POLITECNICO
MILANO 1863

DIPARTIMENTO DI
INGEGNERIA CIVILE E AMBIENTALE

Executive Summary of the Thesis

DYNUTOP: Dynamically Updating Topology Optimization Prediction using Artificial Intelligence

Laurea Magistrale in Civil Engineering - Ingegneria Civile

Author: Gabriel Garayalde

Advisors: Prof. Alberto Corigliano & Prof. Matteo Bruggi & Matteo Torzoni

Academic year: 2022-2023

1. Introduction

Topology Optimization (TO) is a powerful tool in computational design that optimizes the distribution of material within a specified domain to create structures that adhere to prescribed design constraints. The resulting topology provides an efficient use of material, leading to cost savings and lightweight structures, which is valuable for engineers. However, computing time remains a bottleneck, limiting its application in real engineering contexts. To address this challenge, researchers are exploring the use of artificial intelligence to accelerate the process. This thesis proposes a multi-stage machine learning model that aims to predict an optimal topology in 2D or 3D, in a single shot without an initial form-finding process. The proposed method utilizes a combination of machine learning models to solve distinct parts of the TO problem, resulting in a near-instantaneous optimization. The thesis explores three problem cases, and the results are presented in an interactive computer application that visualizes and updates the predicted topology in real-time in response to user changes of the loading parameters. The proposed method can generate mean average error accuracy of less than 0.035 for the density field for all three test cases, with a predictive speed that is less than 0.5% that of a traditional TO algorithm.

2. Topology Optimization formulation

This chapter presents the general formulation for the topology optimization problem, and the solution algorithm based on the SIMP method. This will be adopted for dataset generation purposes, as detailed in the next section. In particular, we adopt the 88-line MATLAB code 'top88.m' developed by Andreassen et al. (2010) [1] for 2D problems, whilst the extension to 3D is based on the 125-line code proposed in the paper by Ferrari and Sigmund (2020) [2]. The focus of the following formulation is restricted to minimum compliance problems with a statically applied load.

2.1. 2D TO formulation

By adopting the SIMP approach, the user is first required to input the number of rectangular finite element along the x and y directions, in which the design domain should be discretized. Each finite element e is characterized by a density value x_e that affect the corresponding value

of the Young's Modulus E_e, as follows:

$$E_e(x_e) = E_{min} + x_e^p(E_0 - E_{min}), \quad x_e \in [0,1]. \tag{1}$$

The first step is to specify the domain, boundary conditions and loading conditions of the problem. Subsequently parameters such as the volume fraction, filter radius and sensitivity filtering can be adjusted to produce an output as shown. The aim of the optimization problem is to find the optimal material distribution within a given domain that satisfies the minimum compliance problem for a fixed constraint on the amount of material, expressed as a fraction of the total design volume.
The mathematical formulation of the TO problem is given in the following set of equations:

$$\begin{aligned} \min_{\boldsymbol{x}}: \quad & \boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{U}^T\boldsymbol{K}\boldsymbol{U} = \sum_{e=1}^{N} E_e(x_e)\boldsymbol{u}_e^T\boldsymbol{k}_0\boldsymbol{u}_e, \\ \text{subject to}: \quad & V(\boldsymbol{x})/V_0 = f, \\ & \boldsymbol{K}\boldsymbol{U} = \boldsymbol{F}, \\ & \boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{1}. \end{aligned} \tag{2}$$

The first line describes the objective function, the so-called structural compliance. It is defined as the work of external loads at equilibrium, providing a measure of the overall stiffness. The minimization problem is subject to a set of constraints on the optimization variables, respectively in terms of prescribed volume fraction, global force-displacement relationship, and admissibility of the element densities. Herein: adopting a regular mesh of finite elements, $\mathbf{x}$ is the vector of design variables (one per element); c is the compliance to be minimized; $\mathbf{U}$ and $\mathbf{F}$ are the global displacement and force vectors, respectively; $\mathbf{K}$ is the global stiffness matrix, assuming a linear elastic isotropic material; $\mathbf{u}_e$ is the element displacement vector; and $\mathbf{k}_0$ is the element stiffness matrix for an element of unit Young's Modulus; N refers to the number of elements within the discretized domain; f refers to the prescribed volume fraction; $V(\mathbf{x})$ and V_0 refer to the volume of the placed material and to the target design volume, respectively. The assumption of small displacement holds.
To update the element density values over each iteration of the optimization problem, we adopt the following standard optimality criteria method with heuristic updating strategy:

