applied sciences

MDPI

Article

Deep Learning-Based Accuracy Upgrade of Reduced Order Models in Topology Optimization

Nikos Ath. Kallioras [1,†], Alexandros N. Nordas [2,†] and Nikos D. Lagaros [1,*,†]

[1] 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; kallioras.nikos@gmail.com

[2] Department of Civil, Environmental and Geomatic Engineering, ETH Zurich, 8093 Zurich, Switzerland; alexandros.nordas@igt.baug.ethz.ch

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

† These authors contributed equally to this work.

Abstract: Topology optimization problems pose substantial requirements in computing resources, which become prohibitive in cases of large-scale design domains discretized with fine finite element meshes. A Deep Learning-assisted Topology OPtimization (DLTOP) methodology was previously developed by the authors, which employs deep learning techniques to predict the optimized system configuration, thus substantially reducing the required computational effort of the optimization algorithm and overcoming potential bottlenecks. Building upon DLTOP, this study presents a novel Deep Learning-based Model Upgrading (DLMU) scheme. The scheme utilizes reduced order (surrogate) modeling techniques, which downscale complex models while preserving their original behavioral characteristics, thereby reducing the computational demand with limited impact on accuracy. The novelty of DLMU lies in the employment of deep learning for extrapolating the results of optimized reduced order models to an optimized fully refined model of the design domain, thus achieving a remarkable reduction of the computational demand in comparison with DLTOP and other existing techniques. The effectiveness, accuracy and versatility of the novel DLMU scheme are demonstrated via its application to a series of benchmark topology optimization problems from the literature.

Keywords: structural model order upgrading; topology optimization; DLTOP; deep belief networks

Citation: Kallioras, N.A.; Nordas, A.N.; Lagaros, N.D. Deep Learning-Based Accuracy Upgrade of Reduced Order Models in Topology Optimization. *Appl. Sci.* **2021**, *11*, 12005. https://doi.org/10.3390/app112412005

Academic Editor: Vladimir M. Fomin

Received: 16 November 2021
Accepted: 10 December 2021
Published: 16 December 2021

1. Introduction

As the size and complexity of numerical models used in structural analysis and design are continuously augmenting and there is no analogous increase in the available computing power, the advantages gained through the exploitation of soft computing techniques are often investigated. In the case of gradient-based optimization algorithms, where sensitivity analysis plays a significant role during the search process, up to 90% of the computations are devoted to the solution of the equilibrium equations:

$$P = K \times U \tag{1}$$

whereas the respective percentage exceeds 97% in metaheuristic search algorithms. In Equation (1), U and P denote the displacement and loading vectors, respectively, and K refers to the stiffness matrix of the structural system. Therefore, the solution of real-world, large-scale, complex structural optimization problems is a computationally demanding undertaking.

Structural optimization problems can generally be distinguished into three categories: sizing, shape and topology optimization [1–3]. The latter aims to identify the optimal topological arrangement of the material over the design domain of a structural system, for

a given set of load and support conditions. In principle, topology optimization supersedes sizing and shape optimization, in the sense that its output is also optimal with respect to sizing and shape criteria. The application of such approaches enables the establishment of structural systems that are virtually optimal in shape, thus providing an invaluable guide throughout the conceptual design phase [4,5].

Over the past decade, research on modern soft computing techniques drew significant attention [6], leading to the development of new methods able to handle large amounts of data and extremely complex problem definitions. The focus of the present study is on the application of soft computing methods in the fields of optimal analysis and design of structures, wherein recently there is a growing interest for exploiting deep learning techniques to accelerate topology optimization procedures. A detailed state-of-the-art review, along with a topology optimization via neural reparameterization framework, can be found in the work of Zhang et al. [7].

Some of the most computationally demanding problems in structural optimization are those that involve several solutions of the equilibrium equation (Equation (1)), as factorizing the stiffness matrix is time- and resource-consuming. To overcome this limitation, it is necessitated to resort to techniques that reduce the order of the stiffness matrix, minimize the necessary iterations of the search process or achieve a combination of these. This work aims to achieve both objectives, by virtue of establishing a novel model order reduction scheme and minimizing the required number of iterations for the topology optimization problem solution. The proposed scheme is labeled DLMU, which stands for "Deep Learning-based Model Upgrading", and is founded on the idea of using deep learning to extrapolate results of small- to large-scale models without compromising the solution quality.

The paper proceeds with providing an overview of deep learning procedures in Section 2, followed by a description of the accelerated topology optimization concept in Section 3. Subsequently, in Section 4 the proposed Deep Learning-based Model Upgrading (DLMU) scheme is described in detail, and Section 5 contains the results of the numerical tests performed to assess its efficiency in comparison with other acceleration techniques of the solution process of the topology optimization problem.

2. Deep Learning

Deep belief networks (DBNs) are regarded as the pioneering model of deep learning practices, as when first introduced [8,9] they were the first deep model that upon successful training managed to outperform Support Vector Machines (SVMs) in a classification problem [10]. DBNs represent generative models of probabilistic nature which contain a population of latent, stochastic parameters. These parameters are used as characteristics detectors of higher order correlations into the training/testing datasets.

The DBN is created when several Restricted Boltzmann Machines (RBMs) are stacked in a sequential manner [11]. This architecture is formed by the principle that the hidden layer of RBM_{i-1} becomes the seeable one for RBM_i. Furthermore, note that in DBNs, all layers have directed connections except for the last two which have undirected ones [12]. Let us consider a representation of such a network composed by five layers of size $L1$, $L2$, $L3$, $L4$ and $L5$, respectively, where a quadruple of RBMs construct a DBN. RBM_1 consists of layers $L1$ and $L2$ while the remaining RBMs are determined correspondingly. Based on the previous description, $L2$ is the hidden layer for RBM_1 while it constitutes also the visual one for RBM_2.

As mentioned before, training of deep networks of such architecture was not possible until the training methodology proposed by Hinton [10] was published. This methodology consists of a two-step training procedure: in the first one, termed as pre-training, each RBM of the DBN is trained according to an unsupervised manner, while in the second one, labeled as fine-tuning, the whole DBN undergoes supervised training [10,13]. Pre-training is performed by applying the contrastive divergence method in each RBM, sequentially. For RBM_1, the visible layer refers to the actual input while the hidden layer, once trained,

will act as the visual layer in the training of RBM_2 as described previously. The weight coefficients (w_{ij}) of every RBM's links are updated as follows:

$$\frac{\partial \log p(v;\theta)}{\partial w_{ij}} = \langle v_i h_j \rangle_{\text{input}} - \langle v_i h_j \rangle_{\text{model}}$$
$$w_{ij}^{new} = w_{ij} + e\Delta w_{ij} \tag{2}$$

where $p()$ denotes the state probability of the network; θ represent the set of weight and bias coefficients; v_i and h_j denote the state of ith and jth unit of the visible (v) and hidden (h) layer, respectively; and e denotes the weight learning rate and defines the range of weight changes. Fine-tuning is performed once pre-training is finished; the latter is performed by means of the back propagation algorithm [14] combined with the conjugate gradient algorithm [15].

A similar architecture to DBNs are the Deep Boltzmann Machines (DBMs), introduced in 2009 [16], which also are probabilistic graphical models with multiple layers of stochastic nodes. DBNs and DBMs share many common features in terms of architecture and functionality, where there are no connections between nodes of the same layer and the adjacent ones are fully connected, i.e., all nodes of a specific layer are linked to all nodes of the next and previous layers.

The main difference is the fact that all layers of a DBM are undirected. Apart from the first layer, also known as visible, all other layers of the network are hidden. A simple DBM with five layers can be seen in Figure 1. As energy-based models, an energy function is used for defining the joint probability distribution of the variables, similarly to DBNs [12]. The energy of a DBM like the one presented in Figure 1 can be calculated as follows:

$$\begin{aligned} E(v, h^1, h^2, h^3, h^4, h^5; \theta) = &-v^T W^1 h^1 \\ &- h^1 W^2 h^2 - h^2 W^3 h^3 \\ &- h^3 W^4 h^4 - h^4 W^5 h^5 \end{aligned} \tag{3}$$

while the probability assigned to input v is defined by the following expression:

$$p(v;\theta) = \frac{1}{Z} \sum_{h^1:h^5} e^{-E(v,h^1,h^2,h^3,h^4,h^5;\theta)} \tag{4}$$

where Z is calculated from the summation of all probabilities of existing pairs [1], θ represents the weights and biases of connections between layers of the network. The training of a DBM is accomplished with the use of a greedy layer-wise pre-training, as in the case of DBNs, by decomposing the DBM into RBMs and using back-propagation.

