Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 398 (2022) 115116
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Machine learning for topology optimization: Physics-based learning through an independent training strategy

Fernando V. Senhora[a], Heng Chi[b], Yuyu Zhang[c], Lucia Mirabella[b], Tsz Ling Elaine Tang[b], Glaucio H. Paulino[a,*]

[a] School of Civil and Environmental Engineering, Georgia Institute of Technology, 790 Atlantic Drive, 30332, Atlanta, GA, USA
[b] Siemens Corporation, Technology, 755 College Rd E, 08540, Princeton, NJ, USA
[c] School of Computational Science and Engineering, Georgia Institute of Technology, 266 Ferst Drive, 30332, Atlanta, GA, USA

Received 14 June 2021; received in revised form 27 April 2022; accepted 9 May 2022
Available online 4 July 2022

Abstract

The high computational cost of topology optimization has prevented its widespread use as a generative design tool. To reduce this computational cost, we propose an artificial intelligence approach to drastically accelerate topology optimization without sacrificing its accuracy. The resulting AI-driven topology optimization can fully capture the underlying physics of the problem. As a result, the machine learning model, which consists of a convolutional neural network with residual links, is able to generalize what it learned from the training set to solve a wide variety of problems with different geometries, boundary conditions, mesh sizes, volume fractions and filter radius. We train the machine learning model separately from the topology optimization, which allows us to achieve a considerable speedup (up to 30 times faster than traditional topology optimization). Through several design examples, we demonstrate that the proposed AI-driven topology optimization framework is effective, scalable and efficient. The speedup enabled by the framework makes topology optimization a more attractive tool for engineers in search of lighter and stronger structures, with the potential to revolutionize the engineering design process. Although this work focuses on compliance minimization problems, the proposed framework can be generalized to other objective functions, constraints and physics.

Keywords: Machine Learning; Topology optimization; Large-scale; 3D convolutional neural network; Physics-based machine learning; Training Set

1. Introduction

Topology optimization is generally associated with a high computational cost because it is necessary to solve the state equations, related to the physics of the problem (e.g. linear elasticity, hyper-elasticity, fluid dynamics, heat conduction), at each iteration of the optimization. To reduce this computational cost, we propose a physics-based machine learning (ML)-driven topology optimization framework to drastically speed up topology optimization without accuracy loss. The ML model eliminates the need to solve the state equations at the fine-resolution

* Corresponding author.
E-mail address: gpaulino@princeton.edu (G.H. Paulino).

https://doi.org/10.1016/j.cma.2022.115116
0045-7825/

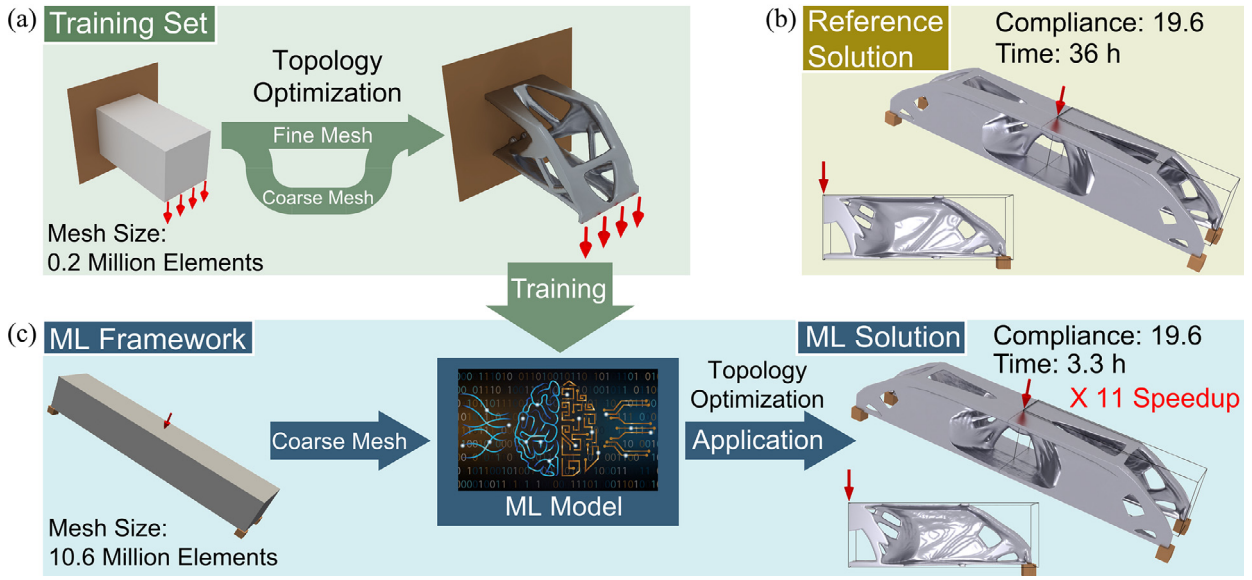

Fig. 1. A summary of the proposed framework: (a) we solve a topology optimization problem and use the data to train the ML model; (c) we then use the trained model to solve problems, which can be completely different from the problems used for training (e.g., different boundary conditions and different mesh size). (b) A comparison with the reference solution obtained by the standard topology optimization approach, i.e. no ML.

discretization (i.e., the discretization of the density variables which parametrize the design). To achieve this, we solve the state equations at a much coarser discretization and feed the solved state variables to a ML model that predicts the sensitivity information on the fine-resolution discretization. To train the ML model, we make use of the topology optimization solutions obtained from a few training problems with distinct design setups, and relatively small mesh size (which thus can be solved at a low computational cost). When collecting the training data, we compute the state variables both in a coarse mesh and in a fine mesh of the selected training problems, so that our ML model can learn the generic mapping between the coarse mesh state variables and the fine mesh sensitivity information, as illustrated in Fig. 1(a). Once the ML model is trained, it can then be applied to solve topology optimization problems with completely different design setups and parameters, including geometries, boundary conditions, mesh sizes, volume constraints, and filter radius, as exemplified in Fig. 1(b). This generalization capability of the ML model to problems that are not encountered during training is made possible by its careful incorporation of the underlying physics of the problem. As demonstrated by the comparison between Fig. 1(b) and (c), we are able to generate optimized designs whose compliance values are almost identical to the standard topology optimization procedure in only a fraction of the computational time.

The efficiency of the proposed ML framework comes from the separation of its training from its application in the optimization. Therefore, we only need to perform the training once, and the same trained model can be used to solve a wide variety of problems. This is the main difference between this work and the prior work [1], which adopts an online training approach. Fig. 2 provides a flowchart of the proposed framework that shows this separation of the training and the application of the ML model. Furthermore, given limited hardware and computational power (e.g. limitations of RAM memory, and processing time), the size of the state equations is the main factor preventing us from solving larger topology optimization problems. Because the proposed framework does not need to solve the state equations in the fine mesh throughout its procedure, it allows us to solve considerably larger problems when given limited hardware and computational power.

The remainder of this paper is organized as follows. Section 2 presents a literature review of existing ML approaches in topology optimization problems, and highlights the difference between our framework, and the works in literature. Section 3 presents the topology optimization formulation used in this work, along with details about the two-resolution discretization (i.e. coarse and fine meshes). Section 4 introduces the proposed ML framework, including the neural network architecture and the independent training strategy. Expanding on the idea

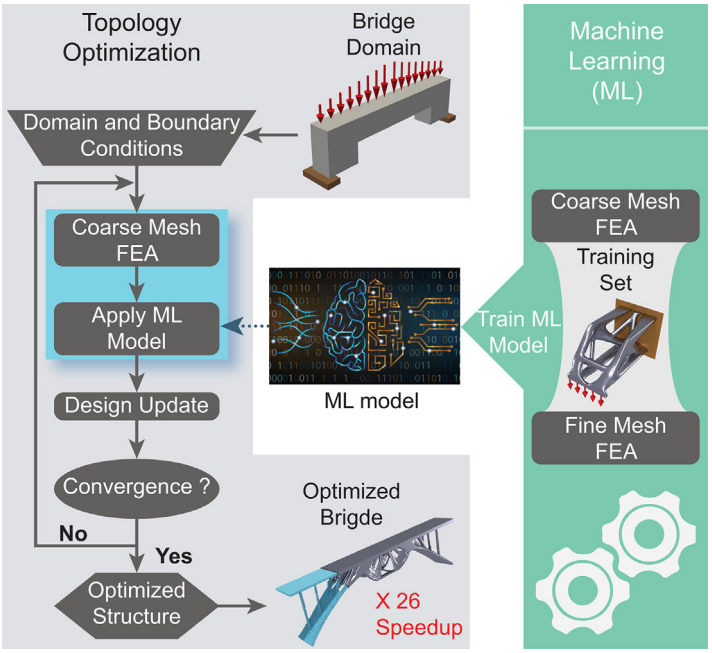

Fig. 2. A flowchart of the proposed framework, which separates the training of the machine learning model from its application to actual topology optimization problems.

of independent training, Section 5 explores the influence of the training set on the performance of the ML model. Section 6 presents numerical results that exemplify the capabilities of the framework, and Section 7 concludes the paper and summarizes the main contributions of this work.

2. Related work

Structural optimization [2] has the potential to revolutionize the way engineers design our world, leading to structures that are lighter, safer and more efficient than the present-day ones. In the field of structural optimization, topology optimization [3] is a popular technique that allows for the free distribution of material in a given domain to minimize a given objective function. Topology optimization has experienced tremendous development in recent years, driven by several educational codes made available in literature [4–6]. Nevertheless, the high computational cost associated with topology optimization remains a challenge that has prevented its widespread use, particularly in large-scale design optimization problems. To address this high computational cost, researchers have resorted to parallel computing [7–12], advanced iterative solvers [13,14], and multi-scale approaches [15–18]. Among them, parallel computing approaches require expensive hardware, such as supercomputers with hundreds of nodes, which are not readily available to the general public at a reasonable cost. Advanced iterative methods require domain knowledge to implement, and the gain in efficiency is limited. Multi-scale approaches generally lack the precision to produce fine detailed structures with accurate physical models. The above-mentioned limitations of the currently available techniques urge the need for new approaches that address the computational bottleneck of topology optimization. To achieve this goal, we propose a machine learning enhanced topology optimization framework that does not require expensive hardware, yet provides considerable speed up with a simple implementation and generates optimized results containing fine details.

Nowadays, ML has found numerous applications in various research fields, including image processing [19], natural language processing [20], finance [21], robotics [22], and mechanics [23], because of its powerful capability in capturing complex patterns in large data sets. In the context of ML applied to topology optimization, most of the literature focuses on the direct prediction of the optimized structures through ML techniques. For example, [24] tries to predict the final optimized topology when given a set of load and boundary conditions. However, the generalizability of their strategy is not fully demonstrated as their ML model is only verified for those design

domains and loading conditions that are close to the ones used to generate the training set. Refs. [25–27] use Convolutional Neural Networks (CNNs) to obtain the final solutions based on partially-converged topologies. However, the results obtained in those papers do not demonstrate substantial advantages over a simple thresholding technique but still require the partial solution of the topology optimization problem, which limits their potential speedup. The work [28] adopts a multi-resolution strategy in which the topology optimization is solved on a coarse mesh and a neural network is trained to project the result to a finer discretization, a process that is similar to a post-processing procedure. Although the approach produces smoother designs, it does not generate fine structural features in its prediction. The work [29] uses a two-resolution framework to predict the optimized structure directly. The authors first use an encoder–decoder CNN model to predict the optimal structure on a coarse-resolution mesh, and then adopt a conditional Generative Adversarial Network (cGAN) to upscale the coarse-resolution solution to a fine-resolution one. Although being successful in some problems, the approach is reported to generate sub-optimal designs with apparent local artifacts, such as hanging and floating members, in several design scenarios. More recently, [30] are able to increase the accuracy of the encoder–decoder architecture by using a Generative Adversarial Network (GAN). In another related work, [31] uses a similar strategy to design conductive heat transfer structures. The authors use GAN for both the coarse-resolution and the fine-resolution meshes and obtain relatively good solutions for one specific geometry. The work [32] uses a deep CNN with a U-net-like architecture to generate designs based on the mechanical information of the domain considering a homogeneous material distribution and demonstrates the performance of the framework for a fixed domain and discretization. In a similar context, [33] uses a CNN with a Wasserstein Generative Adversarial Networks (WGAN) architecture to generate 3D structures. However, the results in [33] exhibit a considerable amount of noise and need heavy post-processing to make the structure apparent. Similarly, [34] presents an approach using a U-net architecture to optimize structures subjected to a non-linear elastic model for 2D coarse meshes. After all, the above-mentioned approaches [29–34] all require a training dataset containing the solutions to a large number of topology optimization problems. The process of collecting this dataset is computationally expensive. Moreover, the performance of the ML models is heavily influenced by the composition of the training dataset, e.g., how many different design domains and loading conditions the dataset includes. Yet, this influence is rarely investigated and quantified in the above-mentioned works.

