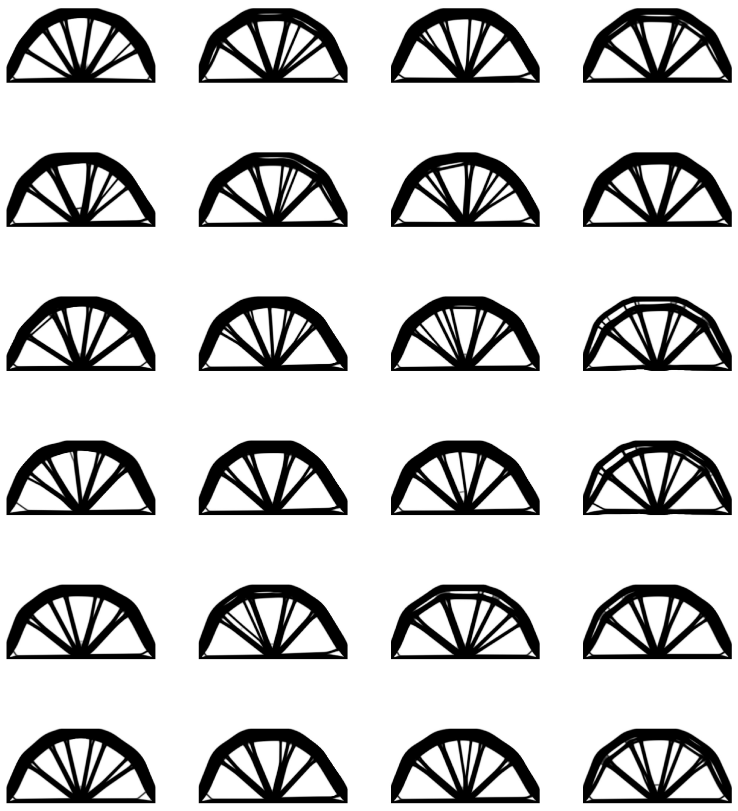

Figure 12. Results for test example D.

5.5. Test Example E

The final test example presented in this work can be seen in Figure 13, where the design domain is also of rectangular shape with dimensions $L \times L/2$. The loading conditions refer to two concentrated loads at the top and bottom right corners of the left edge of the design domain facing upwards and downwards, respectively. The support conditions are defined as fully fixed boundary conditions along the left vertical side of the domain. With respect to the structured FE discretization ratio, $ne_y/ne_x = 1/3$ while the final volume fraction is set equal to $V_t = 50\%$. A randomly chosen sample of the optimized results produced by means of the MLGen methodology is presented in Figure 14.

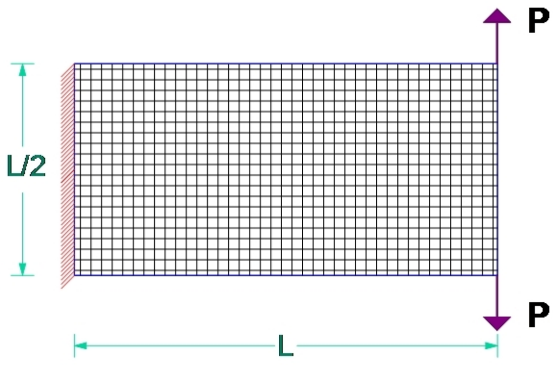

Figure 13. Definition of test example E.

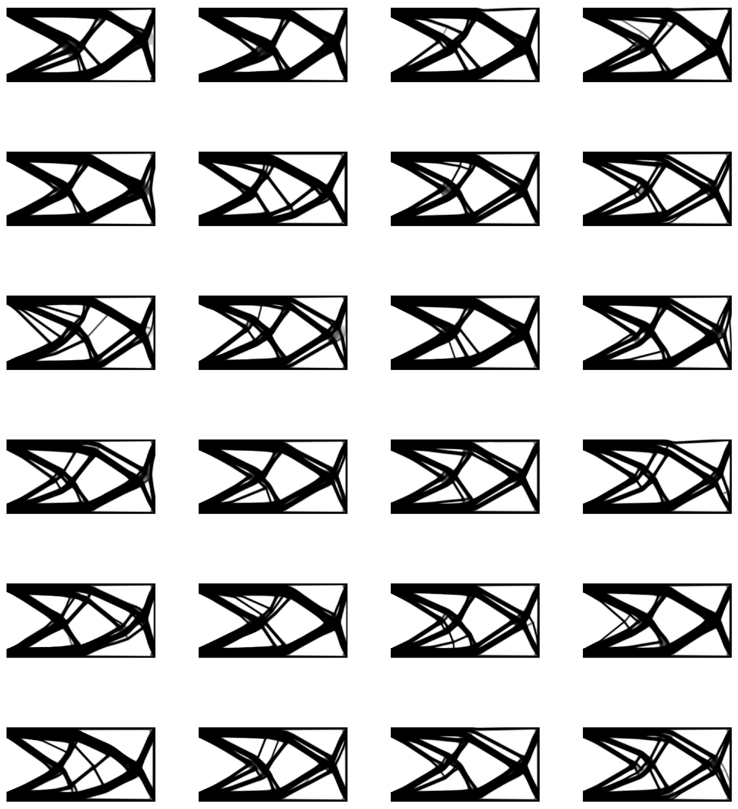

Figure 14. Results for test example E.