$$x_e^{new} = \begin{cases} \max(0, x_e - m) & \text{if } x_e B_e^\eta \le \max(0, x_e - m) \\ \min(1, x_e + m) & \text{if } x_e B_e^\eta \ge \min(1, x_e + m) \\ x_e B_e^\eta & \text{otherwise} \end{cases}, \tag{3}$$

where m is a suitable threshold on the iteration step, η (=1/2) is a numerical damping coefficient, B_e is derived from the optimality condition, as below:

$$\mathbf{B}_e = \frac{-\frac{\partial c}{\partial x_e}}{\lambda \frac{\partial V}{\partial x_e}}, \tag{4}$$

with λ being a suitable Lagrangian multiplier value, selected using a bisection algorithm whilst still satisfying the volume constraint. Eq.(4) involves the sensitivities of the objective function c and the material volume V with respect to the element densities x_e, computed under the assumption that each element has unit volume. An additional feature is the filtering strategy; that is, the application of sensitivity filter which involves a weighted average over different elements. This is a partially heuristic, yet efficient solution to the arising of undesired checkerboard patterns (unphysical minima) and mesh dependence issues.

2.2. Extension to 3D TO formulation

The extension to 3D problems is instead based on the '125-line' code, known as 'top3D125', and released in 2020 by Ferrari and Sigmund [2]. This code employs significantly more efficient filter assembly and implementation procedures compared to the 'top88' code, as well as shortcuts in the design update step. One of the other main differences employed in the 3D formulation, as opposed to the 2D formulation, is the filtering of the densities (in the 2D formulation the filtering is applied only to the sensitivities). This in turn necessitates a minor reformulation to the sensitivity equations. The sensitivities of the objective function c and volume V with respect to the design variables x_j are obtained through use of the chain rule. The only other minor modifications from the 2D formulation is the definition of the elemental stiffness matrix K_e^s for the 8-node hexahedron, and the addition of the extra 'space-dimension' in the necessary lines to account account for 3D.

3. Matlab Implementation and Dataset formulation

For both the 2D and 3D cases, it is necessary to create a robust and large enough dataset to properly train the machine learning model. We

define *cases* here to mean unique sets of boundary conditions, which can be implemented by means of a suitable parametrization of the MATLAB code. In particular, we consider case studies involving an MBB beam as 2D example and a cantilever and a bridge as extension to 3D. In this executive summary, we include only the 3D bridge case for conciseness. To properly sample the parametric input space (which accounts for the loading conditions, and also the boundary conditions in the bridge case), a Latin hypercube sampling (LHS) strategy was employed. LHS is a statistical method for generating a near-random sample of parameter values from a multidimensional distribution and allows a uniform sampling of the full sample space.

3.1. 2D MBB

The MBB case was discretized in a domain with size `nelx = 120` and `nely = 40`. The volume fraction was prescribed as `volfrac = 0.5`, the penalization power `penal = 3` and the filter radius `r_min = 1.5`. The code is split into a main file which uses a `for` loop to call the TO algorithm for each unique set of loading parameters. Each topology optimization loop produces three outputs; a grey scale optimal topology, a VM Stress map, and a TorC field. For the 2D MBB case two loading parameters that are allowed to vary are the `angle` $= \theta$, which controls the angle of the force vector, measured anti-clockwise from the positive x-axis, and the `nodeID` which controls the node number onto which the force vector is applied. As a constraint, the force vector may be applied only on the nodes on the free surface boundary, shown in Fig. 1.