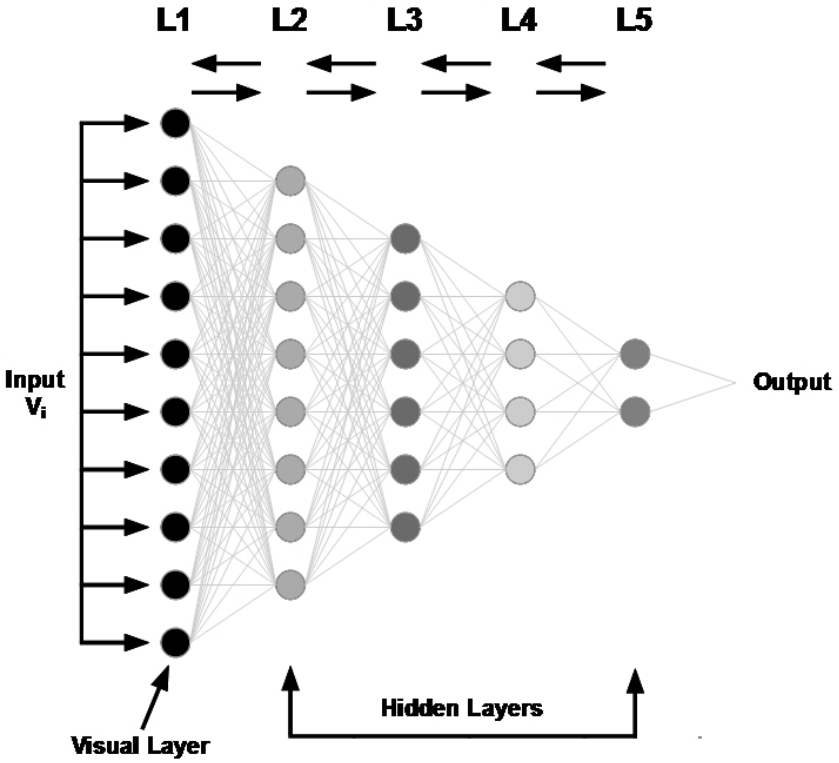

Figure 1. Deep Boltzmann Machine example.

3. Accelerated Structural Topology Optimization

Topology optimization (TO) aims at establishing the optimal material distribution over the design domain with reference to performance objectives (e.g., compliance) [17], subject to predefined loading and boundary conditions. Its application spectrum extends over several fields, including implants manufacturing [18], aerospace engineering [19], architectural engineering [20], design of materials [21], fluid mechanics [22], structural engineering [4] and others.

Several approaches have been proposed for solving TO problems, the main ones being [17] (i) Level-Set method, (ii) Density method, (iii) Phase field method, (iv) Topological derivative method and (v) Evolutionary method. SIMP is the most well-known variant of the density method, which was proposed in the 1990s [23–25]. The general formulation of a TO problem is summarized below:

$$\begin{aligned} &Minimize \quad F(x) \\ &subject \quad to: \\ &\qquad K \times U = P \\ &\qquad g(x) \leq 0 \\ &\qquad 0 \leq x \leq 1 \end{aligned} \tag{5}$$

where $F(x)$ denotes the criterion/objective (e.g., compliance of the structural system) to be optimized, x refers the vector of unknowns, i.e., the FEs densities, K is the stiffness matrix of the structural system, vectors P and U contain the loads and displacements, respectively and $g(x)$ is the vector of constraint functions (volume fraction, etc.).

3.1. The SIMP Approach

SIMP is conceivably the most commonly employed approach for solving TO problems. In structural topology optimization (STO) problems, the system compliance C is the most widely adopted performance indicator. If ne FEs are used to discretize the design domain Ω, the distribution of material over Ω is expressed via x_i, where $i \in [1, 2, \ldots, n]$ and

$x_i \in (0,1]$. x_i correspond to the density values of each FE: $x_i = 0$ indicates no material over the ith FE domain, while $x_i = 1$ indicates that the ith FE domain is filled with material. Therefore, Equation (5) can be written as follows:

$$\begin{aligned} Minimize \quad & C(x) = U(x)^T \times P \\ subject \quad & to: \\ & K(x) \times U(x) = P \\ & \frac{V(x)}{V_0} = V_t \\ & 0 < x \leq 1 \end{aligned} \tag{6}$$

where $C(x)$ is the compliance for the current material distribution x, and $V(x)$, V_0 and V_t, respectively, denote the volume values corresponding to the current and initial density vectors x and x_0, as well as the target volume value of the optimized domain. In SIMP, the Young modulus E is associated via a power law to the density value of each FE as follows:

$$E_x(x_i) = x_i^p E_0 \iff K_x(x_i) = x_i^p K^0 \tag{7}$$

where the penalization parameter p is usually set equal to $p = 3$. Thus, compliance can be expressed as follows:

$$\begin{aligned} & C(x) = U(x)^T \times P \iff \\ & C(x) = U^T(x) \times K(x) \times U(x) \iff \\ & C(x) = \sum_{i=1}^{n} x_i^p U_i^T K_i^0 U_i \end{aligned} \tag{8}$$

Accordingly, the formulation of Equation (6) can be expressed as

$$\begin{aligned} Minimize \quad & C(x) = \sum_{i=1}^{n} x_i^p U_i^T K_i^0 U_i \\ subject \quad & to: \\ & K(x) * U(x) = P \\ & \frac{V(x)}{V_0} = V_t \\ & 0 < x \leq 1 \end{aligned} \tag{9}$$

In the literature, various search algorithms have been used in conjunction with SIMP for solving the optimization problem of Equation (9). The most commonly employed ones are the Optimality Criteria (OC) algorithm and the Method of Moving Asymptotes (MMA).

3.2. The Deep Learning-Assisted Topology Optimization (DLTOP) Methodology

SIMP is one of the most widely employed, highly accurate and robust methodologies, applicable to a wide spectrum of TO problems. Nonetheless, in the field of STO, the required model scale, complexity and discretization level are continuously increasing, and thus the application of SIMP to such problems encounters bottlenecks, even in modern computing facilities, due to the substantial associated demand in computing time and resources.

This insufficiency of SIMP motivated the development of DLTOP by the authors. In cases of pattern recognition problems, DBNs are able to discover the different levels of representation of the data nonlinearity. This capability of DBNs motivated the recent development of the so-called Deep Learning-assisted Topology Optimization (DLTOP) methodology by the authors of [1], which enables reducing the iterations required by SIMP. The novelty of DLTOP lies in the use of deep learning techniques (DBNs) to propose

a close-to-final optimized configuration of a structural system, based upon the system configuration after only a few SIMP iterations. This eliminates the greatest portion of the required SIMP iterations to obtain the final optimized structural topology, thus resulting in a substantial reduction in the required computing time and resources, while overcoming potential bottlenecks that would otherwise be encountered. The novel DLMU methodology proposed in this paper partially relies on the idea of DLTOP; however, it is additionally assisted by information provided by reduced models of the design domain, thus allowing for further acceleration of SIMP. Before presenting the features and advantages of DLMU, it is useful to provide here a detailed description of DLTOP.

DLTOP is a combination of SIMP and DBNs, where a DBN is trained to transform the pattern of density fluctuation of FEs generated during the starting iterations of SIMP to the final distribution of the density values x_i over the design domain. DBN prediction capability is built based on a training procedure performed once over benchmark TO problems. The main advantage of DLTOP is that DBN needs not be trained again before being applied to any STO problem, irrespective of the FE mesh configuration and type (structured/unstructured), domain dimensions, target density, loading and boundary conditions, SIMP implementation features (e.g., filter value), etc. This key feature of DLTOP is attributed to its implementation such that every FE is handled separately, without requiring information regarding its location over the domain, its specific boundary and loading conditions, etc. The validation of DLTOP via several benchmark topology optimization test examples indicates a reduction of the number of iterations originally required by SIMP by more than one order of magnitude. Expectedly, the gain in computing time offered by DLTOP is proportional to the TO problem size. DLTOP is described hereafter with the use of a qualitative example.