The parameters of all test examples previously described can be seen in Table 1.

Table 1. Test examples parameters.

Example	Load Type	Load Num.	Support Type	Support Num.	Volume
A	Distributed	1	Fixed joints	2	40%
B	Concentrated	1	Fully fixed	1	35%
C	Distributed	1	Fixed joints	5	25%
D	Concentrated	1	Fixed joints	2	45%
E	Concentrated	2	Fully fixed	1	50%

6. Discussion

In this work, the application of MLGen in several topology optimization problems with varying parameters is presented in order to validate its performance against differently defined problems in terms of loading and support condition, volume percentage, etc. As presented, MLGen is capable of creating a large number of different shapes regardless of problem parameters. MLGen is successful in generating differentiated shapes, regardless of the type and population of supports and loads used. It is also able to efficiently handle problems, even in the case of a very low volume percentage equal to 25% as in test example C or of a high volume percentage (50%) as presented in test example E. It is also worth pointing out that the above results are acquired in less than 3 min per generated shape. It is logical though that the time needed is analogous to the total number of finite elements in the mesh discretization of the final sample. It must also be noted that no symmetry is imposed in any of the test examples.

7. Conclusions

MLGen is proposed as a framework for performing automated shape generation under specific design rules and constraints, acting as a support tool for engineers and designers. To achieve that, the incorporation of several methods is needed. Topology optimization supported by its solution approach (SIMP) is used for generating a shape that satisfies all design goals and constraints set by the designer. Machine learning (LSTM) is used for accelerating the solution procedure along with imposing a shape differentiation. Image filtering is used for further increasing the number of shapes generated, while metaheuristics (ACO) are used for shape visualization. As the LSTM is trained on classifying density time-series of independent finite elements, the trained network can be applied to any example, regardless of the type and population of the loading conditions, mesh sizes, support conditions, SIMP filtering type and value, whether it is a 2D or 3D problem, the density of the FE mesh discretization, etc. There are several points for future research. The incorporation of different machine learning techniques could further increase the number of proposed shapes for each test example. It is also very important to work on reducing the necessary iterations of the SIMP approach both for the initial ones as well as the fine-tuning ones, as those iterations represent the biggest work load of the procedure. Additionally, the incorporation of other topology optimization approaches, such as level-set [29] and BESO [30], is also worth examining. Finally, a web application where MLGen can be freely used will be developed by the authors in the future.

Author Contributions: Conceptualization, N.A.K. and N.D.L.; methodology, N.A.K. and N.D.L.; software, N.A.K.; validation, N.A.K.; formal analysis, N.A.K.; investigation, N.A.K.; writing—original draft preparation, N.A.K. and N.D.L.; writing—review and editing, N.A.K. and N.D.L.; visualization, N.A.K.; supervision, N.D.L.; project administration, N.D.L.; funding acquisition, N.A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was co-financed by Greece and the European Union MIS-5033021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: This research is co-financed by Greece and the European Union (European Social Fund—ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the project "Reinforcement of Postdoctoral Researchers—2nd Cycle" (MIS-5033021), implemented by the State Scholarships Foundation (IKY).

Ευρωπαϊκή Ένωση
European Social Fund

Operational Programme
Human Resources Development, Education and Lifelong Learning
Co-financed by Greece and the European Union

ΕΣΠΑ
2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

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

Abbreviations

The following abbreviations are used in this manuscript:

RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
FE	Finite Element
SIMP	Solid Isotropic Material with Penalization
STO	Structural Topology Optimization
TO	Topology Optimization
TOP	Topology Optimization Problem