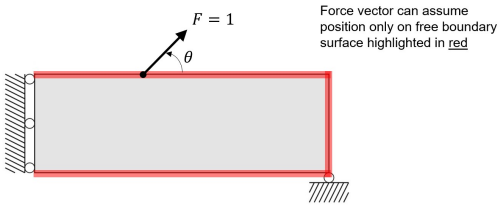

Figure 1: Allowable force vector positions in the MBB case.

The VM stresses were calculated using the following equation:

$$\sigma_{VM,e} = \sqrt{(\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\tau_{xy}^2)}. \quad (5)$$

The result is shown in Fig. 2, whereby the optimal topology is element wise multiplied with the initial VM stress field to produce the final VM stress field,

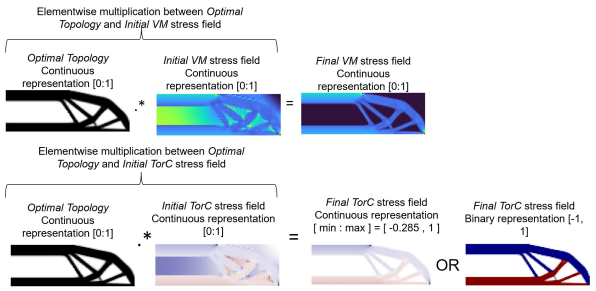

Figure 2: Optimal topology element wise multiplied with the initial stress field to produce the final stress field for both VM and TorC

The TorC zones utilise a similar concept of passing a stress field through the optimal topology 'masking' layer. However, in this case, the considered stress field is characterized by values normalized to range from -1 to 1, with negative and positive values representing elements mainly in tension and compression, respectively. This kind of representation is assumed to incorporate important information about the dominant principal stress acting on each element during the optimization procedure.

3.2. 3D Bridge

The 3D Bridge case was discretized in a domain with size `nelx = 60` and `nely = 20` and `nelz = 4`. The volume fraction was prescribed as `volfrac = 0.12`, the penalization power `penal = 3` and the filter radius `r_min` $= \sqrt{3}$. The deck is constrained as full material (density $= 1$) and has a unit thickness. The void region located below the deck has a width of 30 units. The deck region, the void region, and the region in which the topology is free to develop are highlighted in Fig. 3 in red, orange and grey, respectively. The parametric input space accounts for 4 parameters adopted to produce unique topologies: two control the x-axis position of the acting loads and the other two control the x-axis position of the supports (see Fig. 3).

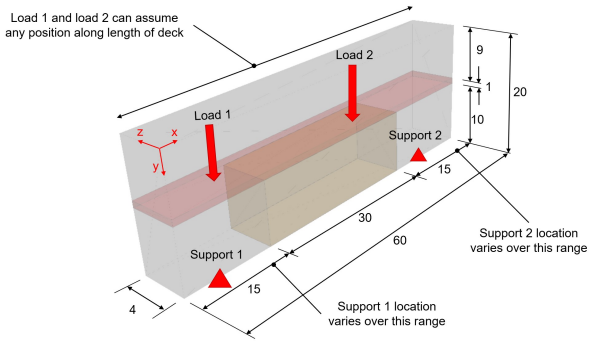

Figure 3: 3D Bridge case: schematic representation of the 4 parameters adopted to create unique topologies.

A procedure analogous to that adopted for the 2D case is used to map the VM stress field onto the optimal topology also in the 3D case. This is computed at the element level as:

$$\sigma_{VM,e} = \sqrt{\frac{1}{2}((\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{xx} - \sigma_{zz})^2 + (\sigma_{yy} - \sigma_{zz})^2 + 6 \cdot (\tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2))}. \tag{6}$$

Fig. 4 shows a randomly chosen optimal topology, as well as the initial VM or TorC stress field, and the resulting elementwise product; the final VM or TorC stress field.

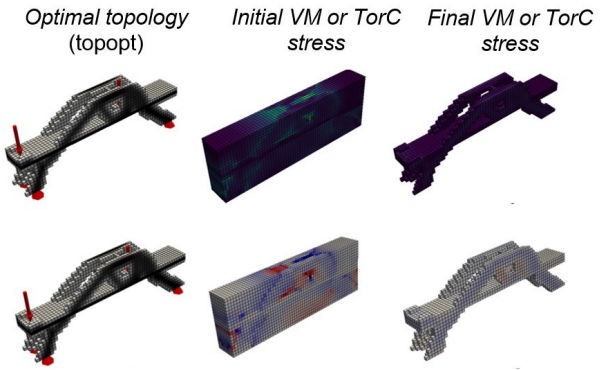

Figure 4: Generated optimal topology, initial VM or TorC stress field and final VM or TorC stress field.