Let us consider a design domain discretized with ne FEs. In every iteration t of SIMP, the density values x_i (that is also denoted as $d_i \equiv x_i$) are being updated for every FE. The fluctuation of the density value d_i of the i_{th} FE can be expressed as a function of t as follows:

$$d_i = f(t) \qquad \forall\, i \in [1, ne] \tag{10}$$

The density value fluctuation with respect to the SIMP iterations varies drastically for different FEs [1], due to their relative position over the domain with respect to loads, supports, etc. In this sense, each FE represents a different optimization history of density values with respect to the SIMP iterations, analogous to a discrete time-series. An initially uniform density value of 0.40 is specified for all FEs over the mesh, equal to the target volume ratio of the system of 40%; this constitutes common practice in SIMP implementation [26,27]. The computational demand of SIMP depends on the number of FEs. Considerable demands are posed even by moderately discretized domains, which becomes more pronounced in finer 3D meshes. As an example, the STO problem of a 3D bridge test case, discretized with a moderately dense mesh of 83,000 FEs, requires up to 7 h for performing 200 iterations of SIMP in serial CPU execution and up to 1 h in parallel GPU environment [4]. The DLTOP methodology can be applied to both serial or parallel, CPU or GPU execution implementations.

DLTOP can be seen as a two-phase methodology. In the first phase, SIMP performs a limited number of iterations; the histories of the density values generated during these iterations are used as input data for the DBN. At the end of the first phase, DBN proposes an optimized distribution of the density values over the design domain based on this input data. Upon evaluation of the input data by the trained DBN, the latter performs a discrete jump from the pool of density values of the initial iterations to a close to final density for every FE. Subsequently, as part of the second phase, SIMP carries out a series of limited additional iterations, which corresponds to a fine-tuning process over the DBN-proposed distribution of optimized densities. A schematic representation of the two-phase DLTOP procedure to a single, randomly selected FE is depicted in Figure 2, where the abscissa correspond to the iterations performed by SIMP and the ordinate to the specific FE densities.

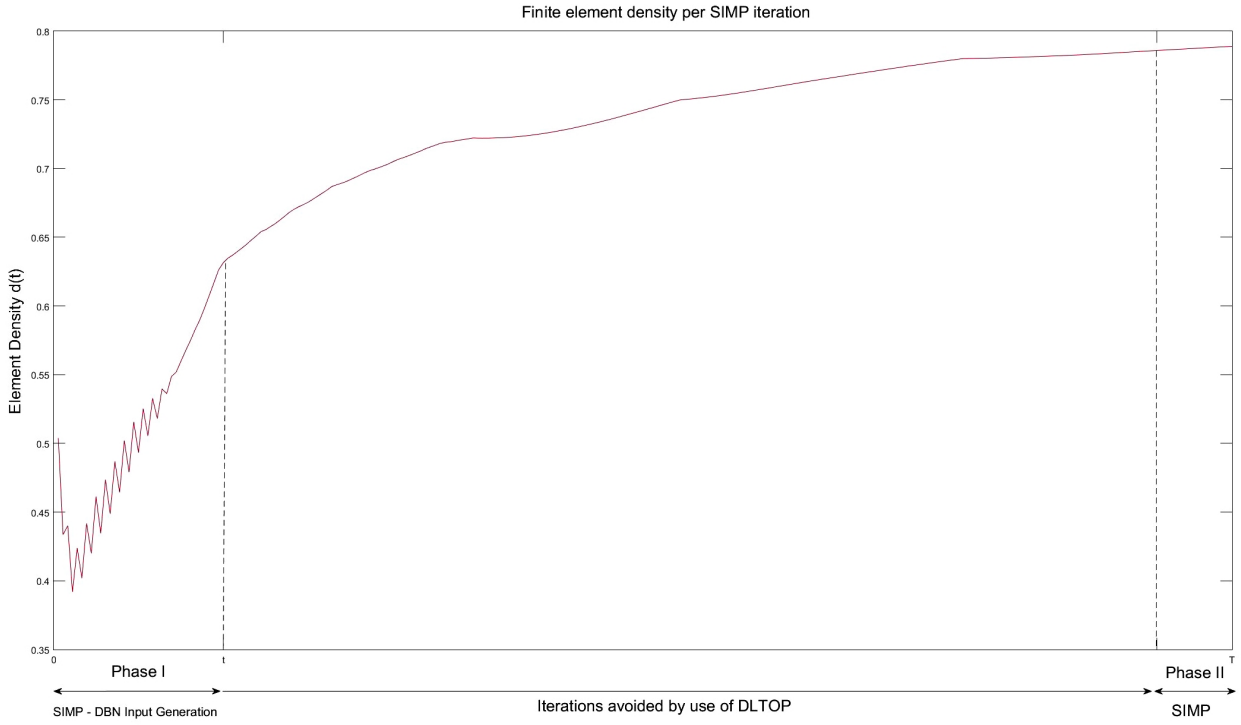

Figure 2. Implementation of the DLTOP methodology for a single finite element.

Classification represents a challenging problem class for predictive models. Comparing the well-known regression predictive models with the classification ones, it is worth mentioning that the latter require information related to the complexity of a sequence dependence amid input parameters. For the STO problem, the density values of the early SIMP iterations represent the sequence dependence information that needs to feed the proposed (classification) methodology. The sequence of discrete-time data, i.e., the density value for every FE and the maximum number of iterations T required for convergence by SIMP, are generated by SIMP and stored in a density matrix D, as shown below:

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,T} \\ d_{2,1} & d_{2,2} & \dots & d_{2,T} \\ \vdots & \vdots & \ddots & \vdots \\ d_{ne,1} & d_{ne,2} & \dots & d_{ne,T} \end{bmatrix} \tag{11}$$

A limited number of the iterations of the optimization procedure equal to the initial t iterations are used as time-series input data for training the DBN while the vector of the final SIMP iteration, i.e., the Tth column of D, is used as the target vector during the DBN training.

$$\left[\begin{array}{cccc|ccc|c} d_{1,1} & d_{1,2} & \dots & d_{1,t} & d_{1,t+1} & \dots & d_{1,T-1} & d_{1,T} \\ d_{2,1} & d_{2,2} & \dots & d_{2,t} & d_{2,t+1} & \dots & d_{2,T-1} & d_{2,T} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{ne-1,1} & d_{ne-1,2} & \dots & d_{ne-1,t} & d_{ne-1,t+1} & \dots & d_{ne-1,T-1} & d_{ne-1,T} \\ d_{ne,1} & d_{ne,2} & \dots & d_{ne,t} & d_{ne,t+1} & \dots & d_{ne,T-1} & d_{ne,T} \end{array}\right] \tag{12}$$

Training Sample | *Not Used* | *Target*

4. The Deep Learning-Based Model Upgrading (DLMU) Scheme

The novelty of DLTOP lies in the use of deep learning techniques (DBNs) to propose a close-to-final optimized configuration of a structural system, based upon the system

configuration after only a few SIMP iterations. This eliminates the greatest portion of the required SIMP iterations to obtain the final optimized structural topology, thus resulting in a substantial reduction in the required computing time and resources, while overcoming potential bottlenecks that would otherwise be encountered. Within the scope of further enhancing the computational efficiency of DLTOP, the authors envisioned a novel approach that combines deep learning with reduced-order modeling, which led to the development of the DLMU scheme presented in this paper. DLMU is based on DLTOP, however its novelty lies in the application of DLTOP to reduced-order models of the original system, instead of the system itself, and the subsequent extrapolation of their results to a full-scale optimized configuration of the original system, using the same deep learning techniques (DBNs). DLMU thus inherits the benefits of DLTOP and achieves equivalent accuracy, but substantially outperforms it in respect of computing time and demand in computing resources. In this section, the proposed DLMU scheme is described together with its implementation features.

Starting with a topology optimization problem formulated with the desired mesh density, five models with sparser meshes are generated, with identical structural and topology optimization problem characteristics. Each finite element (FE) of the dense mesh is linked with an element from each of the sparse meshes, based on their geometric center proximity, which enables generating volume-density data for each element of the dense mesh. Using two benchmark examples obtained from the literature, a deep belief network (DBN) is trained, by feeding as input the final optimized shape obtained for the five TOPs where the sparsely-meshed identical design domains were used, and as the target of the DBN training the per element volume of a densely meshed identical design domain.

Towards implementing the DLMU scheme on topology optimization problems, the user needs to apply the Solid Isotropic Material with Penalization (SIMP) approach assisted by deep learning [1] on five identical but sparsely-meshed problems. The final volumes per element and per mesh are fed as input to the previously trained DBN and the network proposes an almost final volume value for each element of the dense meshed problem. This output is then fine-tuned by the SIMP approach, by performing a limited number of iterations. Note that the population of fine-tuning iterations is not user-defined, but rather determined by the quality of the DBN output. The SIMP termination criterion is the same with the typical SIMP approach. By upgrading reduced order models without the requirement to retrain the DBN, DLMU achieves a substantial reduction in the computing time of topology optimization problems, regardless of the problem formulation, as will be shown in several numerical tests presented in the numerical tests section.