In a separate direction, [1] proposes a machine learning approach for topology optimization which does not require a pre-collected dataset. To achieve that, the work adopts an online training and update strategy where the history data (i.e., design variables and their corresponding sensitivities) of topology optimization are used as the training data. To ensure scalability of the approach, [1] introduces a two-resolution setup consisting of a coarse-resolution mesh and a fine-resolution mesh. The structural response (i.e, displacement and strain fields) computed on the coarse-resolution mesh is fed together with the stiffness variable on the fine-resolution mesh as inputs to a ML model to predict the sensitivity on the fine-resolution mesh. Although the approach proposed by [1] successfully demonstrates speedup up to $\sim$10 times for design problems with roughly one million design variables, the online training and update strategy adopted by [1] limits it from achieving higher speedup in larger problems. The limitations on the attainable speedup mainly come from: (1) The training of the ML model needs to be done during topology optimization, which could be computationally expensive for very large-scale problems. (2) The solution of the state equation on the fine-resolution mesh is still needed at the optimization steps where the training data are collected, which restricts the size of problems that can be handled when given limited hardware resources.

In this work, we present a novel ML-driven topology optimization framework that is free of the above-mentioned limitations. Our approach adopts a two-resolution setup similar to [1], in which the coarse-resolution structural response is fed together with fine-resolution stiffness as inputs to a ML model to predict the sensitivity on the fine-resolution mesh. Yet, unlike [1], our proposed framework adopts an offline strategy that separates the training of the ML model from its application in the optimization procedure. Combined, the two-resolution setup and offline training strategy allow the proposed framework to handle significantly larger design problems (up to $\sim$38 million elements mesh as compared to $\sim$1 million elements mesh in [1]) and to achieve a considerably higher speedup (up to $\sim 27\times$ compared to $\sim 10\times$ in [1]). Furthermore, the adopted offline training strategy allows us to use more advanced ML architectures with enhanced model capacity without impacting the speedup performance. Thus, this work adopts the state-of-the-art CNN architecture together with batch normalization, residual network connections, and normalized inputs to improve the accuracy of the prediction. In addition, the separation of training and prediction allows for the validation of the ML model before being applied to actual topology optimization problems. For

example, we validate the ML model using a portion of the data used in training to make sure that the ML model actually learns the underlying mapping between the input and output data (see Appendix B, Fig. B.17). This effectively eliminates the need for online updates. With several large-scale design examples, we demonstrate that the proposed framework is robust, scalable and efficient, and has a remarkable generalization capability across different design problems with distinct geometry, optimization parameters (filter radius, volume fraction), and number of load cases.

3. Topology optimization formulation

The topology optimization formulation used throughout this paper is based on the formulation presented in [6]. The formulation aims to minimize the compliance of the structure with a volume constraint (i.e. maximizing the stiffness of the structure considering a specified volume of material). In the optimization problem, we use the density-based Solid Isotropic Material with Penalization (SIMP) model [35,36] with Heaviside projection [37] to represent the material distribution. We apply a density filter to regularize the problem and to impose a minimum length scale. In this problem, we consider linear elastic materials.

The mathematical statement of the optimization problem is presented as follows

$$\begin{aligned} \min_{\mathbf{z}} \quad & C(\mathbf{z}) = \mathbf{f}^T\mathbf{u} \\ \text{s.t.} \quad & g_V(\overline{\mathbf{z}}) = \mathbf{v}^T\overline{\mathbf{z}} - V_{max} \leq 0 \\ & 0 \leq z_i \leq 1, \quad \forall\, i \in \{1,\ \ldots,\ M\} \\ \text{with:} \quad & \mathbf{K}(\overline{\mathbf{z}})\mathbf{u} = \mathbf{f} \\ & \overline{\mathbf{z}} = \mathcal{H}\,(\mathbf{P}\mathbf{z}) \end{aligned} \tag{1}$$

where $\mathbf{z}$ are the design variables, $C(\mathbf{z})$ is the compliance of the structure, $\mathbf{f}$ is the external load vector, $\mathbf{u}$ is the displacement vector, $g_V(\overline{\mathbf{z}})$ is the volume constraint, V_{max} is the maximum allowable volume of material, and $\mathbf{v}$ is the vector of volumes of each element. The physical density vector, denoted by $\overline{\mathbf{z}}$, is obtained by first applying the density filter operator [38,39], and then the Heaviside operation to the design variable vector $\mathbf{z}$. The filter operation multiplies the design variable by the matrix $\mathbf{P}$, whose entry P_{ij} is defined as:

$$P_{ij} = \frac{\max\left(R - \left|\mathbf{x}_i^\star - \mathbf{x}_j^\star\right|,\ 0\right)}{\sum_{j=1}^{M} \max\left(R - \left|\mathbf{x}_i^\star - \mathbf{x}_j^\star\right|,\ 0\right)} \tag{2}$$

where R denotes the filter radius that controls the minimum length scale, and $\mathbf{x}_i^\star$ and $\mathbf{x}_j^\star$ denote the centroid of the ith and jth elements of the fine mesh. After we apply the filter operation, we perform the Heaviside projection on the filtered variables. To ensure differentiability of the projection operation, we use a smooth approximation of the Heaviside function described in [37] as:

$$\overline{\mathbf{z}} = \mathcal{H}(\mathbf{P}\mathbf{z}) = \frac{\tanh(\beta\eta) + \tanh(\beta(\mathbf{P}\mathbf{z} - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))} \tag{3}$$

where η is the value of the threshold for the Heaviside function and β controls the sharpness of such function. Typical values for these parameters are $\eta = 0.5$ and $\beta = 10$. The physical density vector $\overline{\mathbf{z}}$ represents the material distribution. For each element in the discretized design domain, the physical density lies in the interval $\overline{z}_i \in [0, 1]$ with 0 representing void and 1 representing solid. We use the Finite Element analysis (FEA) to compute the displacement vector $\mathbf{u}$ based on the linear elastic model. In the FEA, we define the material interpolation stiffness function using the SIMP model as

$$E_i = \epsilon + (1 - \epsilon)(\overline{z}_i)^p \tag{4}$$

where ϵ is the ersatz stiffness, which is set to be a small value (in this case 10^{-4}) to prevent the stiffness matrix, $\mathbf{K}(\overline{\mathbf{z}})$, from becoming singular, and the exponent p is the SIMP penalization factor. Higher values of p penalize more the intermediate values of the design variable $\mathbf{z}$, promoting a final design with only zeros and ones. Once we compute the material interpolation stiffness function, we can assemble the stiffness matrix as:

$$\mathbf{K}(\overline{\mathbf{z}}) = \bigwedge_{i=1}^{M} \mathbf{k}_i, \quad \text{with} \quad \mathbf{k}_i = E_i(\overline{z}_i)\mathbf{k}_0, \tag{5}$$

in which $\mathop{\mathbb{A}}_{i=1}^{M}$ stands for the FEA assembly operator, $\mathbf{k}_i$ are the element stiffness matrices, and $\mathbf{k}_0$ is the stiffness matrix of a solid element. We highlight that the assembly of the stiffness matrix, and the solution of the FEA, is only performed for the coarse-resolution mesh, reducing the overall computational cost.

3.1. Sensitivity analysis

In order to solve the problem in Eq. (1), we rely on gradient-based optimization algorithms, which require the sensitivity information of the objective and constraint functions with respect to the design variables. This section presents the analytical derivation of the sensitivity information. The sensitivity of the objective function with respect to the interpolated stiffness is given by:

$$\frac{\partial C(\mathbf{z})}{\partial E_i} = -\mathbf{u}^T \frac{\partial \mathbf{K}(\overline{\mathbf{z}})}{\partial E_i}\mathbf{u} = -\mathbf{u}^T \mathbf{k}_0 \mathbf{u}. \tag{6}$$

Further using the chain rule to account for the density filter, the Heaviside projection operator and the SIMP interpolation gives

$$\frac{\partial C(\mathbf{z})}{\partial \mathbf{z}} = \mathbf{P}^T \left[\text{diag}\left(\frac{\beta(1-\tanh(\beta(\mathbf{P}\mathbf{z}-\eta)^2))}{\tanh(\beta\eta)+\tanh(\beta(1-\eta))} \right) \text{diag}\left(p(1-\epsilon)\overline{\mathbf{z}}^{(p-1)} \right) \frac{\partial C(\mathbf{z})}{\partial \mathbf{E}} \right], \tag{7}$$

where diag(**a**) stands for a diagonal matrix with the elements of vector **a** on the main diagonal.

In addition, the sensitivity of the volume constraint function is straightforwardly obtained as

$$\frac{\partial g_V(\overline{\mathbf{z}})}{\partial \mathbf{z}} = \frac{\partial(\mathbf{v}^T\overline{\mathbf{z}} - V_{max})}{\partial \mathbf{z}} = \mathbf{P}^T \left[\text{diag}\left(\frac{\beta(1-\tanh(\beta(\mathbf{P}\mathbf{z}-\eta)^2))}{\tanh(\beta\eta)+\tanh(\beta(1-\eta))} \right) \mathbf{v} \right]. \tag{8}$$

Once we have the sensitivity of the objective and the constraint functions, we use the Optimality Criteria (OC) method [3] to update the design variables. For more details on the derivation of the sensitivity and the OC method, we refer the readers to the Refs. [3,40].

3.2. Two-resolution topology optimization

Similar to the previous work [1], we use a two-resolution topology optimization setup to enable a scalable sensitivity prediction using the machine learning model. In the two-resolution setup, a fine mesh is embedded in a coarse mesh. The fine mesh contains the stiffness field (the design), which is mapped to the coarse mesh, where we solve the state equations (notice that the linear system associated with the coarse mesh is relatively small and can be solved quickly). We use a simple averaging approach to define the mapping of stiffness distribution from the fine mesh to the coarse mesh. The stiffness of each coarse-resolution element is taken as the average value of those of the fine-resolution elements which lie inside that coarse-resolution element, namely,

$$E_e^C = \frac{1}{|\omega_e|}\sum_{i\in\omega_e}(E_i) \tag{9}$$

where E_e^C is the stiffness of the eth element of the coarse mesh, ω_e is the set of the elements of the fine mesh that lie inside the coarse element e with $|\omega_e|$ measuring its number of elements, and E_i is the SIMP-penalized stiffness of the fine-resolution element i (see Eq. (4)). Once obtained, the coarse-resolution displacements are taken as input features to the ML model. They carry coarse-grained information about the mechanical response of each design, which allows the ML model to generalize to problems that it has not seen in the training data, such as problems with different domains and/or boundary conditions.

The two-resolution framework is summarized in Fig. 3. The difference in length scale between the fine and the coarse meshes is characterized by the block size (N_B), which is displayed in the bottom left corner of Fig. 3. The block size N_B is defined as the number of fine-mesh elements each coarse-mesh element contain along one of its edges. Thus, each coarse-mesh element contains $(N_B)^d$ number of fine-mesh elements and, accordingly, the ratio of the total number of elements in the fine mesh to that in the coarse mesh is $(N_B)^d$ as well, where d is the number of dimensions (i.e. 2D, 3D). The choice of N_B represents a trade-off between computational efficiency and prediction accuracy. As we increase the block size N_B, we make the proposed framework more efficient because

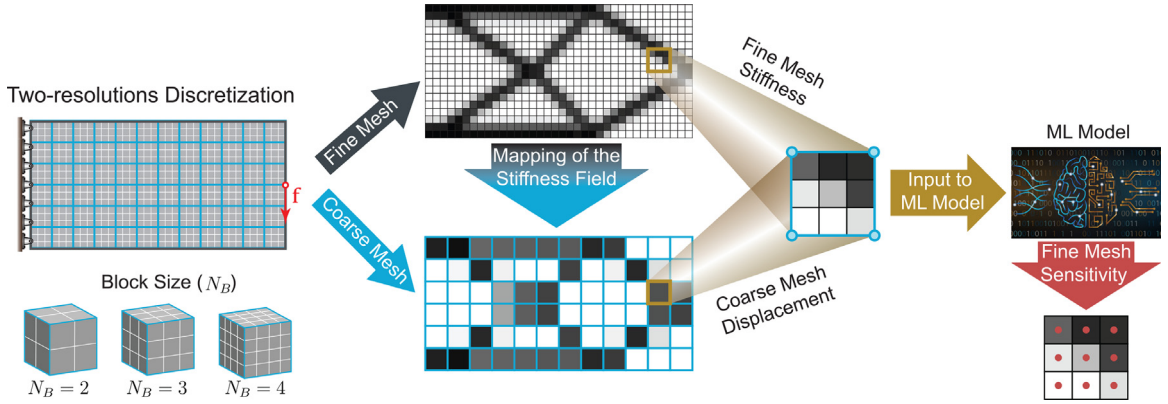

Fig. 3. A schematic illustration of the two-resolution approach used in this framework. We have a fine-resolution mesh and a coarse-resolution mesh that provide information to the ML model. The number of the fine-mesh elements contained in each side of a coarse-mesh element is defined as block size (N_B), as displayed in the bottom left corner.

we reduce the size of the system of equations associated with the coarse-mesh FEA. However, a larger block size N_B also decreases the prediction accuracy of the framework, because less local information is contained in the coarse-resolution displacements. For 3D design problems, we find that setting $N_B = 2$ or $N_B = 3$ leads to a good balance between the efficiency and prediction accuracy of the proposed framework.