4. Machine learning formulation

The proposed DL framework is schematized in Fig. 5. The first two steps of the process consist of training the two DL models, whilst the final testing step combines the trained models to predict the optimal topology for a given set of loading parameters. In particular, the three steps involve: first, an AE is trained to map optimal topologies into itself; second, a MLP model is trained to map the input loading parameters onto the latent space representation of the corresponding optimal topology, as provided by the encoding branch of the trained AE; finally, the MLP model trained in step 2 and the decoding branch of the AE trained in step 1 are synergistically exploited to create a DL pipeline capable of predicting, almost in real-time, the correct topology for given a set of loading parameters. These steps will be explained in more detail in the following sections.

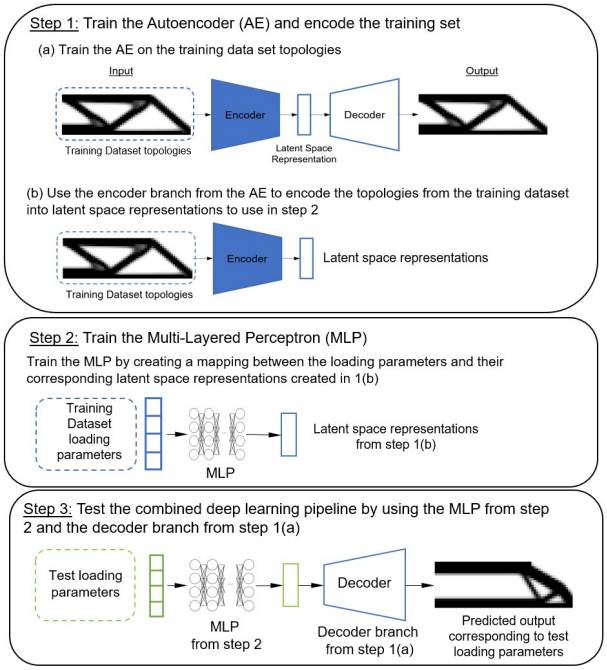

Figure 5: Scheme of the proposed deep learning pipeline.

5. Results

To evaluate the similarity between the predicted structures and the optimal topologies produced using the SIMP algorithm, three accuracy metrics are used: the binary accuracy (BA) [3], the MAE and the root mean squared accuracy (RMS). The BA metric is given by:

$$BA = \frac{\omega_{00} + \omega_{11}}{n_0 + n_1}. \tag{7}$$

Herein:, n_l with $l \in \{0, 1\}$, is the total number of pixels of class l, and ω_{tp}, with $t \in \{0, 1\}$ and $p \in \{0, 1\}$, is the total number of pixels of class t predicted to belong to class p. Because the predicted structures contained pixel values within a continuum range (between 0 and 1 for optimal topologies and VM stress, and between -1

and 1 for TorC zones), a threshold function was utilised to binarize the predicted outputs.

$$x_{i,binary} = \begin{cases} 1, & \text{if } x_i > x_{threshold} \\ 0, & \text{if } x_i < x_{threshold} \end{cases} \quad \text{for } i = 1, ..., n. \tag{8}$$

Herein: x_i is the input topology, $x_{i,binary}$ is the binarized output field, and $x_{threshold}$ is the binary threshold value: 0.5 for optimal topologies and VM stress, and 0 for TorC zones. The MAE and RMS accuracy metrics are given as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |x_i - \widehat{x_i}| \quad \text{for } i = 1, ..., n. \tag{9}$$

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - \widehat{x_i})^2} \quad \text{for } i = 1, ..., n. \tag{10}$$

Where x_i refers to the density value assumed by element i, $\widehat{x_i}$ refers to the predicted density value assumed by element i, and n is the total number of elements.