As a result, SIMP can be applied on a reduced order model until the termination criterion is reached. The output is read by the trained Convolutional Neural Network (CNN) which enhances the "resolution" of the SIMP results and provides a close to final result for the same problem but with a significantly finer mesh. This CNN output is fine-tuned by SIMP. As evident in all 2D test cases examined in numerical tests section, the application of CNN-assisted procedures in various topology optimization problems leads to drastic reduction in the required time without compromising the quality of the final result. Again, the population of fine-tuning iterations is not user-defined, but rather a result of the quality of the CNN output.

4.1. The Methodology

An STO problem TOP^f, where subscript f stands for the fully refined FE model, is defined by its problem properties $[L^f_{xyz},\ ne^f,\ S^f_c,\ P^f_c,\ V^f_t]$, which, respectively, stand for the design domain dimensions, number of FEs adopted for the discretization of the model to be optimized, support conditions, loading conditions and target volume ratio, where in cases of a structured mesh $ne^f = ne^f_x \times ne^f_y \times ne^f_z$. STO problems, associated with models of reduced order with respect to the number of FEs (ne) and identical properties otherwise, can be described as $TOP^R_{(i)}:\ [L^f_{xyz},\ ne^R_{(i)},\ S^f_c\ P^f_c,\ V^f_t]$, where superscript R stands for the

reduced order model and subscript i denotes the *id* of the reduced order model. The core part of the DLMU scheme is described hereafter.

4.1.1. Step 1: Solving the STO Problem Based on Reduced Order Models

Let us assume that TOP^f denotes the STO problem that is to be solved. The first step involves the generation of a number n_{ROMs} of reduced order models (ROMs) (five in this study) along with the formulation of the corresponding ($TOP^R_{(i)}$) problems ($i = 1, 2, \ldots, n_{ROMs}$). With regard to the problem formulation, the only difference between each one of them and TOP^f is the number of FEs used for discretizing the design domain. The number of FEs used for the $n_{ROMs} = 5$ problems follows the rule

$$ne^R_1 < ne^R_2 < ne^R_3 < ne^R_4 < ne^R_5 << ne^f \tag{13}$$

A set of matrices $D^R_{(i)}$, ($i = 1, 2, \ldots, n_{ROMs}$), contains the list of the domains generated during solution of the corresponding ($TOP^R_{(i)}$) problems. The solution is performed by means of the DLTOP methodology, which is implemented independently for each of the n_{ROMs} STO problems of the ROMs. For each ($TOP^R_{(i)}$) problem the procedure is terminated when DLTOP converges, i.e., the termination criterion (percentage of change) of the refinement part performed by SIMP is fulfilled. The output consists of the optimized domain $D^R_{(i),OPT}$ for each one of the n_{ROMs} STO problems. Without loss of the generality of the DLMU scheme, let us assume that the STO problem TOP^f to be solved refers to a 2D design domain discretized with a structured mesh, and thus the optimized domains $D^R_{(i),OPT}$ of the $TOP^R_{(i)}$ problems can be described as follows:

$$D^R_{(i),OPT} = \begin{bmatrix} d^{opt}_{1,1} & d^{opt}_{1,2} & \cdots & d^{opt}_{1,ne^R_{(i),x}} \\ d^{opt}_{2,1} & d^{opt}_{2,2} & \cdots & d^{opt}_{2,ne^R_{(i),x}} \\ \vdots & \vdots & \ddots & \vdots \\ d^{opt}_{ne^R_{(i),y},1} & d^{opt}_{ne^R_{(i),y},2} & \cdots & d^{opt}_{ne^R_{(i),y},ne^R_{(i),x}} \end{bmatrix} \tag{14}$$

where $d^{opt}_{l,k}$, with $l = 1, 2, \ldots, ne^R_{(i),y}$ and $k = 1, 2, \ldots, ne^R_{(i),x}$, denotes the material density of the (l, k) element of the $ne^R_{(i),x} \times ne^R_{(i),y}$ structured mesh.

4.1.2. Step 2: Definition of the Initial Design

In the second step, the initial design $D^f_{(0)}$ that will be used for solving the TOP^f problem is created based on the $D^R_{(i),OPT}$ by means of extrapolation, for every $i = 1, 2, \ldots, n_{ROMs}$, where

$$D^f_{(0)} = \begin{bmatrix} d^{(0)}_{1,1} & d^{(0)}_{1,2} & \cdots & d^{(0)}_{1,ne^f_x} \\ d^{(0)}_{2,1} & d^{(0)}_{2,2} & \cdots & d^{(0)}_{2,ne^f_x} \\ \vdots & \vdots & \ddots & \vdots \\ d^{(0)}_{ne^f_y,1} & d^{(0)}_{ne^f_y,2} & \cdots & d^{(0)}_{ne^f_y,ne^f_x} \end{bmatrix} \tag{15}$$

In particular, every optimized design $D^R_{(i),OPT}$ is used for generating an optimized topology of D^f via interpolation from scale $ne^R_{(i)}$ to ne^f. This provides a dataset D_{TOP} of n_{ROMs} optimized topologies for the TOP^f problem, which can be represented as

$$D_{TOP} = \begin{bmatrix} T_{1,1}^{(d)} & T_{1,2}^{(d)} & \dots & T_{1,ne_x^f}^{(d)} \\ T_{2,1}^{(d)} & T_{2,2}^{(d)} & \dots & T_{2,ne_x^f}^{(d)} \\ \vdots & \vdots & \ddots & \vdots \\ T_{ne_y^f,1}^{(d)} & T_{ne_y^f,2}^{(d)} & \dots & T_{ne_y^f,ne_x^f}^{(d)} \end{bmatrix} \tag{16}$$

where

$$T_{i,j}^{(d)} = \left[\{d_{k,l}\}_{TOP_1^R}^{opt}, \{d_{m,n}\}_{TOP_2^R}^{opt}, \dots, \{d_{b,c}\}_{TOP_5^R}^{opt} \right], \tag{17}$$

and

$$i \in [1, ne_y^f],\ j \in [1, ne_x^f],\ k \in [1, ne_{1,y}^R],\ l \in [1, ne_{1,x}^R], \dots, b \in [1, ne_{5,y}^R],\ c \in [1, ne_{5,x}^R] \tag{18}$$

As can be observed in Equation (16), ne^f vectors $T_{i,j}^{(d)}$ are generated, containing the density values of the FE in each $D_{(i),OPT}^R$ that is closest in terms of centroidal distance to the FE of TOP^f that $T_{i,j}^{(d)}$ describes.

4.1.3. Step 3: Prediction

In the last step of the DLMU scheme, a DBN is implemented for deriving a final density value for each of the ne^f FEs of the TOP^f problem, based on its $T^{(d)}$ information accompanied by a convolution step. The output of the DBN-convolution procedure is used as input for the refinement that is implemented by SIMP, where the required iterations for convergence are performed. The schematic representation of the DLMU scheme is given in Figure 3. In order for DBN to be able to discover a correlation between $T^{(d)}$ of each FE and its final density value in TOP_{OPT}^f, a calibration procedure needs to be performed after first creating a training dataset.

4.2. DBN Calibration—Training Dataset

The training dataset required for training the DBN used in the 3rd step of the DLMU scheme consists of the results taken for two benchmark test problems presented in a recent study by the authors [1]: a cantilever beam and a simply supported beam. The required form of the dataset is described in Equation (19).

$$\begin{array}{lcl} \left[d_{TOP_1^R}^{opt}, d_{TOP_2^R}^{opt}, d_{TOP_3^R}^{opt}, d_{TOP_4^R}^{opt}, d_{TOP_5^R}^{opt} \right]^{(1)} & \rightarrow & d_{1,TOP^f} \\ \left[d_{TOP_1^R}^{opt}, d_{TOP_2^R}^{opt}, d_{TOP_3^R}^{opt}, d_{TOP_4^R}^{opt}, d_{TOP_5^R}^{opt} \right]^{(2)} & \rightarrow & d_{2,TOP^f} \\ \vdots \quad \vdots \quad \ddots \quad \vdots & & \\ \left[d_{TOP_1^R}^{opt}, d_{TOP_2^R}^{opt}, d_{TOP_3^R}^{opt}, d_{TOP_4^R}^{opt}, d_{TOP_5^R}^{opt} \right]^{(ne^f-1)} & \rightarrow & d_{ne^f-1,TOP^f} \\ \left[d_{TOP_1^R}^{opt}, d_{TOP_2^R}^{opt}, d_{TOP_3^R}^{opt}, d_{TOP_4^R}^{opt}, d_{TOP_5^R}^{opt} \right]^{(ne^f)} & \rightarrow & d_{ne^f,TOP^f} \\ Input & & Target \end{array} \tag{19}$$

For each of the two test examples, the number of FEs required for the formulation of the fully refined TOP^f problem is chosen as 200,000. Additionally, consideration is given to five different combinations of reduced order models and corresponding $TOP_{(i)}^R$ problems.