Furthermore, the interaction between filter radius and block size deserves special attention because, if the filter radius is much smaller than the block size (e.g. 5 times smaller), the minimum length scale of the structural features of the optimized result can be much smaller than the coarse-mesh element. In such cases, it is difficult to represent these small features in the coarse FEA. In practice, to achieve a reasonable representation of the underlying structure in topology optimization, the filter radius should be at least 2 to 3 elements wide [39]. Typical results in this paper include block sizes that are 2 and 3 elements wide, and filter radius around 4 elements wide. As a word of caution, we remark that any filter radius that is small enough to adversely interact with the block size will also be small enough to cause problems in traditional topology optimization techniques.

4. Machine learning enhanced topology optimization

This section introduces a ML enhanced topology optimization framework, which follows the developments of [1]. With the proposed framework, we are able to significantly reduce the computational cost associated with topology optimization while maintaining its accuracy. This is achieved by the introduction of an inexpensive deep neural network-based surrogate model to directly predict the sensitivity of the objective function without performing the fine-mesh FEA computation.

As an overview, the ML model takes the stiffness distribution of the fine-resolution mesh (i.e., $E_i(\overline{z}_i)$ defined in Eq. (4)) and the displacement field on the coarse-resolution mesh as input features, and outputs the sensitivity of the objective function with respect to the material stiffness of the fine mesh, $\partial C(\mathbf{z})/\partial E_i$, see Eq. (6). Once we obtain this sensitivity, we can then efficiently compute the sensitivity with respect to the design variables using the chain rule, as described in Eq. (7), which are then used to update the fine-resolution design. Because we use the interpolated stiffness as input E_i, and predict the sensitivity in regard to the stiffness itself, the ML model is independent of the material interpolation function and filter used in the topology optimization. This means that the model, once trained, can be used with any combination of material interpolation functions, penalization factor, and/or filter radius, adding flexibility and robustness to this framework.

The remainder of this section explains the details of the framework. First, we introduce the deep neural network architecture of the ML model by elaborating the details of the convolutional layers and the fully connected layers. Second, we describe the input normalization procedure that helps the ML model to handle a broad range of problems of different scales. Finally, we discuss the independent training and prediction procedure, which is one of the main contributions of this work.

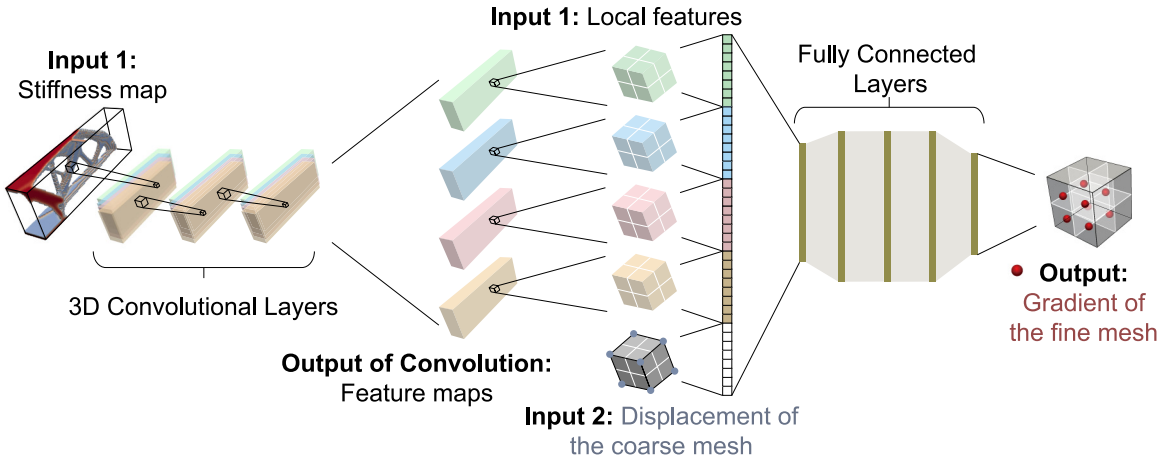

Fig. 4. The neural network architecture used for the ML model in this work. The neural network consists of 3D CLs that extract local features from the stiffness map. The local features, together with the displacements of the coarse mesh, serve as the inputs to fully connected layers that ultimately predict the sensitivity of the fine-resolution mesh.

4.1. Neural network architecture

A deep neural network (DNN) [41] is used as the ML model. The architecture of the DNN is inspired by the models used for image classification [42], in which convolutional layers (CLs) extract the main features of an image and fully connected layers are subsequently used for the final classification. In a similar manner, we use 3D CLs to extract information from the discretized stiffness map of the structure. The discretized stiffness map can be interpreted as a 3D image, where elements in the discretization are the voxels of the 3D object, in analogy to the pixels of a 2D image. Therefore, similar to the way 2D CLs extract local features from images for image classification, we use 3D CLs to extract local features from the stiffness map.

Subsequently, the features extracted by the 3D CLs, together with the displacements solved on the coarse mesh, serve as input to fully-connected layers that perform the final gradient prediction. The full DNN architecture is illustrated in Fig. 4.

4.2. 3D convolutional layers

We use 3D CLs in our ML model because they are capable of extracting relevant local information from the stiffness map while preserving geometrical information of the input domain at a reasonable computational cost. The 3D CLs achieve this by extracting local features from the stiffness map the same way 2D CLs extract features from images (see Fig. 5(a)). The CLs operate by the convolution of a kernel throughout the domain of interest. A kernel is a function with local support that is used in the convolution. The parameters of this kernel function are learned through the ML training procedure.

In the case of 3D CLs, these kernel functions can be represented as 3D tensors. During the convolution process, the kernel strides over the domain of interest (i.e., the input stiffness map), as displayed in Fig. 5(c), which is generally much bigger than the kernel itself. At each stride of the kernel operation, the entries of the kernel are multiplied by the entries of the input at the position where the kernel is positioned, and then the result is summed together, which gives the value of the voxel of the output that corresponds to that position. This process is further illustrated in Fig. 6. The entries of the kernel are learned through the training of the ML model to represent the feature of interest. During the convolution process, whenever the kernel encounters the feature of interest in the input, the corresponding output voxel will have a high value, giving us an indication of where we can find that feature of interest on the input map (see Fig. 5(b)).

The output of the convolution is a 3D object whose value in each voxel is obtained through the process outlined above. This 3D object is a map of the feature represented by the kernel. In general, a 3D CL can have multiple kernels, each of which generates one feature map. Thus, the ML model can represent as many features as we need

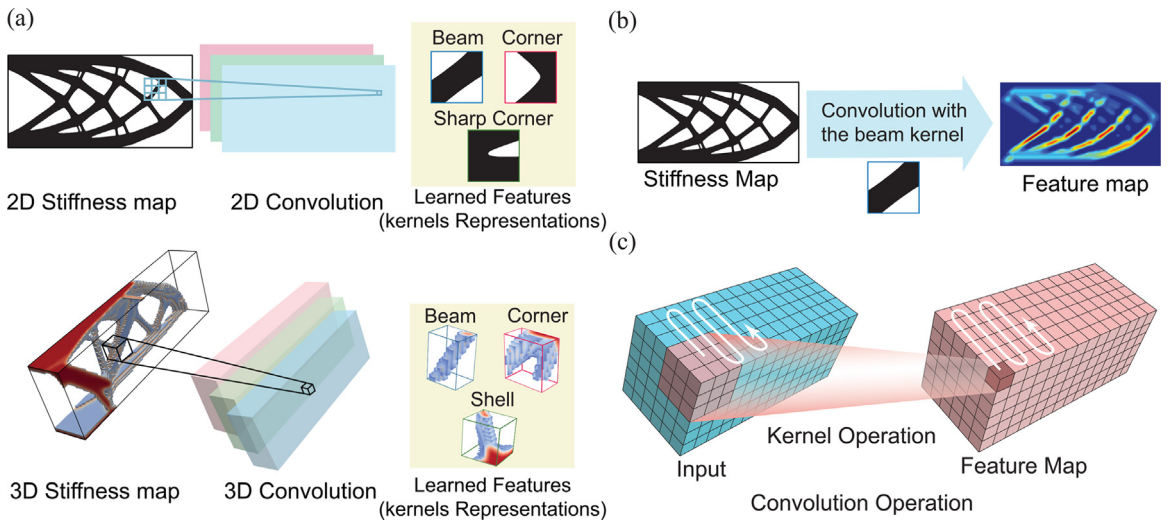

Fig. 5. The concepts related to CLs of the neural network architecture: (a) displays the analogy between the use of CLs for image processing and our use of CLs on the stiffness maps; (b) displays the convolution operation over an image and its resulting feature map; (c) displays the convolution operation with the kernel stride and kernel operation to generate the feature map in 3D. We highlight that the learned features and the kernels of the CLs are intrinsically different, and the learned features presented in the figure are merely representations of the kernels, which are learned during the training of the ML model.

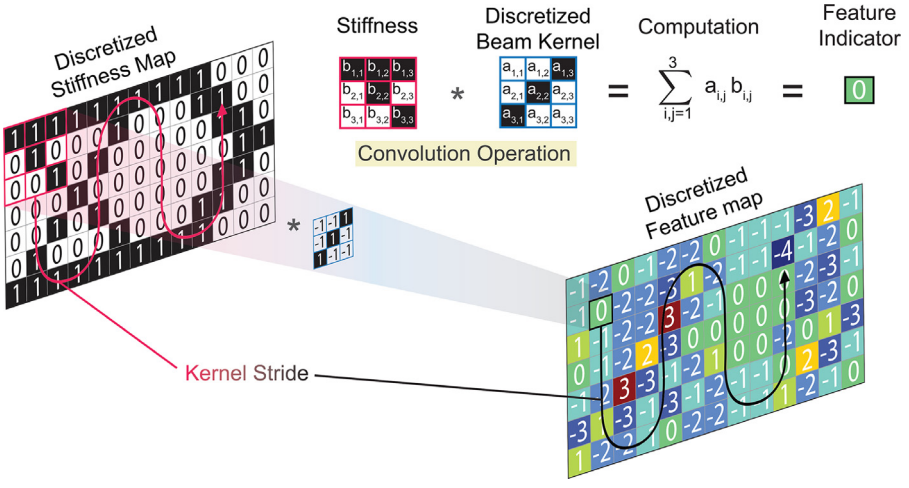

Fig. 6. The demonstration of the convolution operation in a 2D discretized stiffness map, with the mathematical formulation, the kernel stride representation, and the output feature map.

to characterize our problem. In the DNN used in this work, we chose the stride and padding values of the CL such that the input stiffness map and the output feature maps of the CL have the same dimensions.

4.3. Fully-connected layers

Fully-connected layers are a powerful ML model capable of representing complex functions. Each fully-connected layer is composed of a matrix of weights, a vector of biases, and an element-wise activation function (except for the output layer, which does not have an activation function). The input vector of a fully-connected layer is multiplied by the matrix of weights, then added to the bias vector, and passed through the activation function, which introduces non-linearity to the model. The entries of the weight matrix and the bias vector of each layer

are learned through the ML training procedure to fit the function of interest. Several fully-connected layers can be stacked in sequence to increase the representation capability of the resulting model.

In our ML model, the fully-connected layers receive inputs including the local features extracted by the CLs and the displacements obtained on the coarse-resolution mesh. The corresponding output is the sensitivity of the objective function with respect to the interpolated stiffness on the fine-resolution mesh. To ensure scalability, the fully-connected layers only receive local information, that is, the information associated with only one of the coarse-mesh elements. This local information consists of the displacements associated with the nodes of the selected coarse-mesh element and the features extracted by the CLs from the stiffness map which lie inside this coarse-mesh element. Based solely on this local information, the ML model then predicts the sensitivity of those fine-mesh elements located inside the selected coarse-mesh element. This localized prediction is performed for every coarse-mesh element until we obtain the full sensitivity vector of the fine-resolution mesh. This local training and prediction strategy makes the ML model independent of the problem size, and greatly reduces the memory requirements of the ML model.