References

1. Lagaros, N.D.; Papadrakakis, M.; Kokossalakis, G. Structural optimization using evolutionary algorithms. *Comput. Struct.* **2002**, *80*, 571–589. [CrossRef]
2. Lagaros, N.; Plevris, V.; Papadrakakis, M. Multi-objective design optimization using cascade evolutionary computations. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3496–3515. [CrossRef]
3. Kazakis, G.; Kanellopoulos, I.; Sotiropoulos, S.; Lagaros, N.D. Topology optimization aided structural design: Interpretation, computational aspects and 3D printing. *Heliyon* **2017**, *3*, e00431. [CrossRef]
4. Lagaros, N.D.; Vasileiou, N.; Kazakis, G. A C# code for solving 3D topology optimization problems using SAP2000. *Optim. Eng.* **2018**. [CrossRef]
5. Papadrakakis, M.; Lagaros, N.D. Soft computing methodologies for structural optimization. *Appl. Soft Comput.* **2003**, *3*, 283–300. [CrossRef]
6. Zhang, Z.; Li, Y.; Zhou, W.; Chen, X.; Yao, W.; Zhao, Y. TONR: An exploration for a novel way combining neural network with topology optimization. *Comput. Methods Appl. Mech. Eng.* **2021**, *386*, 114083. [CrossRef]
7. Bendsøe, M.P. Optimal shape design as a material distribution problem. *Struct. Optim.* **1989**, *1*, 193–202. [CrossRef]
8. Zhou, M.; Rozvany, G. The COC algorithm, Part II: Topological, geometrical and generalized shape optimization. *Comput. Methods Appl. Mech. Eng.* **1991**, *89*, 309–336. [CrossRef]
9. Mlejnek, H. Some aspects of the genesis of structures. *Struct. Optim.* **1992**, *5*, 64–69. [CrossRef]
10. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; Technical Report; California Univ San Diego La Jolla Inst for Cognitive Science: La Jolla, CA, USA, 1985.
11. Fister, I., Jr.; Yang, X.S.; Fister, I.; Brest, J.; Fister, D. A brief review of nature-inspired algorithms for optimization. *arXiv* **2013**, arXiv:1307.4186.
12. Sigmund, O.; Maute, K. Topology optimization approaches. *Struct. Multidiscip. Optim.* **2013**, *48*, 1031–1055. [CrossRef]
13. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533. [CrossRef]
14. Schmidhuber, J. Netzwerkarchitekturen, Zielfunktionen und Kettenregel. Ph.D. Thesis, Technische Universität München, München, Germany, 1993.
15. Kallioras, N. Reduced Order Models and Machine Learning in Analysis and Optimum Design of Structures. Ph.D. Thesis, National Technical University of Athens, Athens, Greece, 2019.
16. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
17. Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.* **2018**, *270*, 654–669. [CrossRef]
18. Tai, K.S.; Socher, R.; Manning, C.D. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*; Association for Computational Linguistics: Beijing, China, 2015; pp. 1556–1566. [CrossRef]
19. Kallioras, N.A.; Kazakis, G.; Lagaros, N.D. Accelerated topology optimization by means of deep learning. *Struct. Multidiscip. Optim.* **2020**, *20*, 21–36. [CrossRef]
20. Dorigo, M.; Stutzle, T. *Ant Colony Optimization*; The MIT Press: Cambridge, MA, USA, 2004.
21. Kennedy, J.; Eberhart, R.C. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
22. Rabanal, P.; Rodríguez, I.; Rubio, F. Using river formation dynamics to design heuristic algorithms. In *International Conference on Unconventional Computation*; Springer: Berlin/Heidelberg, Germany, 2007, pp. 163–177.
23. Kallioras, N.A.; Lagaros, N.D.; Avtzis, D.N. Pity beetle algorithm—A new metaheuristic inspired by the behavior of bark beetles. *Adv. Eng. Softw.* **2018**, *121*, 147–166. [CrossRef]
24. Christensen, P.W.; Klarbring, A. *An Introduction to Structural Optimization*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 153.
25. Svanberg, K. The method of moving asymptotes—A new method for structural optimization. *Optim. Syst. Theory* **1987**, *24*, 359–373. [CrossRef]

26. Vogiatzis, P.; Chen, S.; Zhou, C. An Open Source Framework for Integrated Additive Manufacturing and Level-Set-Based Topology Optimization. *J. Comput. Inf. Sci. Eng.* **2017**, *17*, 041012. [CrossRef]
27. Brackett, D.; Ashcroft, I.; Hague, R. Topology optimization for additive manufacturing. In *2011 International Solid Freeform Fabrication Symposium*; University of Texas at Austin: Austin, TX, USA, 2011.
28. Liu, K.; Tovar, A. An efficient 3D topology optimization code written in Matlab. *Struct. Multidiscip. Optim.* **2014**, *50*, 1175–1196. [CrossRef]
29. Allaire, G.; Jouve, F.; Toader, A.M. A level-set method for shape optimization. *Comptes Rendus Math.* **2002**, *334*, 1125–1130. [CrossRef]
30. Querin, O.; Steven, G.; Xie, Y. Evolutionary structural optimisation (ESO) using a bidirectional algorithm. *Eng. Comput.* **1998**, *15*, 1031–1048. [CrossRef]