The respective number of FEs for each combination of $TOP_{(i)}^{R}$ problems is given in the matrix of Equation (20) for test example m:

$$DAT_m = \begin{bmatrix} 1000 & 2000 & 3000 & 4000 & 5000 \\ 2000 & 3000 & 4000 & 5000 & 6000 \\ 4000 & 5000 & 6000 & 7000 & 8000 \\ 2000 & 5000 & 7000 & 8000 & 10{,}000 \\ 6000 & 7000 & 8000 & 9000 & 10{,}000 \end{bmatrix} \quad (20)$$

where $m = 1$ for the cantilever beam and $m = 2$ for the simply supported beam. Both matrices DAT_1 and DAT_2 are combined to formulate the training dataset, which consists of nearly 1,500,000 training patterns of $[T^{(d)} \rightarrow d_{i,TOPf}]$. Once the calibration is completed, the DLMU scheme can be applied for solving any STO problem without the need for recalibration, as density fluctuations of any FE with respect to model upgrading are examined. The performance of the DLMU scheme is evaluated against five well known test examples in the next section.

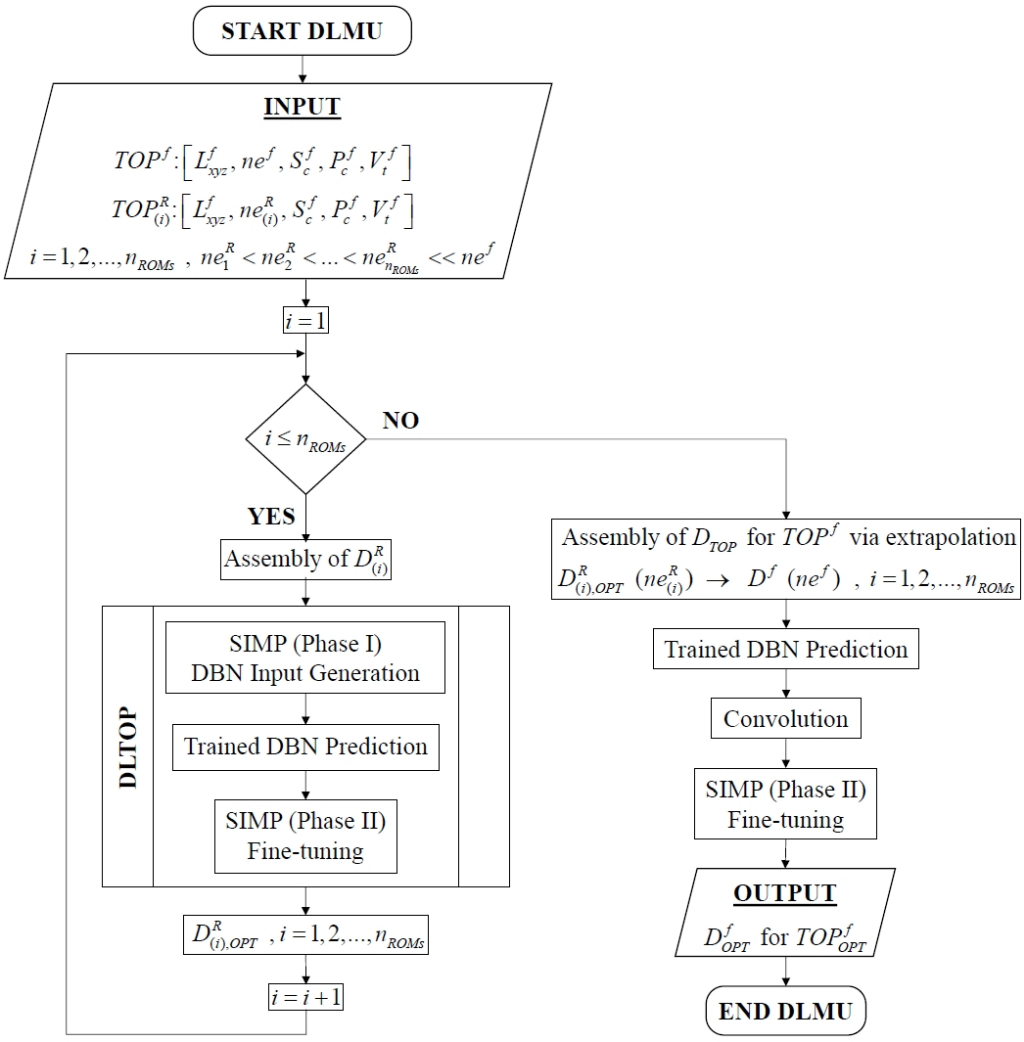

Figure 3. Flowchart of the DLMU scheme.

5. Numerical Tests

The performance evaluation of the DLMU scheme is conducted based on five 2D test examples (see Figure 4) taken from the literature. The topology optimization is performed using the SIMP approach and record is kept of the necessary iterations for convergence, objective function value and execution time. Without loss of generality, the compliance is

used as the objective function to be minimized. The same test examples are also optimized with the use of the proposed DLMU scheme and the same records are kept. As on the basis of the DLMU scheme iterations are performed on domains with less dense meshes, the comparison is based solely on the execution time, whereas iterations are mentioned for completeness and refer to those performed for the fine meshes. In the case of SIMP, the total time from the domain definition up to the end of the final iteration is recorded. In the case of the DLMU scheme, the time from the definition of $D^{R}_{(i),OPT}$ to the end of the final iteration of $D^{f}_{(0)}$ is recorded. The DLMU scheme requires the formulation of five domains with reduced number of FEs per test case for creating $D^{R}_{(i),OPT}$, where the specified $ne^{R}_{(i)}$ are equal to [5000, 7000, 10,000, 15,000, 20,000], respectively. Regarding the number of FEs in the final domain of each test example, four different cases are examined to evaluate the performance of DLMU scheme: [75,000, 100,000, 150,000, 200,000]. At the last part of the comparative study the DLMU scheme is additionally compared with DLTOP [1] and DL-SCALE [28] methodologies for the case of 200,000 FE. The five test-examples are described hereafter.