4.4. Residual network

The use of the DNN architecture introduces several challenges during training. One of the challenges is the vanishing gradient issue illustrated in Fig. 7(a). The gradient here is referred to the gradient of the loss function of the DNN. The vanishing gradient problem happens because, as the gradient is back-propagated through the neural network layers, it decreases in magnitude. If the magnitude of the gradient becomes too small, then the training becomes slow as the update of the learnable parameters is proportional to the gradient.

To address this problem, we adopted a residual network-type (ResNet) strategy [19], in which we add connections between the layers of the neural network, as shown in Fig. 7(b). These connections map the input of a layer directly to its output. By adding these connections, we offer the gradient a path through which it can propagate backward more easily, and thus the gradient signal associated with the first few layers becomes stronger. A stronger gradient signal allows us to train the DNN more effectively and efficiently.

4.5. Data normalization and batch normalization

Data pre-processing is crucial to the performance of ML models [43], because it can simplify the mapping between the input and the output data, making it easier for the ML model to learn such mapping. Data pre-processing also helps the ML model by reducing the feasible domains of the input and output data. For example, data normalization can transform the data from the entire space of the real numbers to an interval of [0, 1], leading to a more manageable feasible space for the ML model.

In topology optimization, the input and output data can take a wide range of values depending on various design parameters, such as the load magnitude, the domain geometry, and the material properties. This wide range in data values can lead to instability during training and, consequently, poor accuracy of the prediction of the ML model. For this reason, we normalize both our input and target data. The input displacements are normalized using standardization, in which we subtract the mean of the data and then divide by the standard deviation, as displayed in Eq. (10). The standardization transforms the data into a distribution with mean equal to zero and standard deviation equal to one and reads

$$\bar{\mathbf{u}} = \frac{\mathbf{u} - \mu(\mathbf{u})}{\sigma(\mathbf{u})} \tag{10}$$

where, $\bar{\mathbf{u}}$ is the normalized displacements by standardization, $\mu(\mathbf{u})$ is the mean of $\mathbf{u}$, and $\sigma(\mathbf{u})$ is the standard deviation of $\mathbf{u}$. We also normalize the target data, i.e. the sensitivity of the compliance with respect to the stiffness of the fine mesh, $\partial C(\mathbf{z})/\partial E_i$. The target data is normalized using the sensitivity of the compliance of the coarse mesh with respect to the stiffness of the coarse mesh, $\partial C^C(\mathbf{z})/\partial E_e^C$, which can be computed with the displacements of the coarse mesh. We perform this normalization by dividing the sensitivity of the fine-mesh element i, $\partial C(\mathbf{z})/\partial E_i$, by the coarse-mesh sensitivity of the element (assuming to be e, such that $i \in \omega_e$) that contains that fine-mesh element, $\partial C^C(\mathbf{z})/\partial E_e^C$.

In addition, it is also beneficial to normalize the data between each layer of the neural network. To that end, we add batch normalization [44] between each layer of the proposed DNN. Batch normalization, as described in

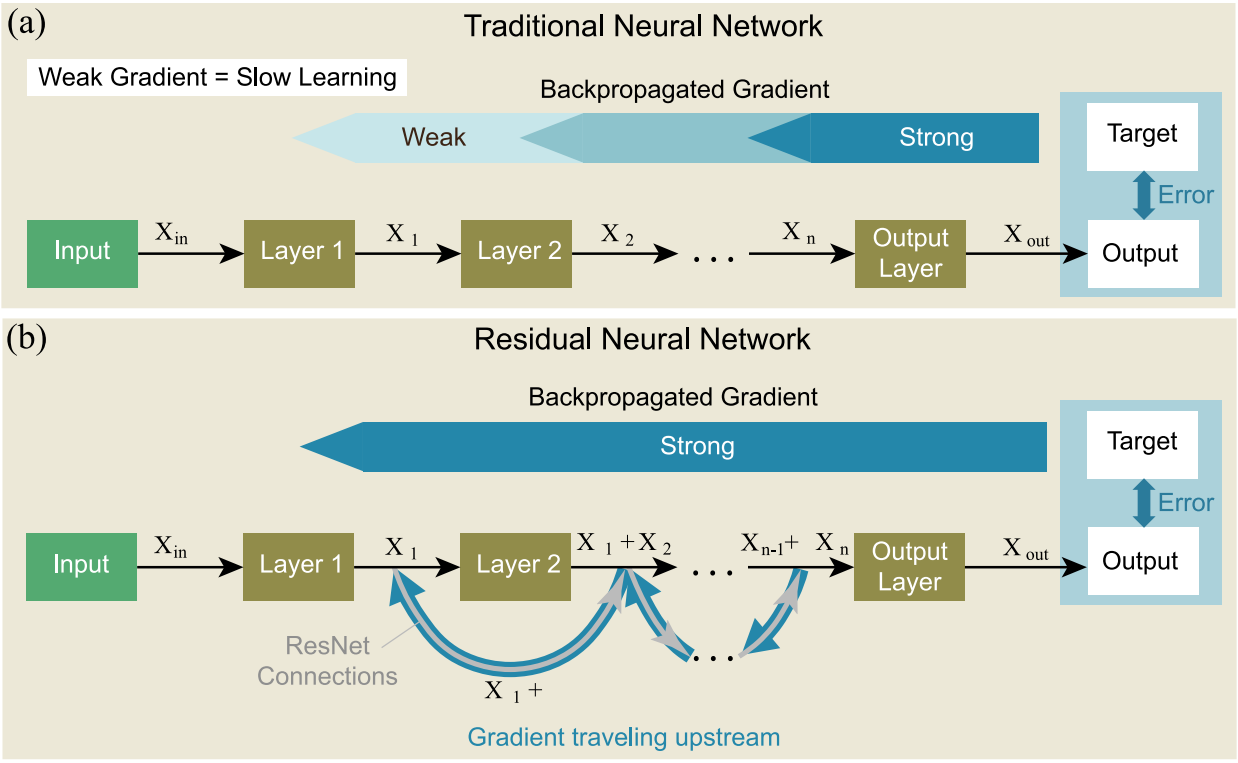

Fig. 7. Schematics show (a) the vanishing gradient problem on traditional neural networks, and (b) how residual networks solve this vanishing gradient problem.

Eq. (11), operates by standardizing the input data batch and then performing a linear shift to the result. It is defined as

$$\bar{\mathbf{x}} = \gamma \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})} + \xi \tag{11}$$

where $\bar{\mathbf{x}}$ is the output of the batch normalization, $\mathbf{x}$ is the input, $\mu(\mathbf{x})$ is the mean value of the batch, $\sigma(\mathbf{x})$ is the standard deviation of the batch, and γ and ξ are the linear shift parameters. The shift parameters are learned during training and their introduction improves the representation capabilities of the neural network.

4.6. Independent training and application of the ML model

Unlike the previous work [1], in which the training of the ML model was done during the topology optimization, this work completely separates the training of the ML model from its application. By doing so, we can further increase the efficiency of our framework by eliminating the cost of the initial training and online updates [1].

In the proposed framework, during training, the state equation is solved on both the fine and coarse meshes and the sensitivity analysis is performed on the fine mesh, so that the ML model learns the mapping of the fine-mesh sensitivity from the fine-mesh stiffness distribution and the coarse-mesh mechanical response (i.e., displacements). During the prediction stage, we apply the trained ML model to a (different) problem of interest, and only require the solution of the state equation on the coarse mesh. Because both training and prediction are done locally, the block sizes N_B of the problems used in the training and in the prediction stage needs to be the same. However, other parameters, such as the total numbers of elements and design variables, do not need to be the same for the problems in training and in prediction.

The capability of the ML model to generalize from the training problems to different problems is grounded on two ideas:

1. The sensitivity of the objective function is uniquely determined when given the information of the geometry of the problem, the coarse-resolution displacement, the fine-resolution stiffness distribution, and the block size N_B.
2. The Saint-Venant principle [45], which allows us to use the local information for the prediction.

To demonstrate the first idea, we recast the sensitivity for the ith fine-mesh element as:

$$\frac{\partial C(\mathbf{z})}{\partial z_i} = -\mathbf{u}^T \frac{\partial \mathbf{K}(\overline{\mathbf{z}})}{\partial z_i}\mathbf{u} \tag{12}$$

where we recall that $\mathbf{u}$ and $\mathbf{K}$ denote are displacements and the global stiffness matrix of the fine mesh, respectively. In addition, we express the fine-mesh displacement $\mathbf{u}$ in terms of the coarse-mesh displacement, denoted by $\mathbf{u}^C$, and the stiffness matrix of the coarse mesh, denoted by $\mathbf{K}^C$, as:

$$\mathbf{u} = \mathbf{K}^{-1}\mathbf{f} = \mathbf{K}^{-1}\mathbf{N}\mathbf{f}^C = \mathbf{K}^{-1}\mathbf{N}\mathbf{K}^C\mathbf{u}^C \tag{13}$$

where $\mathbf{f}^C$ is the load vector of the coarse mesh, and $\mathbf{N}$ is a matrix that maps the coarse load vector to the fine-mesh load vector $\mathbf{f}$.

Substituting the expression (13) into Eq. (12) gives

$$\frac{\partial C(\mathbf{z})}{\partial z_i} = -\left(\mathbf{K}^{-1}\mathbf{N}\mathbf{K}^C\mathbf{u}^C\right)^T \frac{\partial \mathbf{K}(\overline{\mathbf{z}})}{\partial z_i}\left(\mathbf{K}^{-1}\mathbf{N}\mathbf{K}^C\mathbf{u}^C\right). \tag{14}$$

Notice that, when given the geometry of the problem, the block size, and the fine-mesh densities, we can obtain the stiffness matrices, $\mathbf{K}^C$ and $\mathbf{K}$, for both of the coarse- and the fine-resolution meshes, as well as the interpolation matrix $\mathbf{N}$. Furthermore, the coarse-mesh displacement, $\mathbf{u}^C$, is known as an input feature. Therefore, we have all the information necessary to compute the sensitivity of the objective function.

However, Eq. (14) requires global information of the problem, namely, it requires the knowledge of the global geometry as well as the full density and coarse-mesh displacement vectors. Although, this suggests a global approach in which a ML model is fed the information of the whole system in order to make the prediction, such a global approach would require prohibitively high computational resources, and, thus, is intractable. To avoid this, we make use of the Saint-Venant principle [45], which states that, for linear elasticity boundary value problems, the influence of the strain field decays quadratically with the distance. Therefore, we can use local information to predict the sensitivity without losing substantial accuracy.

5. Influence of the training set

The influence of the data set used for training is quite relevant in this study. By collecting our training data from the solution of actual topology optimization problems, as illustrated by Fig. 11, we obtain a consistent ML model in the sense that the data is directly related to the problems that we are trying to solve. Yet, insights are needed in order to appropriately select the training problems to ensure sufficient generalizability.

We discover that the relation between the problems in the training set and the problems that we aim to apply the model to is directly related to the mechanics of those problems. More specifically, it is closely related to the type of load conditions in the problems. For example, a ML model trained by problems subjected mainly to bending loads will perform poorly in a problem subjected to torsional loads, as illustrated in Fig. 8(a). Instead, if we include both problems with bending loads and problems with torsional loads in the training set, the trained ML model will perform well for both torsional and bending problems, as illustrated in Fig. 8(b).