5.1. 2D MBB

The first quantitative evaluation of the benefit of the proposed method is the comparison between the time taken to generate the optimal topology using the SIMP algorithm, and the time taken to predict the optimal topology using the trained neural network. This is shown below in table. 1.

Table 1: Comparison of average computational run-time between SIMP algorithm and proposed methodology.

	SIMP	Proposed Method: Optimal Topology	Proposed Method: Optimal topology + VM or TorC
Dataset creation	-	8.10 hours	8.10 hours
Training DL model	-	0.44 hours	0.88 hours
Average run-time (in seconds)	12.00 s	0.08 s	0.16 s

A comparison for the average accuracy metrics on the test set for the 2D MBB is shown below: An accuracy comparison between the ground truth and predicted topology is shown for the 3D bridge case in Fig. 6 below.

Table 2: Comparison of average accuracy metrics for the predicted topologies when compared to the ground truth.

	Optimal Topologies	Initial VM stress field	Initial TorC stress field
BA	96.46%	99.29%	95.26%
MAE	0.035	0.018	0.025
RMS	0.107	0.030	0.048

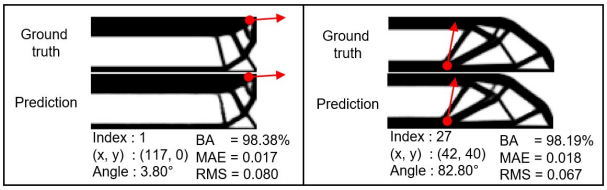

Figure 6: Accuracy comparison of the ground truth and prediction for two optimal topologies for the 2D MBB case.

5.2. 3D Bridge

A similar comparison between the average computational run-times between the SIMP algorithm and the proposed methodology is shown below.

Table 3: Comparison of average computational run-time between SIMP algorithm and proposed methodology.

	SIMP	Proposed Method: Optimal Topology	Proposed Method: Optimal topology + VM or TorC
Dataset creation	-	25.91 hours	25.91 hours
Training DL model	-	0.61 hours	1.22 hours
Average run-time (in seconds)	37.31 s	0.13 s	0.26 s

A comparison for the average accuracy metrics on the test set for the 3D bridge is shown below:

Table 4: Comparison of average accuracy metrics for the predicted topologies when compared to the ground truth.

	Optimal Topologies	Initial VM stress field	Initial TorC stress field
BA	98.50%	98.27%	88.36%
MAE	0.015	0.037	0.047
RMS	0.051	0.057	0.078

An accuracy comparison between the ground

truth and predicted topology is shown for the 3D bridge case in Fig. 7 below.

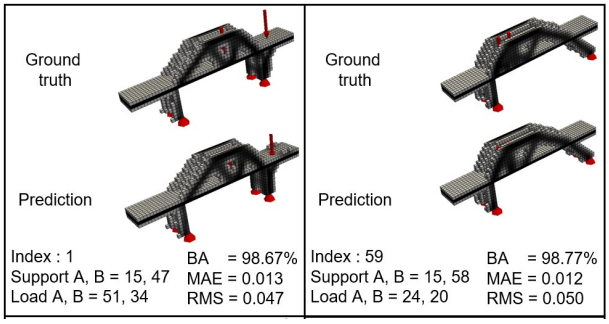

Figure 7: Accuracy comparison of the ground truth and prediction for six optimal topologies for the 3D bridge case.

5.3. App: 3D Bridge

Although the traditional SIMP algorithm can provide very optimized solutions for structural problems, the long computation times, especially for increasingly larger domains, has prevented it from developing into a serious and widespread design tool for structural engineers. Design tools that can provide solutions with shorter computation times, and allow an engineer to iterate over many potential solutions to get an understanding of how the problem changes with respect to variations in the loading and boundary conditions can save time and increase efficiency. In the following section an app is presented, whereby the loading parameters for each 3D case (explained in detail in chapter 3 is parametrized and controlled by the user. This app has been created using the Python package *PyVista* [4]. PyVista is a Python library for 3D visualization and analysis of scientific datasets such as meshes, point clouds, and volumetric data and provides a user-friendly interface to generate 3D plots and interactive visualizations with advanced rendering capabilities. In this app, the predicted topology changes in response to the chosen loading parameters, and updates almost instantaneously as the loading parameters are updated. The loading parameters are controlled using a sliding bar with a predefined range. The topology is displayed within the window of the user interface as a 3D object. Fig. 8 shows three screenshots of the app with randomly chosen loading parameters and predicted topologies.