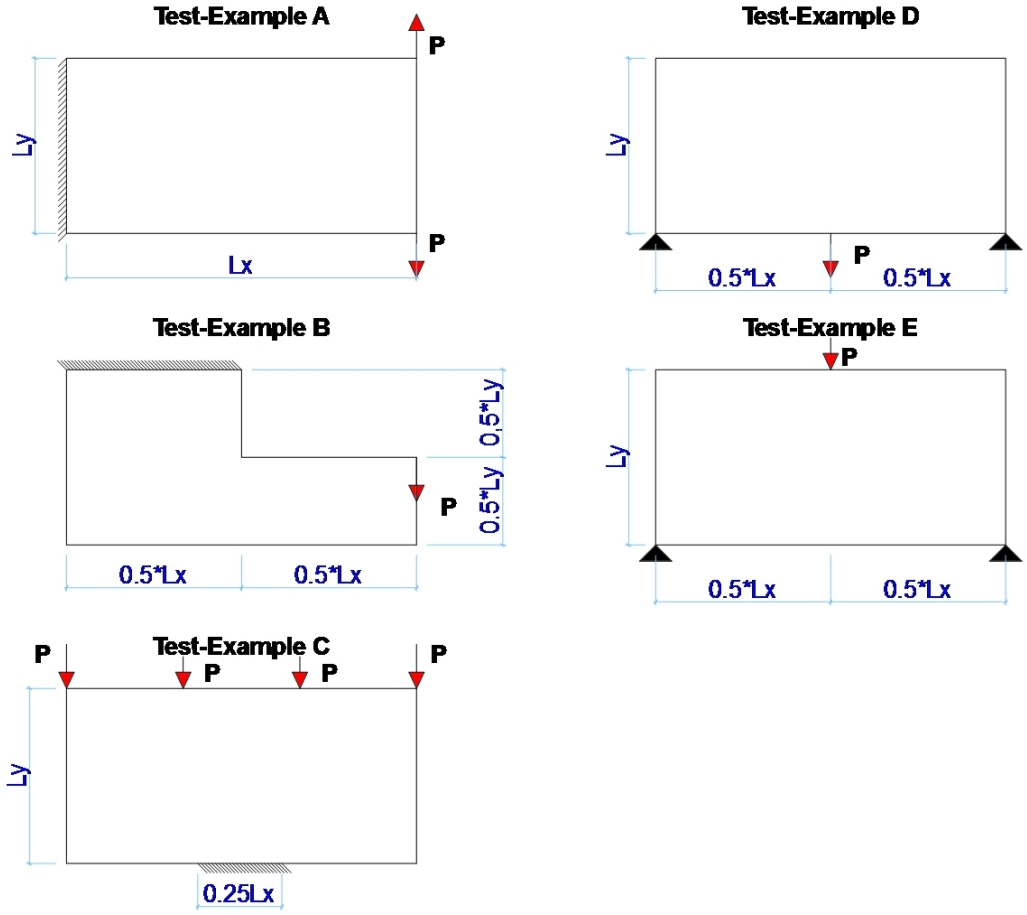

Figure 4. Schematic representation of five test examples.

Regarding the computational environment, it should be noted that both SIMP runs and network built were performed in Matlab, SIMP runs were implemented in CPU and training was implemented both in CPU and GPU environment. The CPU and the GPU that were used for performing the training part are an Intel i9-9920x and an Nvidia Titan RTX while RAM was equal to 64 Gb.

5.1. Description

The geometric configuration of the five design domains, along with the boundary and loading conditions considered for the test examples are presented in Figure 4. Without

loss of the generality of the proposed scheme, a structured FE mesh discretization was used, the corresponding ratio ne_y/ne_x and the target volume fraction V_t for each of the test examples are as follows. Test example A: $ne_y/ne_x = 1/3$, $V_t = 30\%$. Test example B: $ne_y/ne_x = 1/2$, $V_t = 30\%$. Test example C: $ne_y/ne_x = 1/2$, $V_t = 45\%$. Test example D: $ne_y/ne_x = 1/3$, $V_t = 30\%$. Test example E: $ne_y/ne_x = 1/3$, $V_t = 30\%$. For all five test examples, a sensitivity filter was used for the SIMP implementation with radius equal to 3.

5.2. Performance of the DLMU Scheme

This section discusses the performance of the DLMU scheme in the five test-examples, for the five fully refined discretizations considered, i.e., [75,000, 100,000, 150,000, 200,000]. For each test example, the performance of the DLMU scheme is compared with the corresponding implementation of SIMP, with respect to the time and number of iterations required for the fully refined discretization, as well as the solution quality, as inferred by the objective function value. For the finer discretization with 200,000 FEs, a comparative evaluation of the DLMU scheme and the DLTOP methodology is further conducted. The computing effort required by DLMU refers to the time needed for implementing Steps 1, 2 and 3 of the scheme. All recorded data for test example A are given in Table 1. DLMU achieved a maximum reduction of computational time equal to 80.64% for the case of 200,000 FEs in $D^f_{(0)}$ with respect to SIMP, while the objective function value slightly increased by 0.73%. The final optimized topologies generated by SIMP and DLMU for the finer discretization with 200,000 FEs are shown in Figure 5a,b, respectively.

Table 1. Test example A—Performance of the DLMU scheme.

ne	SIMP			DLMU			Acceleration (%)	Reduction (%)
	Iterations	Objective	Time (s)	Iterations	Objective	Time (s)		
75,000	251	248.41	404.89	80	249.40	219.40	45.87	−0.40
100,000	331	246.46	736.53	61	247.37	224.06	69.58	−0.37
150,000	340	243.52	1203.95	72	245.15	337.66	71.95	−0.67
200,000	521	242.62	2559.05	86	244.39	495.55	80.64	−0.73

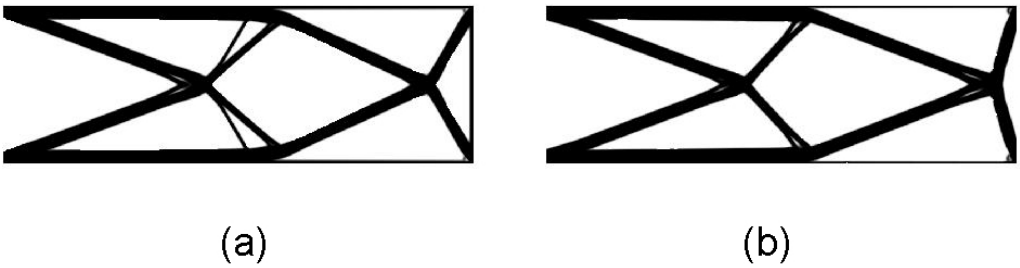

Figure 5. Optimized domain for test example A for the case of 200,000 FEs discretization: (**a**) SIMP output and (**b**) DL-SCALE output.

Accordingly, all recorded data for test example B are given in Table 2. Similar to the previous case, DLMU achieved a maximum reduction of computational time equal to 56.91% for the case of 200,000 FEs in $D^f_{(0)}$ with respect to SIMP, while a decreased objective function value by 0.58% was achieved. The final optimized topologies generated by SIMP and DLMU for the finer discretization with 200,000 FEs are shown in Figure 6a,b, respectively.

Table 2. Test example B—Performance of the DLMU scheme.

ne	SIMP			DLMU			Acceleration (%)	Reduction (%)
	Iterations	Objective	Time (s)	Iterations	Objective	Time (s)		
75,000	401	159.86	505.88	79	158.83	292.43	42.19	0.64
100,000	444	158.44	774.02	58	157.83	291.76	62.31	0.38
150,000	514	156.35	1438.26	98	155.73	461.50	67.91	0.40
200,000	367	156.01	1382.55	107	155.10	595.76	56.91	0.58

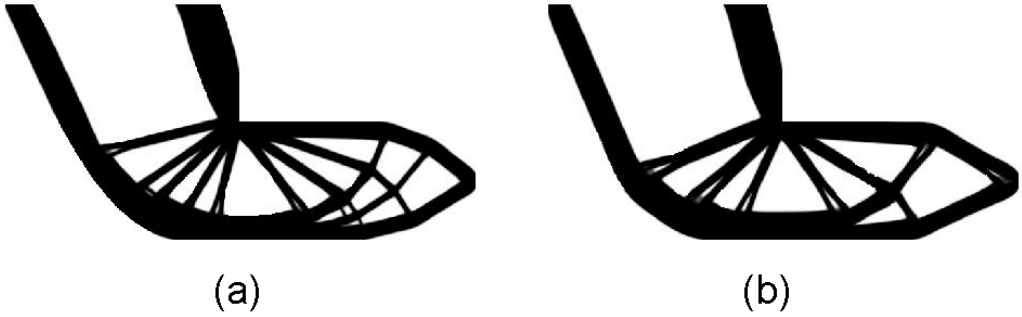

Figure 6. Optimized domain for test example B for the case of 200,000 FEs discretization: (**a**) SIMP output and (**b**) DL-SCALE output.

Subsequently, all recorded data for test example C are given in Table 3. Similar to the previous cases, DLMU achieved a maximum reduction of computational time equal to 78.69% for the case of 200,000 FEs in $D^{f}_{(0)}$ with respect to SIMP, while the objective function value slightly increased by 0.11%. The final optimized topologies generated by SIMP and DLMU for the finer discretization with 200,000 FEs are shown in Figure 7a,b, respectively.