Problems subjected to bending, and problems subjected to torsion, however, are not always easily identifiable. Furthermore, other types of problems with different loading conditions might also result in poor performance of the ML model if not included in the training set. For those reasons, this subsection develops an approach to classify topology optimization problems according to the loading configuration. The idea is to introduce a measure of the similarity/dissimilarity between two topology optimization problems. Ideally, the measure should satisfy the following three criteria. First, the measure should be computable even before we run the optimization. Thus, the measure should not be dependent on the material distribution over the domain. Second, the measure should be able to quantify if the ML model will be able to generalize the data in the training set to the problem at hand. To achieve that, the measure needs to be based on the relevant parameters for the ML prediction, which is the mechanical behavior of the structure in this work. Third, the measure should be invariant under rotations and/or

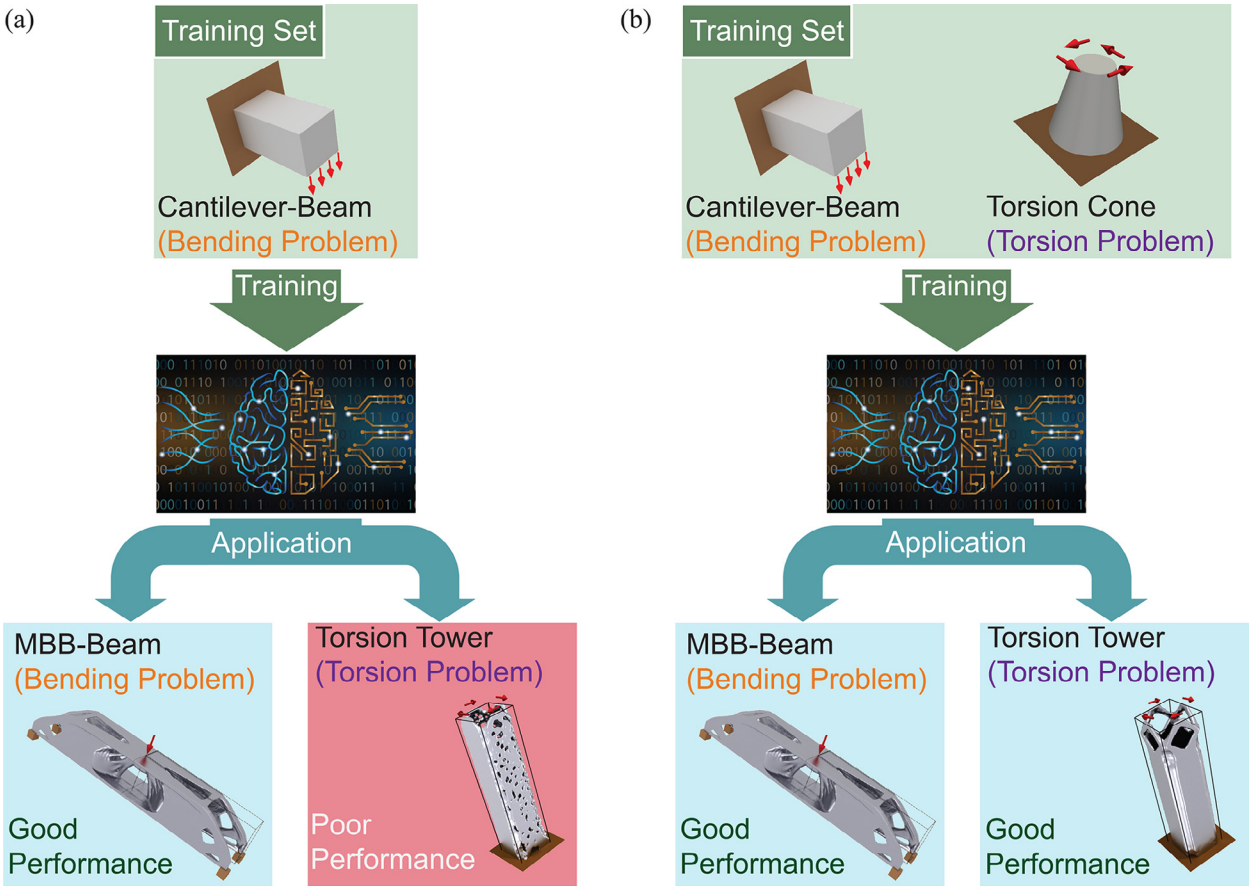

Fig. 8. The influence of the training set on the performance of the ML model: (a) models trained with only bending problems perform badly when they encounter a torsion problem; (b) if we expand our training set to contain bending problems and torsion problems the model performs well for both types of problems.

translations of the domain and boundary conditions, because such operations do not change the overall mechanical behavior of the structure.

In order to satisfy all these criteria, we develop a measure based on the stress distribution in the design domain of the topology optimization problem. To satisfy the first criterion, we perform the stress analysis on the design domain with a homogeneous distribution of materials. The stress tensor is not invariant under rotations of the domain. Thus, to satisfy the third criterion, we define the measure as a function of the principal stresses, denoted by σ_1, σ_2 and σ_3 such that $\sigma_1 \geq \sigma_2 \geq \sigma_3$. We use these principal stresses to compute the diameters of the Mohr's circles [46] of the stress state given by $d_1 = \sigma_1 - \sigma_3$, $d_2 = \sigma_1 - \sigma_2$ and $d_3 = \sigma_2 - \sigma_3$, as displayed in Fig. 9. The Mohr's circles are 2D representations of the stress tensor that have been widely used to study stress states. We set the diameters of the Mohr's circles as the components of a 3D vector, $\mathbf{w}$, and then normalize this vector as

$$\mathbf{w} = [\sigma_1 - \sigma_3, \sigma_1 - \sigma_2, \sigma_2 - \sigma_3]^T \qquad \overline{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|}, \tag{15}$$

where $\|\cdot\|$ stands for the Euclidean norm of a vector. By normalizing this vector, we map all the possible values of $\overline{\mathbf{w}}$ to a sphere. In addition, since $d_1 = d_2 + d_3$, the possible values of $\overline{\mathbf{w}}$ are further restricted to a circle. This circle intersects the xy-plane at the point $[1/\sqrt{2}, 1/\sqrt{2}, 0]^T$. We then measure the angle, θ, between the vector $\overline{\mathbf{w}}$ and the vector $[1/\sqrt{2}, 1/\sqrt{2}, 0]^T$. This angle gives the final measure that we seek (see Fig. 9).

By measuring θ for every element in the mesh, we obtain a distribution that can be represented by a histogram. The histograms for four different problems, namely the MBB beam, the torsional tower, the cantilever beam, and the torsional cone, are displayed in Fig. 10. We can see from the figure that the histogram of the MBB beam (Fig. 10(a))

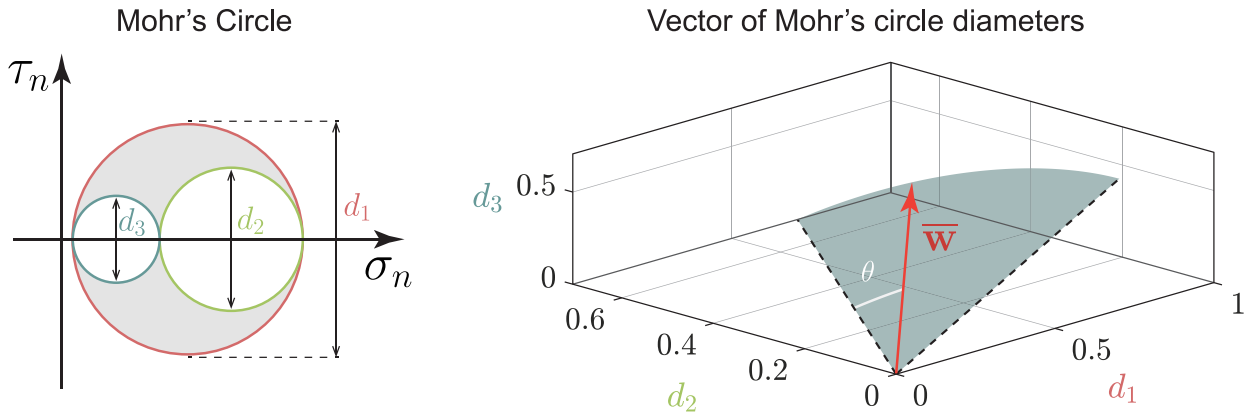

Fig. 9. Mohr's circle representation on the left and the θ measure of vector $\overline{\mathbf{w}}$ on the right.

Table 1
Table of the measured difference between the histograms of four different problems using Eq. (16).

	MBB			
MBB	0	Cantiliver beam		
Cantiliver	16	0	Torsion tower	
Torsion tower	65	58	0	Torsion cone
Torsion cone	53	45	18	0

and the Cantilever beam (Fig. 10(c)) are similar to each other, reflecting that they are both subjected to bending. The histogram for the torsion tower (Fig. 10(b)) and the torsion cone (Fig. 10(d)) are also similar, reflecting that they are both subjected to torsion. Thus, we can obtain a quantitative measure of the similarity between two problems by computing a measure of the difference between their histograms. This measure of the difference between histograms is computed as:

$$D(h_1, h_2) = \sqrt{\sum_{i=1}^{nBins} (b_{1i} - b_{2i})^2} \tag{16}$$

where h_1 and h_2 denote two histograms, $nBins$ is the number of bins in both histograms, b_{1i} is the percentage of elements of histogram 1 in bin i and b_{2i} is the percentage of elements of histogram 2 in bin i. The values of this difference for the four problems displayed in Fig. 10 are presented in Table 1. Notice that the value of this difference follows the expected trend, being smaller for problems subjected to similar loading conditions (bending/bending, or torsion/torsion), and being larger for problems under different loading conditions (bending/torsion).

6. Numerical results

This section presents the numerical results that demonstrate the capabilities of the framework. For all the results presented in this section, we start with a SIMP penalization factor $p = 1$ (see Eq. (4)), and gradually increase it by 0.5 every 25 iterations, until we reach $p = 3$. As for the Heaviside parameters (see Eq. (3)), we set $\eta = 0.5$, and we start with $\beta = 1$, and after we reach 140 iteration, we gradually increase β by 0.5 every 5 iterations until we reach $\beta = 10$. The maximum number of iterations is 200. We set the solid material Young's modulus to 1 and the Poisson ratio to 0.3. All meshes (including both coarse-resolution and fine-resolution ones) used here are regular hexahedral meshes.

The DNN used has 5 convolutional layers, each with 50 ($3 \times 3 \times 3$) kernels, and parametric rectified linear unit as activation functions (PReLU) [47], followed by 10 fully connected layers each with 300 neurons and PReLU activation functions after each layer, except for the output layer, which is not followed by an activation function. The main implementation, used to obtain the results, is developed in python 3.6, using the PyTorch library [48] for the

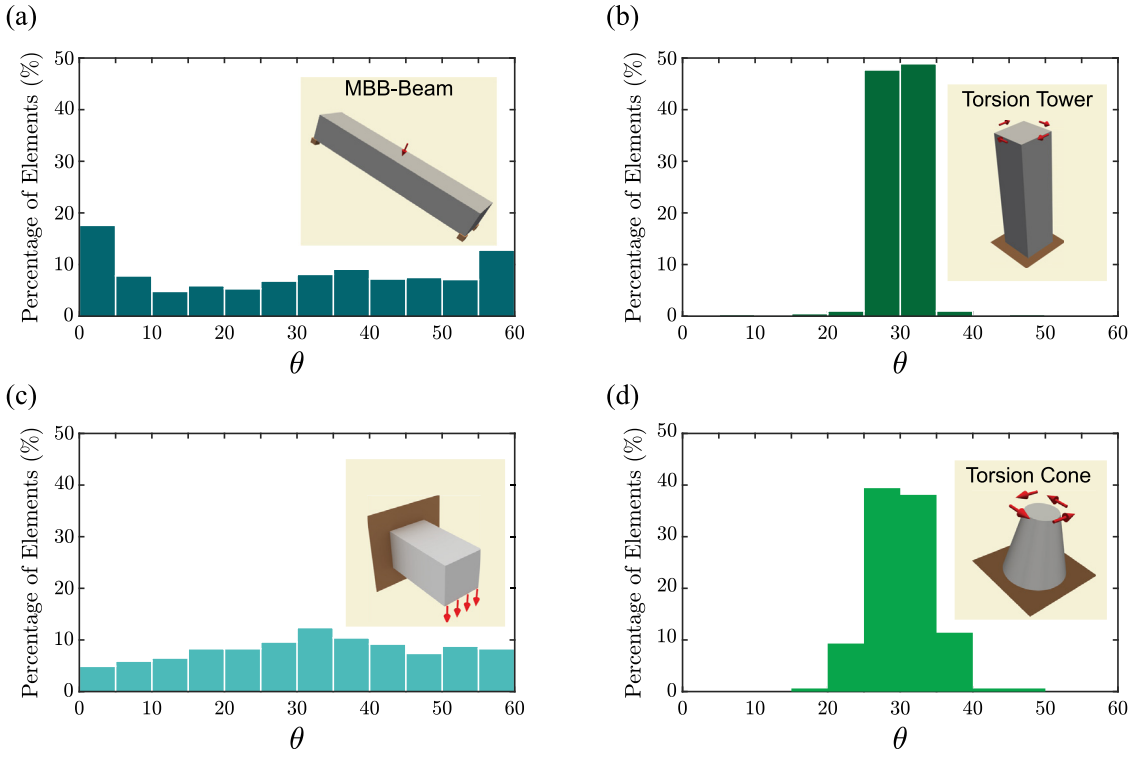

Fig. 10. Histograms of the measure θ distribution, based on the stress states, for different problems.

implementation of the ML model. The solution of the FEA linear system is performed by an in-house C++ /CUDA code developed in the research group of the last author at the Georgia Institute of Technology.