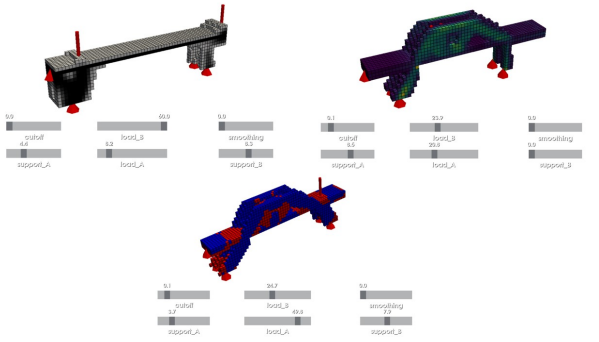

Figure 8: PyVista render of three randomly predicted topologies for (a) optimal topology (b) final VM stress and (c) final TorC stress.

6. Conclusions

This thesis proposes an efficient non-iterative multistage DL model that is capable of predicting a near optimal solution for the topology optimization problem in 2D and 3D. The model consists of two parts; first, an MLP model that takes as inputs the loading parameters for the specific problem case and outputs the corresponding latent space representation. Second, the decoder branch of an autoencoder that takes as input this previously generated latent space representation and predicts either the optimal topology, the VM stress field, or the TorC field. Together, these two parts combine to produce the final DL pipeline that is able to predict accurate solutions almost instantaneously. The final VM or TorC results, are an element-wise multiplication between the predicted optimal topology and the either the predicted VM stress field or predicted TorC field respectively. The predicted optimal topology thus acts as a masking layer, and the predicted VM stress field or predicted TorC field will be mapped onto those elements with a positive density value, whilst the elements with a null value will remain void spaces.

Comparing the total average scores for the predicted *optimal topologies* when compared to the ground truth, the 3D bridge scored the highest (98.50%, 0.015, 0.051), and then the 2D MBB (96.46%, 0.035, 0.107), for the three accuracy metrics (binary accuracy, mean absolute error, root mean square). Comparing the total average scores for the predicted *initial VM stress field* when compared to the ground truth, the 2D MBB scored the highest (99.29%, 0.018, 0.030), followed by 3D bridge (98.27%, 96.32%,

94.28%). Comparing the total average scores for the predicted *initial TorC stress field* when compared to the ground truth, the 2D MBB scored the highest (95.26%, 0.025, 0.048), followed by 3D bridge (88.36%, 0.047, 0.078).
A total training set of 2,500 images was generated for all three problem cases. The training time varied depending on the size of the domain for each case; for the 2D MBB, and 3D bridge, the training time was 8.1 hours, and 25,91 hours, respectively. Additionally, time was required to train the DL models, ranging from 0.44 hours to 0.66 hours for the 2D and 3D cases, respectively. The one key feature of the proposed method is that it requires a large up front time investment to create the training dataset and train the DL model. This training was all done 'offline', that is, completed before the model was deployed. Once the training was completed however, the prediction times are near-instantaneous, and achieve near-optimal results when compared to the traditional SIMP topology optimization algorithms.

References

[1] Erik Andreassen, Anders Clausen, Mattias Schevenels, Boyan S. Lazarov, and Ole Sigmund. Efficient topology optimization in MATLAB using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16, nov 2010.

[2] Federico Ferrari and Ole Sigmund. A new generation 99 line matlab code for compliance topology optimization and its extension to 3d. 62(4):2211–2228, May 2020.

[3] Ivan Sosnovik and Ivan Oseledets. Neural networks for topology optimization. September 2017.

[4] Bane Sullivan and Alexander Kaszynski. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, May 2019.