Table 3. Test example C—Performance of the DLMU scheme.

ne	SIMP			DLMU			Acceleration (%)	Reduction (%)
	Iterations	Objective	Time (s)	Iterations	Objective	Time (s)		
75,000	131	109.84	225.78	61	109.73	196.48	12.98	0.10
100,000	438	111.53	1025.44	65	111.03	243.43	76.26	0.45
150,000	509	111.52	1888.79	74	21.62	321.29	61.44	0.09
200,000	556	112.51	2838.85	101	112.63	604.94	78.69	−0.11

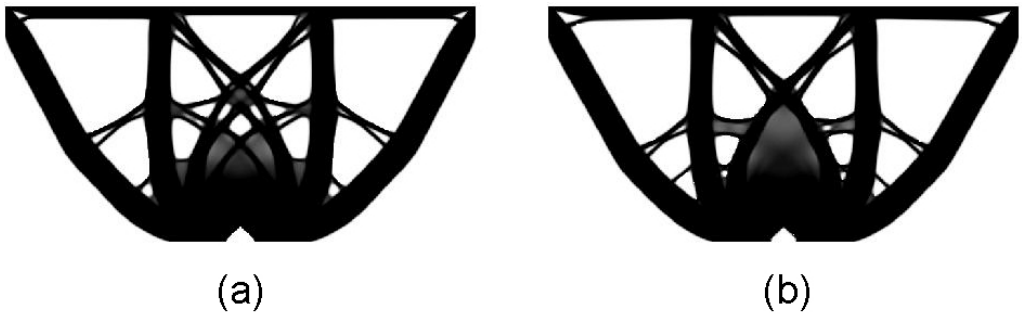

Figure 7. Optimized domain for test example C for the case of 200,000 FEs discretization: (**a**) SIMP output and (**b**) DL-SCALE output.

All recorded data for test example D are given in Table 4. Similar to the previous cases, DLMU achieved a maximum reduction of computational time equal to 60.58% for the case of 200,000 FEs in $D^{f}_{(0)}$ with respect to SIMP, while the objective function value is almost the same with that achieved by SIMP. The final optimized topologies generated by SIMP and DLMU for the finer discretization with 200,000 FEs are shown in Figure 8a,b, respectively.

Table 4. Test example D—Performance of the DLMU scheme.

ne	SIMP			DLMU			Acceleration (%)	Reduction (%)
	Iterations	Objective	Time (s)	Iterations	Objective	Time (s)		
75,000	261	21.30	330.54	74	21.22	204.30	38.19	0.38
100,000	326	21.45	559.69	79	21.38	252.59	54.87	0.33
150,000	310	21.60	833.30	74	21.62	321.29	61.44	−0.06
200,000	358	21.79	1306.80	99	21.79	515.14	60.58	−0.01

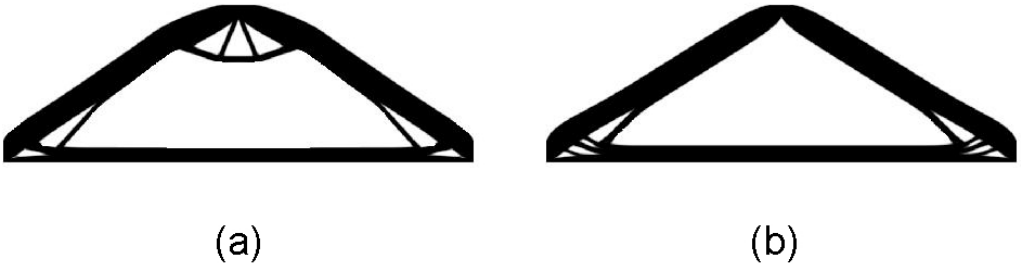

Figure 8. Optimized domain for test example D for the case of 200,000 FEs discretization: (**a**) SIMP output and (**b**) DL-SCALE output.

All recorded data for test example E are given in Table 5. Similar to the previous cases, DLMU achieved a maximum reduction of computational time equal to 67.48% for the case of 200,000 FEs in $D^f_{(0)}$ with respect to SIMP, while the objective function value slightly increased by 0.77%. The final optimized topologies generated by SIMP and DLMU for the finer discretization with 200,000 FEs are shown in Figure 9a,b, respectively.

Table 5. Test example E—Performance of the DLMU scheme.

ne	SIMP			DLMU			Acceleration (%)	Reduction (%)
	Iterations	Objective	Time (s)	Iterations	Objective	Time (s)		
75,000	314	23.71	462.43	80	23.72	246.44	46.71	−0.05
100,000	393	23.79	813.58	114	23.78	261.53	55.56	0.08
150,000	429	23.99	1396.71	183	23.87	718.71	48.54	0.49
200,000	573	24.16	2538.61	156	23.97	820.53	67.68	0.77

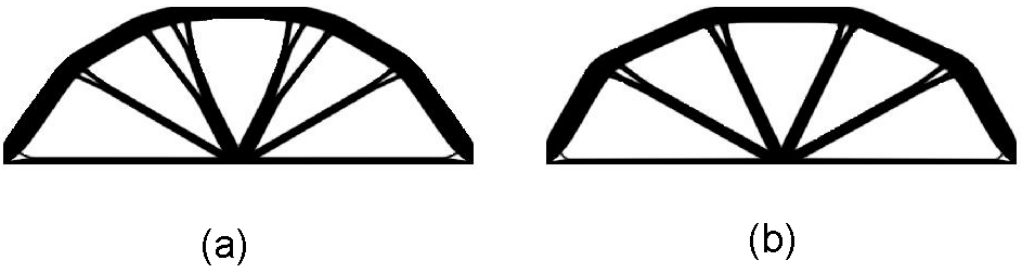

Figure 9. Optimized domain for test example E for the case of 200,000 FEs discretization: (**a**) SIMP output and (**b**) DL-SCALE output.

5.3. Comparison with the DL-SCALE and DLTOP Methodologies

Recently, a new methodology named DL-SCALE [28] has been proposed by the authors that also aims to reduce the computational demand of topology optimization with the use of deep learning under a reduced order modeling framework. In this section, the efficiency of the proposed DLMU scheme is assessed in terms of computing time and objective function value over the past ones proposed by the authors (i.e., the DLTOP [1] and DL-SCALE [28] methodologies) for the case of 200,000 number of FEs. The comparison

is presented in Table 6. The final optimized configuration achieved by any of the three methodologies (DLMU/DL-SCALE/ DLTOP) is more or less identical to the one obtained using SIMP, and thus a comparison of the results is made only in respect of iterations, objective function value and running time.

Regarding the implementation details, DLTOP requires performing 36 conventional steps of SIMP, then the already trained DBN is used to perform a close to final density distribution for the case of three classes classification (i.e., 0, 0.5 and 1), followed by the refinement steps. Regarding the DL-SCALE methodology, its implementation requires a set of n_{probs} auxiliary STO problems that rely on reduced order models compared to the fully refined TOP^f one. Based on the investigation performed in [28], the number of these problems is equal to $n_{probs} = 5$ and the number of FEs used to discretize these coarse meshes are equal to [3000, 4000, 5000, 7000, 10,000] elements, respectively. Evidently both DLMU and DL-SCALE outperform DLTOP in all test examples considered, both in terms of acceleration and reduction of the objective function value, all compared with the time required and the result obtained by the corresponding implementation of SIMP approach. Furthermore, DLMU slightly outperforms DL-SCALE in terms of the quality of the solution achieved as denoted by the reduction of the objective function value compared to the reference one obtained by SIMP.

Table 6. Comparison of the DLMU scheme with the DL-SCALE and DLTOP methodologies for the case of 200,000 number of FE.

	SIMP			**DLMU/DL-SCALE/DLTOP**			**Acceleration (%)**	**Reduction (%)**
	Iterations	**Objective**	**Time**	**Iterations**	**Objective**	**Time**		
Example A								
DLMU				86	244.39	495.55	80.64	−0.73
DL-SCALE	521	242.62	2559.05	107	246.82	670.52	73.80	−1.73
DLTOP				141	244.96	765.96	70.07	−0.97
Example B								
DLMU				107	155.10	595.76	56.91	0.58
DL-SCALE	367	156.01	1382.55	90	156.08	539.94	60.95	−0.04
DLTOP				115	161.65	639.29	53.76	−3.61
Example C								
DLMU				101	112.63	604.94	78.69	−0.11
DL-SCALE	556	112.51	2838.85	88	112.79	561.89	80.21	−0.25
DLTOP				121	110.56	696.58	75.46	1.73
Example D								
DLMU				99	21.79	515.14	60.58	−0.01
DL-SCALE	358	21.79	1306.8	86	21.76	480.23	63.25	0.12
DLTOP				165	21.82	848.38	35.08	−0.15
Example E								
DLMU				156	23.97	820.53	67.68	0.77
DLSCALE	573	24.16	2538.61	132	24.82	737.59	70.95	−2.73
DLTOP				220	26.35	1156.65	54.44	−9.07

6. Discussion

SIMP is considered as one the most widely used approaches for solving TO problems, that, like every structural optimization procedure, is associated with increased demand in computing time and resources. The DLTOP methodology previously developed by the authors of [1], motivated by the insufficiency of SIMP approach, employs deep learning approaches to reduce the number of SIMP iterations, thus achieving a substantial acceleration of the STO problem solution procedure. Specifically, a great amount of SIMP iterations is eliminated by employing a trained DBN, which uses the data generated during the first

few iterations to propose a close to final optimized configuration of the design domain, that is then fine-tuned with only a few additional iterations of SIMP. The efficiency of DLTOP stems from the fact that the DBN needs only be trained once and can be subsequently applied to any 2D or 3D STO problem, irrespective of its specifications. Within the scope of further enhancing the computational efficiency of DLTOP, the authors envisioned a novel approach that combines deep learning with reduced-order modeling, which led to the development of the DLMU scheme presented in this paper. Building upon the main principle of DLTOP, the novel DLMU scheme further employs deep learning to extrapolate the results of reduced order models to the full model of the design domain. DLMU thus inherits the benefits of DLTOP, in respect of training and calibrating the DBN only once, while also achieving further acceleration of SIMP via the employment of reduced order models, without compromising the solution quality.