6.1. Training the machine learning model

To train the ML model, we use four topology optimization problems: the MBB beam, the Cantilever beam, the L-shaped beam, and the torsion tower. The design domains and boundary conditions of these four problems are displayed in Fig. 11. To collect data for the training, the four problems are solved using the traditional topology optimization, i.e. solving the fine-mesh FEA at each iteration. At the same time, we also solve the coarse-mesh FEA at each iteration. By solving both the fine- and coarse-mesh FEA, we obtain the information on how the coarse-mesh solution is related to the fine-mesh sensitivity. To solve these problems, we used fine-resolution meshes with 165,888 elements (for the MBB beam), 221,184 elements (for the cantilever beam), 262,144 elements (for the L-shaped beam), and 186,624 elements (for the torsion tower). The size of the coarse-resolution meshes is determined via the selected block size value N_B. Notice that the sizes of the fine-resolution meshes used for training are much smaller than those of the fine-resolution meshes used during the actual prediction stage of the ML model. The relatively smaller mesh size of the training problems means that they do not require a vast amount of computational resources and thus can be obtained relatively more efficiently. Regarding the parameters of topology optimization, we set a volume fraction of 0.12 and a filter radius of 0.08 for all the problems in the training set.

Once the training problems are solved and the data are collected, we train two ML models, one for block size $N_B = 2$ and the other for block size $N_B = 3$, both using the ADAM [49] optimizer. *We emphasize that, once trained, these two ML models are used to solve all the design problems presented in this section without any re-training or updating.*

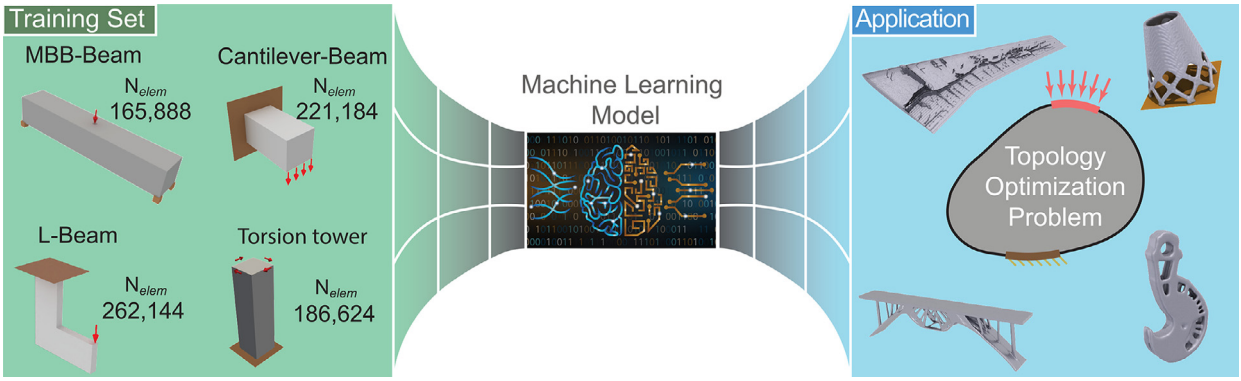

Fig. 11. Training set containing the MBB beam, the cantilever beam, the L-shaped beam, and the torsion tower domains used to train the machine learning model. We use a small training set to teach our machine learning model the underlying physics, so that we can obtain a general model to solve any topology optimization problem.

6.2. The MBB beam example

The first example that we present is the MBB beam problem for which the geometry[1] and boundary conditions are displayed in Fig. 12(a), with a unitary load. We take advantage of the symmetry of the geometry and boundary conditions of this problem and solve only one-quarter of the design domain. The goal of this example is to demonstrate the robustness of the proposed framework.

To study how the model performs as we scale up the problem, we vary both the mesh size and block size while maintaining a constant volume fraction of 0.12, and filter radius of 0.08. Notice that, as we increase the block size N_B, the corresponding coarse mesh size decreases for a given fine mesh. This reduction in the coarse mesh size increases the efficiency of the framework because the coarse mesh FEA solution becomes faster. However, the increase in block size tends to decrease the prediction accuracy of the ML model.

The results are presented in Fig. 12(b). From the results, we see that the solutions for block size $N_B = 2$ and block size $N_B = 3$ are different even for the same fine-resolution mesh. The different solutions correspond to different local optima of the optimization problem. This is a manifestation of the non-convexity of the optimization problem, in which small perturbations in the sensitivity can lead to completely different solutions.

In most cases, these local optima behave almost equivalently as they possess almost identical compliance, although sometimes the compliance of the results obtained with $N_B = 2$ is slightly better (by only a negligible amount) than those obtained with $N_B = 3$. Because of the similar mechanical performance of the results, we conclude that the proposed ML model with block size up to $N_B = 3$ is sufficiently accurate for high-quality topology optimization solutions.

Next, to investigate if the trained ML model can solve problems with completely different topology optimization design parameters than those seen during training, we solve the MBB beam problem with a fixed fine-mesh containing $1,327,104$ elements and block size $N_B = 2$ with various combinations of volume fraction and filter radius.

The results are displayed in Fig. 13, in which we also compare the compliance of each design obtained with the proposed ML framework and that of the design obtained via standard topology optimization (termed as reference compliance). By comparing the compliance values, we conclude that all compliance values obtained by the proposed ML framework are within 1.5% range of the reference compliance values, indicating that the proposed ML-driven topology optimization is sufficiently accurate to obtain equivalent solutions to those of the standard topology optimization. Furthermore, we highlight that, although we set the filter radius and volume fraction to values that the ML model has not seen in training, we obtain consistent solutions (i.e. no hanging members nor disconnected components) unlike some of the existing imagine-based ML-driven topology optimization frameworks.

[1] We present dimensionless parameters for the geometry, and we used consistent units in all examples.

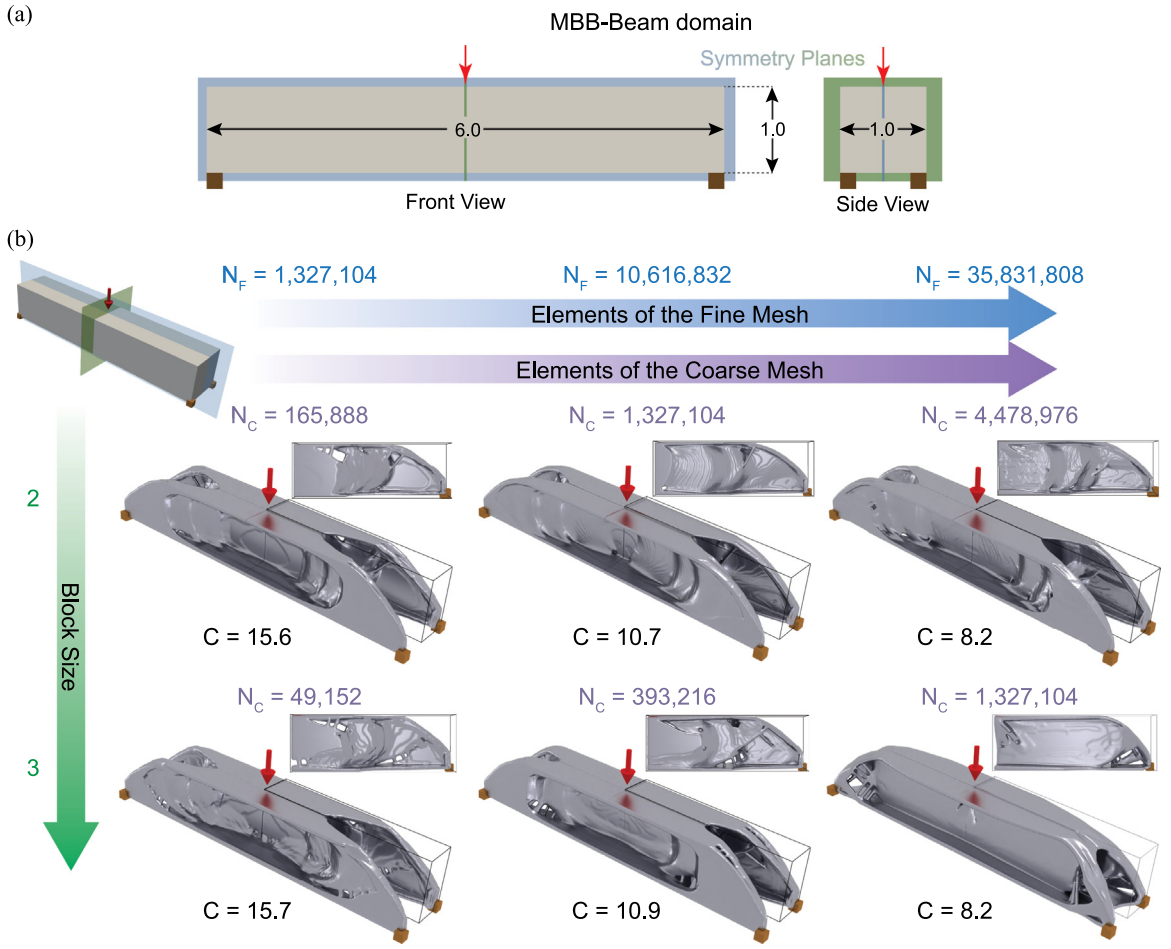

Fig. 12. Results for the MBB beam domain obtained through the proposed ML-driven approach. (a) The MBB beam domain with the appropriate dimensions (dimensionless units) and boundary conditions (unitary load). (b) Proposed framework results for the MBB beam varying mesh size, in which N_F is the size of the fine mesh, and N_C is the size of the coarse mesh.

6.3. The bridge example

The next example that we present is the bridge design example, for which the domain and boundary conditions are displayed in Fig. 14. Unlike the previous example, this bridge example is not part of the training set of the ML model. Therefore, the ML model has to generalize what it has learned from the problems in the training set to a problem with a completely different domain and boundary conditions. To showcase the proposed ML model has such generalization capability, we present the result of the bridge design, which is obtained on a $38,016,000$-element (fine-mesh) mesh with 0.12 volume fraction, 4.0 filter radius, and block size $N_B = 2$, in Fig. 15. With the proposed ML-driven topology optimization framework, this bridge example takes 18 hours to run. Due to limitations on computing hardware, we are unable to compute the exact compliance of the fine mesh, but we can estimate it by the compliance of the coarse mesh which is 31.3.

The demonstrated generalization capability of the ML model to problems it has not seen before indicates that the ML model is actually learning the underlying relation between the coarse-mesh displacements, fine-mesh stiffness and the fine-mesh sensitivity.

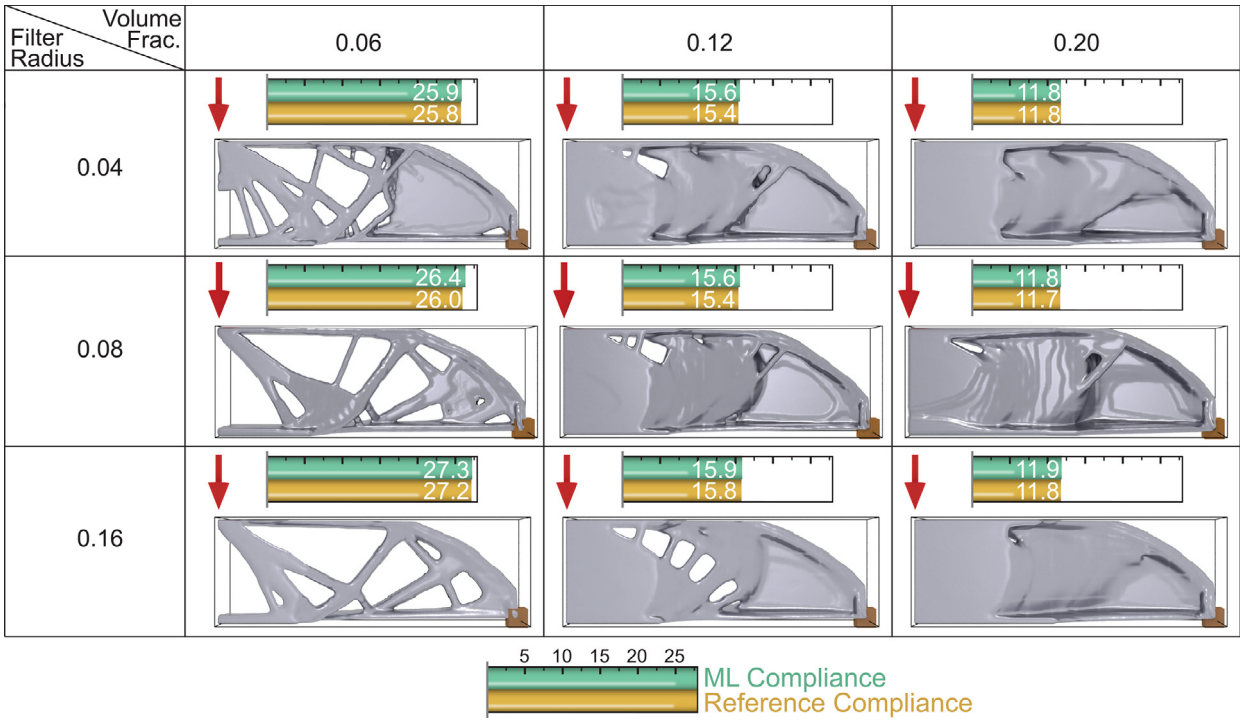

Fig. 13. The results obtained by the proposed framework for the MBB beam example with a $1,327,104$-element mesh varying volume fraction and filter radius.