The conclusions drawn regarding the DLMU scheme performance, based upon its application to a series of benchmark STO problems from the literature, as well as its comparison to SIMP, DLTOP and the novel DLSCALE approach also proposed by the authors [28], are summarized hereafter: (i) DLMU achieves a remarkable acceleration of SIMP, of the order of 40–80%; (ii) the acceleration of SIMP offered by DLMU becomes more pronounced with increasing mesh density, as a result of the enhanced reduced order modeling efficiency (acceleration increases by 15–65% between 75,000 and 200,000 FEs); (iii) DLMU does not compromise the solution quality offered by SIMP, irrespective of the employed mesh density (less than 1% reduction of objective function value for all test examples and mesh densities); (iv) DLMU, DLTOP and DL SCALE outperform SIMP, achieving similar acceleration values of the order of 35–80%, with negligible impact on the solution quality (less than 2% reduction of objective function value for all test examples and mesh densities); (v) in the majority of cases (test examples B, C, D, E), DLMU marginally underperforms DL SCALE; and (vi) in all cases DLMU and DL SCALE outperform DLTOP, which highlights the efficiency enhancement stemming from the employment of a reduced order model framework.

The future work is oriented towards the development of a web application where the two trained machine learning components of the proposed scheme will be provided for free. With regard to the topology optimization part, in order to replicate the results presented above, TOP88 needs to be used (freely available). With regard to the deep learning part of the work, the reader should contact the corresponding author (nlagaros@central.ntua.gr) for providing the trained DBM networks.

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. and A.N.N.; investigation, N.A.K.; writing—original draft preparation, N.A.K., A.N.N. and N.D.L.; writing—review and editing, N.A.K., A.N.N. and N.D.L.; visualization, N.A.K.; supervision, N.D.L.; project administration, N.D.L.; funding acquisition, N.D.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by European Union funds grant number 101007595.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: This research has been supported by the ADDOPTML project: "ADDitively Manufactured OPTimized Structures by means of Machine Learning" (No: 101007595) belonging to the Marie Skłodowska-Curie Actions (MSCA) Research and Innovation Staff Exchange (RISE) H2020-MSCA-RISE-2020.

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

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Networks
DBN	Deep Belief Network
DBM	Deep Boltzmann Machine
DLMU	Deep Learning-based Model Upgrading
DLTOP	Deep Learning-assisted Topology Optimization
FE	Finite Element
RBM	Restricted Boltzmann Machine
SIMP	Solid Isotropic Material with Penalization
STO	Structural Topology Optimization
TO	Topology Optimization
TOP	Topology Optimization Problem

References

1. Kallioras, N.A.; Kazakis, G.; Lagaros, N.D. Accelerated topology optimization by means of deep learning. *Struct. Multidiscip. Optim.* **2020**, *20*, 21–36. [CrossRef]
2. Lagaros, N.D.; Papadrakakis, M.; Kokossalakis, G. Structural optimization using evolutionary algorithms. *Comput. Struct.* **2002**, *80*, 571–589. doi: 10.1016/S0045-7949(02)00027-5. [CrossRef]
3. Lagaros, N.; Plevris, V.; Papadrakakis, M. Multi-objective design optimization using cascade evolutionary computations. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3496–3515. [CrossRef]
4. 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]
5. Lagaros, N.D.; Vasileiou, N.; Kazakis, G. A C# code for solving 3D topology optimization problems using SAP2000. *Optim. Eng.* **2018**, *20*, 1–35. [CrossRef]
6. Papadrakakis, M.; Lagaros, N.D. Soft computing methodologies for structural optimization. *Appl. Soft Comput.* **2003**, *3*, 283–300. [CrossRef]
7. 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]
8. Hinton, G.E.; Salakhutdinov, R.R. Reducing the Dimensionality of Data with Neural Networks. *Science* **2006**, *313*, 504–507. [CrossRef]
9. Hinton, G.E. Learning multiple layers of representation. *Trends Cogn. Sci.* **2007**, *11*, 428–434. [CrossRef] [PubMed]
10. Hinton, G.E.; Osindero, S.; Teh, Y.W. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef]
11. Hinton, G.E. Boltzmann machine. *Scholarpedia* **2007**, *2*, 1668. [CrossRef]
12. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 1 November 2021).
13. Hinton, G.E. A practical guide to training restricted Boltzmann machines. In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 599–619.
14. Rumelhart, D.E.; McClelland, J.L. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1. Foundations*; MIT Press: Cambridge, MA, USA, 1986.
15. Hestenes, M.R.; Stiefel, E. *Methods of Conjugate Gradients for Solving Linear Systems*; NBS: Washington, DC, USA, 1952; Volume 49.
16. Salakhutdinov, R.; Hinton, G. Deep Boltzmann Machines. In Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics, Clearwater, FL, USA, 16–18 April 2009; van Dyk, D., Welling, M., Eds.; Volume 5, pp. 448–455.
17. Sigmund, O.; Maute, K. Topology optimization approaches. *Struct. Multidiscip. Optim.* **2013**, *48*, 1031–1055. [CrossRef]
18. Wang, X.; Xu, S.; Zhou, S.; Xu, W.; Leary, M.; Choong, P.; Qian, M.; Brandt, M.; Xie, Y.M. Topological design and additive manufacturing of porous metals for bone scaffolds and orthopaedic implants: A review. *Biomaterials* **2016**, *83*, 127–141. [CrossRef] [PubMed]
19. Zhu, J.H.; Zhang, W.H.; Xia, L. Topology optimization in aircraft and aerospace structures design. *Arch. Comput. Methods Eng.* **2016**, *23*, 595–622. [CrossRef]
20. Dapogny, C.; Faure, A.; Michailidis, G.; Allaire, G.; Couvelas, A.; Estevez, R. Geometric constraints for shape and topology optimization in architectural design. *Comput. Mech.* **2017**, *59*, 933–965. [CrossRef]
21. Sigmund, O. Design of Material Structures Using Topology Optimization. Ph.D. Thesis, Technical University of Denmark, Kgs. Lyngby, Denmark, 1994.
22. Papoutsis-Kiachagias, E.M.; Giannakoglou, K.C. Continuous Adjoint Methods for Turbulent Flows, Applied to Shape and Topology Optimization: Industrial Applications. *Arch. Comput. Methods Eng.* **2016**, *23*, 255–299. [CrossRef]
23. Bendsøe, M.P. Optimal shape design as a material distribution problem. *Struct. Optim.* **1989**, *1*, 193–202. [CrossRef]

24. 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]
25. Mlejnek, H. Some aspects of the genesis of structures. *Struct. Optim.* **1992**, *5*, 64–69. [CrossRef]
26. Sigmund, O. A 99 line topology optimization code written in Matlab. *Struct. Multidiscip. Optim.* **2001**, *21*, 120–127. [CrossRef]
27. Andreassen, E.; Clausen, A.; Schevenels, M.; Lazarov, B.S.; Sigmund, O. Efficient topology optimization in MATLAB using 88 lines of code. *Struct. Multidiscip. Optim.* **2011**, *43*, 1–16. [CrossRef]
28. Kallioras, N.A.; Lagaros, N.D. DL-SCALE: A novel deep learning-based model order upscaling scheme for solving topology optimization problems. *Neural Comput. Appl.* **2021**, *33*, 7125–7144. [CrossRef]