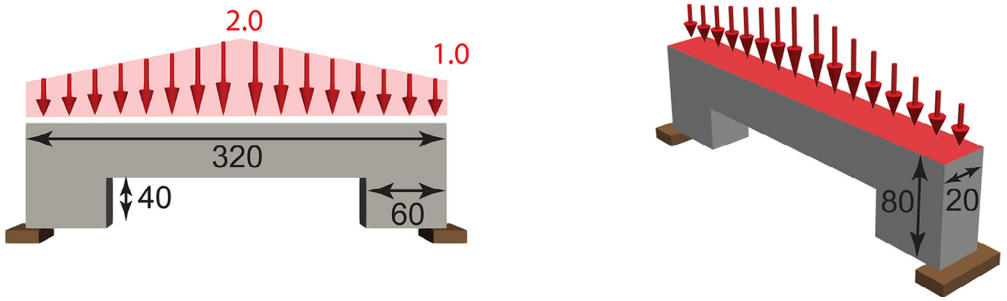

Fig. 14. Bridge domain with the appropriate dimensions (dimensionless units) and boundary conditions.

6.4. Computational efficiency

This subsection formally compares the computational time of our proposed ML-driven framework with different block sizes to that of the standard topology optimization (which solves the fine-resolution FEA at each iteration). For this comparison, we use the MBB beam problem with the different mesh sizes solved in the first example (i.e., 1 million, 10 million, and 35 million elements). Because of the hardware limitation (i.e. insufficient RAM memory of the GPU), we are unable to solve the standard topology optimization with the 35 million mesh. Thus, this subsection only presents the comparison for the 1-million-element, and 10-million-element meshes. Both of the meshes are solved using a machine with 24 Intel Xeon CPUs, 251 GB of RAM, and 2 (one for ML prediction and the other for performing the FEA) NVIDIA Titan Xp GPUs with 12 GB of RAM.

The breakdown of computational time and the corresponding speedup for various problem sizes and block sizes are displayed in Fig. 16. From Fig. 16, we observe that the speedup achieved by the proposed ML-driven framework increases with both problem size and block size. For the 10-million-element mesh and with $N_B = 3$, we are able to

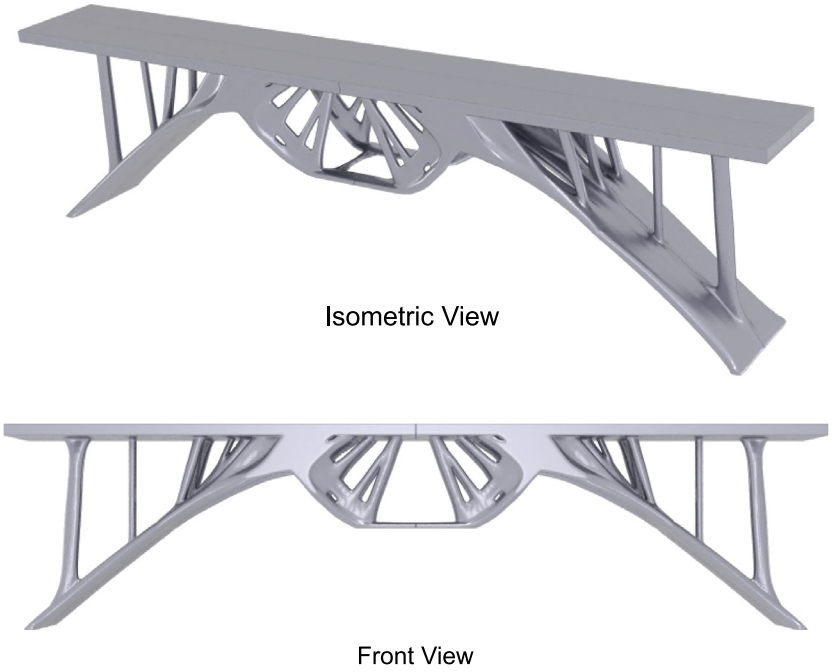

Fig. 15. The results obtained by the proposed framework for the bridge design with a $38,016,000$-element mesh.

reach up to a 26.5-time speedup compared to the standard topology optimization. Although we expect the speedup to be even larger for the 35-million-element mesh, we are not able to quantify the exact value because of our current computing hardware is unable to perform the standard topology optimization on a problem of this size. Nevertheless, being able to achieve close to 30 times of speedup in a problem with 10 million elements and design variable is a good indication of the efficiency and scalability of the proposed ML-driven topology optimization framework.

To understand how we could further improve the performance of the framework, we measure the time spent in each step of the topology optimization, both for the standard approach and for our ML framework. In particular, we break down the computational time into four segments: fine-mesh FEA, coarse-mesh FEA, ML prediction, and others (including filtering, optimization algorithm, material interpolation model). The segmentation of computational time is shown in Fig. 16. From the figure, we observe that, for the standard topology optimization, the solution of the fine-mesh FEA is responsible for over 90% of the total computational time. By contrast, the proposed ML-driven approach replaces the fine-mesh FEA with the solution of the coarse-mesh FEA and the ML prediction, both of which contribute to a drastic reduction in total computational time. In addition, the percentage of the computational time associated with the coarse-mesh FEA decreases as the block size increases. For block size $N_B = 3$, the coarse-mesh FEA only constitutes 17% of the total time, implying that further increasing block size could improve the computational time by at most 17% for this problem. Nonetheless, increasing the block size could lead to the solution of larger problems, and for such larger problems the improvement in efficiency for larger block sizes might be considerably better. A better way to further improve the efficiency of the proposed ML-driven framework is to improve the efficiency of the ML prediction (e.g. multi-GPUs setups, just-in-time compilation) and/or the other tasks related to the topology optimization (e.g., filtering, optimization algorithm, and material interpolation model), which are beyond the scope of this work.

7. Conclusion

We propose a ML-driven framework that is able to significantly accelerate (up to 26 times) traditional topology optimization without sacrificing accuracy. We demonstrate that the proposed framework can capture the underlying physics of the problem. As a result, we are able to separate the training of the ML model from its application in topology optimization. This offline strategy enables us to handle considerably larger design problems and achieve

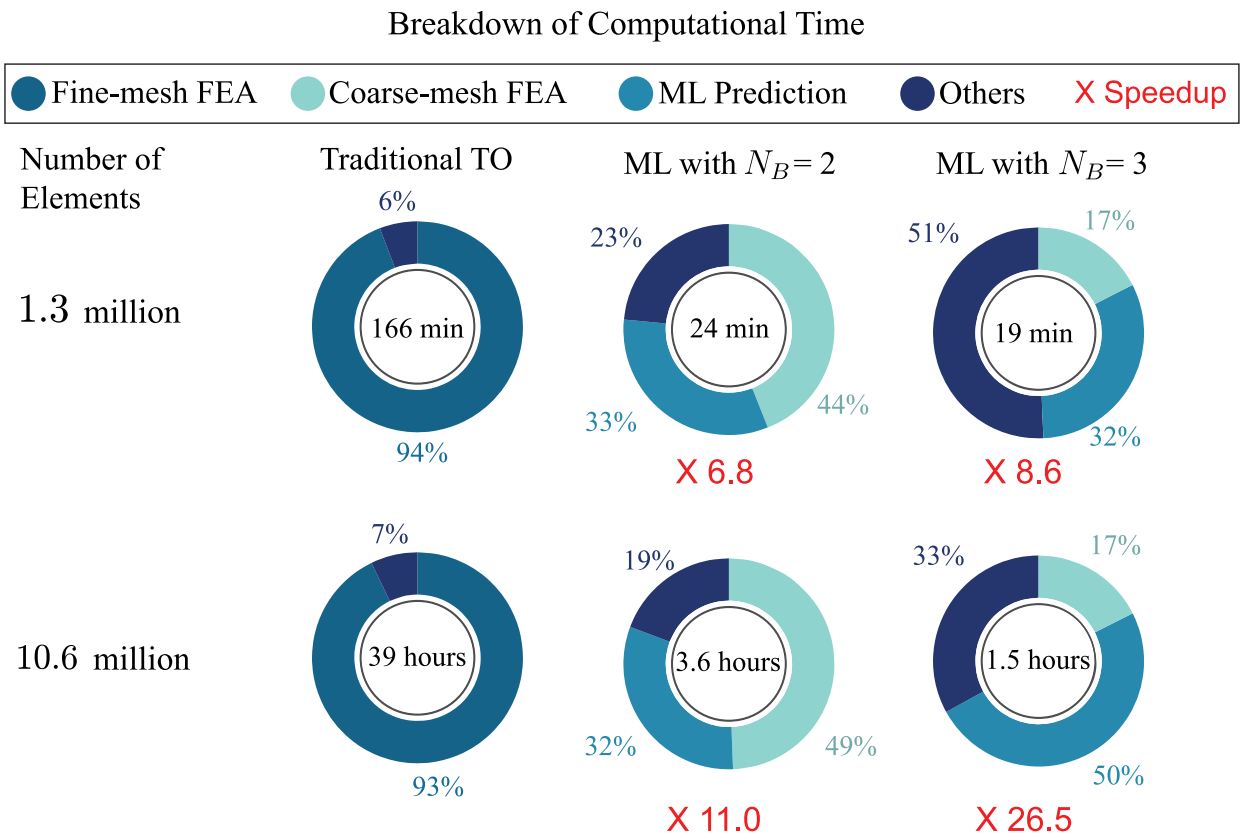

Fig. 16. A demonstration of the computational efficiency of the ML framework, with a breakdown of the computational time by the fine mesh FEA, coarse mesh FEA, ML prediction, and other tasks related to the topology optimization (filtering, optimization algorithm, and material interpolation model).

significantly higher speedup as compared to our prior work [1]. We also demonstrate that the trained ML model is generalizable and can accurately solve design problems that are not present in the training set. To provide a theoretical basis to our physics-based learning framework, we perform a mathematical analysis based on continuum mechanics theory to explain and quantify the generalization capability of the ML model.

To learn the underlying physics in a scalable manner, we adopt a two-resolution topology optimization setup, in which a fine-resolution mesh is embedded in a coarse-resolution mesh. The fine-resolution mesh contains the geometric and topological information of the design, and the coarse-resolution mesh provides the physical information. The physical information coming from the coarse-resolution mesh is extracted through a FEA, which is computationally efficient because of the moderate number of elements in the coarse mesh. Together, the physical information from the coarse-resolution mesh and material stiffness distribution from the fine-resolution mesh serve as the inputs to the ML model to predict the sensitivity information. The predicted sensitivity is then used to update the design variables on the fine-resolution mesh. In traditional topology optimization, the computation of the sensitivity requires a FEA on the fine-resolution mesh at each optimization step, whose computational time dominates the overall computational cost. Our proposed framework completely eliminates the need of performing a FEA on the fine-resolution mesh, and therefore drastically reduces the overall computational time. In addition, only performing FEA on the coarse-resolution mesh also allows the proposed ML-driven framework to solve considerably larger design problems using our current available hardware (which has limited RAM memory).

On the ML side, we use a state-of-the-art CNN architecture, which is inspired by deep learning models used in image processing tasks. Similar to those models used in image processing, we use convolutional layers to extract information from the stiffness map. The convolutional layers can learn to detect relevant features of the stiffness map, which, together with the physical information from the coarse-resolution mesh, are treated as input to a set

of fully-connected layers to predict the sensitivity. Furthermore, we add residual links to the fully-connected layers and use batch normalization between the layers, both of which promote more effective training. We show that the resulting ML model is accurate and efficient.

To further understand the behavior of the ML model, we investigate the influence of the training set on the performance of the trained ML model. We discover that this performance is heavily influenced by the type of problems collected in the training set. We further conclude that at least two distinct problem types, bending and torsion, need to be included in the training set to obtain a generalizable model. Based on this insight, we propose a measure to classify the topology optimization problems accordingly. We recognize that other types of problems could also influence the performance of the ML model, and we believe that this is an interesting topic for future investigation.

We present several design examples to demonstrate the capabilities of the proposed ML framework. These design examples vary in domain size, boundary conditions, mesh size, volume fraction, and filter radius. We demonstrate that the proposed ML framework, once trained, is robust to variations in all the above-mentioned design parameters. Furthermore, we present a study on the computational efficiency of the proposed framework that showcases its potential in not only accelerating the topology optimization process but also enabling the ability to solve large-scale design problems under limited hardware requirements.

Future research could possibly extend the proposed ML framework to other objective functions, constraints and physics, which would be especially advantageous for non-linear state equations, such as Navier–Stokes and hyper-elastic materials, because of the computational cost associated with such state equations. However, such extension does not come without its challenges, and more complex problems will probably require more sophisticated ML models (e.g. more nodes in the DNN model), more data in the training set, and some new ideas. Moreover, the integration of the proposed ML approach with other large-scale techniques, such as reduced order models and/or parallel computing approaches, has the potential to further increase efficiency and lead to effective solutions of even larger problems.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Fernando V. Senhora reports that financial support was provided by Siemens Corporation, Technology.

Acknowledgments

The authors acknowledge the financial support from “Siemens Corporation, Technology, USA” under the project titled “Deep Learning Enhanced Topology Optimization”, and through a Doctoral Fellowship to FVS. We also acknowledge support from the US National Science Foundation (NSF) under grant # 2105811.

Appendix A. Nomenclature

Machine learning

$\gamma,\ \xi$	Linear shift parameter of batch normalization
$\mu(.)$	Mean function
$\sigma(.)$	Standard deviation function
$\sigma_1,\ \sigma_2,\ \sigma_3$	Principal stresses
θ	Measure the angle that the vector v makes with (1/2,1/2,0)
CL	Convolutional layer
$d_1,\ d_2,\ d_3$	Diameter of the Mohr's circles
ML	Machine Learning
N_B	Block size (ratio of the fine mesh element side to the coarse mesh element side)
DNN	Deep Neural Network
CNN	Convolutional Neural Network
ResNet	Residual network
$\overline{\mathbf{u}}$	Standardized displacements
$\mathbf{w}$	Vector composed of the diameters of the Mohr's circles

Topology optimization

β	Parameter that controls the sharpness of the Heaviside projection function
ϵ	Ersatz stiffness
η	Threshold value of the Heaviside projection function
ω_e	Set of the elements of the fine mesh that lie inside the coarse element e
$\mathop{\mathbb{A}}_{i=1}^{M}$	Assembly information of the global stiffness matrix
C	Compliance
C_C	Compliance of the coarse mesh
C_F	Compliance of the fine mesh
$(E_C)_e$	Stiffness of eth element of the coarse mesh
E_i	Interpolated stiffness of the ith element of the fine mesh
$\mathbf{f}$	Load vector
FEA	Finite Element Analysis
g_V	Volume constraint
$\mathcal{H}$	Heaviside projection function
$\mathbf{K}$	Stiffness matrix
$\mathbf{k}_0$	The local stiffness matrix of a solid element
$\mathbf{k}_i$	The ith element local stiffness matrix
$\mathbf{N}$	Matrix that interpolates the coarse load vector to the fine load vector
$\mathbf{P}$	Filter matrix
p	Penalization parameter of the SIMP formulation
R	The radius of the filter that controls the minimum length scale
SIMP	Solid Isotropic Material with Penalization
tanh	Hyperbolic tangent function
$\mathbf{u}$	Displacement vector
$\mathbf{v}$	Vector that contains the volume of each element
V_{max}	Maximum allowable volume of material for the volume constraint
$\mathbf{x}_i^\star,\ \mathbf{x}_j^\star$	Position of the centroid of the ith and jth elements, respectively
$\mathbf{z}$	Vector of design variables
$\overline{\mathbf{z}}$	Vector of filtered densities
z_i	The ith component of the vector of design variables

Appendix B. Training loss data of the ML models

Fig. B.17 displays the training and validation loss data of the ML models for block size 2 and 3.

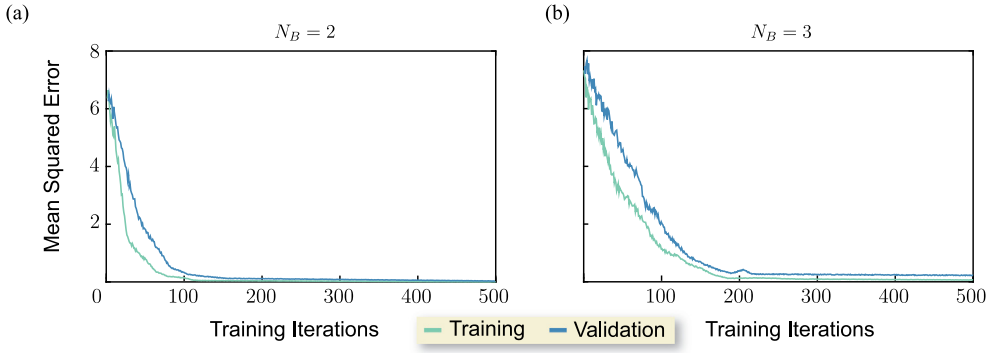

Fig. B.17. Training and validation loss data of the ML models for (a) block size 2, and (b) block size 3.

References

[1] H. Chi, Y. Zhang, T.L.E. Tang, L. Mirabella, L. Dalloro, L. Song, G.H. Paulino, Universal machine learning for topology optimization, Comput. Methods Appl. Mech. Engrg. 375 (2021) 112739.
[2] P. Christensen, A. Klarbring, An introduction to structural optimization, in: Solid Mechanics and Its Applications, Springer Netherlands, 2008.
[3] M. Bendsøe, O. Sigmund, Topology Optimization: Theory, Methods and Applications, Springer Berlin Heidelberg, 2003, http://dx.doi.org/10.1007/978-3-662-05086-6.
[4] O. Sigmund, A 99 line topology optimization code written in Matlab, Struct. Multidiscip. Optim. 21 (2001) 120–127, http://dx.doi.org/10.1007/s001580050176.
[5] E. Andreassen, A. Clausen, M. Schevenels, B.S. Lazarov, O. Sigmund, Efficient topology optimization in Matlab using 88 lines of code, Struct. Multidiscip. Optim. 43 (1) (2011) 1–16.
[6] C. Talischi, G. Paulino, A. Pereira, I.M. Menezes, PolyTop: A Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes, Struct. Multidiscip. Optim. 45 (3) (2012) 329–357.
[7] T. Borrvall, J. Petersson, Large-scale topology optimization in 3D using parallel computing, Comput. Methods Appl. Mech. Engrg. 190 (46) (2001) 6201–6229, http://dx.doi.org/10.1016/S0045-7825(01)00216-X.
[8] S. Schmidt, V. Schulz, A 2589 line topology optimization code written for the graphics card, Comput. Vis. Sci. 14 (6) (2011) 249–256, http://dx.doi.org/10.1007/s00791-012-0180-1.
[9] N. Aage, B.S. Lazarov, Parallel framework for topology optimization using the method of moving asymptotes, Struct. Multidiscip. Optim. 47 (4) (2013) 493–505, http://dx.doi.org/10.1007/s00158-012-0869-2.
[10] N. Aage, E. Andreassen, B.S. Lazarov, Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework, Struct. Multidiscip. Optim. 51 (3) (2015) 565–572, http://dx.doi.org/10.1007/s00158-014-1157-0.
[11] R. Labanda, Mathematical programming methods for large-scale topology optimization problems (Ph.D. Thesis), 2015, p. 240.
[12] N. Aage, E. Andreassen, B.S. Lazarov, O. Sigmund, Giga-voxel computational morphogenesis for structural design, Nature 550 (7674) (2017) 84–86.
[13] S. Wang, E.d. Sturler, G.H. Paulino, Large-scale topology optimization using preconditioned Krylov subspace methods with recycling, Internat. J. Numer. Methods Engrg. 69 (12) (2007) 2441–2468.
[14] O. Amir, N. Aage, B. Lazarov, On multigrid-CG for efficient topology optimization, Struct. Multidiscip. Optim. 48 (2014) 815–829, http://dx.doi.org/10.1007/s00158-013-1015-5.
[15] Y.Y. Kim, G.H. Yoon, Multi-resolution multi-scale topology optimization — A new paradigm, Int. J. Solids Struct. 37 (39) (2000) 5529–5559, http://dx.doi.org/10.1016/S0020-7683(99)00251-6.
[16] T.H. Nguyen, G.H. Paulino, J. Song, C.H. Le, A computational paradigm for multiresolution topology optimization (MTOP), Struct. Multidiscip. Optim. 41 (4) (2010) 525–539.
[17] H. Liu, Y. Wang, H. Zong, M.Y. Wang, Efficient structure topology optimization by using the multiscale finite element method, Struct. Multidiscip. Optim. 58 (4) (2018) 1411–1430, http://dx.doi.org/10.1007/s00158-018-1972-9.
[18] F.V. Senhora, E.D. Sanders, G.H. Paulino, Optimally-tailored spinodal architected materials for multiscale design and manufacturing, Advanced Materials (2022) 2109304.
[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[20] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734, http://dx.doi.org/10.3115/v1/D14-1179.

[21] B.M. Henrique, V.A. Sobreiro, H. Kimura, Literature review: Machine learning techniques applied to financial market prediction, Expert Syst. Appl. 124 (2019) 226–251, http://dx.doi.org/10.1016/j.eswa.2019.01.012.
[22] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533, http://dx.doi.org/10.1038/nature14236.
[23] F.E. Bock, R.C. Aydin, C.J. Cyron, N. Huber, S.R. Kalidindi, B. Klusemann, A review of the application of machine learning and data mining approaches in continuum materials mechanics, Front. Mater. 6 (2019) 110, http://dx.doi.org/10.3389/fmats.2019.00110.
[24] E. Ulu, R. Zhang, M.E. Yumer, L.B. Kara, A data-driven investigation and estimation of optimal topologies under variable loading configurations, in: Y.J. Zhang, J.M.R.S. Tavares (Eds.), Computational Modeling of Objects Presented in Images. Fundamentals, Methods, and Applications, Springer International Publishing, Cham, 2014, pp. 387–399.
[25] I. Sosnovik, I.V. Oseledets, Neural networks for topology optimization, 2017, arXiv:1709.09578,
[26] S. Banga, H. Gehani, S. Bhilare, S. Patel, L. Kara, 3D Topology optimization using convolutional neural networks, 2018, ArXiv arXiv:1808.07440.
[27] N.A. Kallioras, G. Kazakis, N.D. Lagaros, Accelerated topology optimization by means of deep learning, Struct. Multidiscip. Optim. 62 (3) (2020) 1185–1212.
[28] V. Keshavarzzadeh, M. Alirezaei, T. Tasdizen, R.M. Kirby, Image-based multiresolution topology optimization using deep disjunctive normal shape model, Comput. Aided Des. 130 (2021) 102947.
[29] Y. Yu, T. Hur, J. Jung, I.G. Jang, Deep learning for determining a near-optimal topological design without any iteration, Struct. Multidiscip. Optim. 59 (3) (2019) 787–799, http://dx.doi.org/10.1007/s00158-018-2101-5.
[30] Z. Nie, T. Lin, H. Jiang, L.B. Kara, Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain, J. Mech. Des. 143 (3) (2021) 031715.
[31] B. Li, C. Huang, X. Li, S. Zheng, J. Hong, Non-iterative structural topology optimization using deep learning, Comput. Aided Des. 115 (2019) 172–180, http://dx.doi.org/10.1016/j.cad.2019.05.038.
[32] Y. Zhang, A. Chen, B. Peng, X. Zhou, D. Wang, A deep convolutional neural network for topology optimization with strong generalization ability, 2019, arXiv:1901.07761.
[33] S. Rawat, M.H. Shen, Application of adversarial networks for 3D structural topology optimization, in: WCX SAE World Congress Experience, SAE International, 2019, http://dx.doi.org/10.4271/2019-01-0829.
[34] D.W. Abueidda, S. Koric, N.A. Sobh, Topology optimization of 2D structures with nonlinearities using deep learning, Comput. Struct. 237 (2020) 106283.
[35] M. Bendsøe, Optimal shape design as a material distribution problem, Struct. Optim. 1 (1989) 193–202.
[36] M. Bendsøe, O. Sigmund, Material interpolation schemes in topology optimization, Arch. Appl. Mech. 69 (9–10) (1999) 635–654.
[37] F. Wang, B. Lazarov, O. Sigmund, On projection methods, convergence and robust formulations in topology optimization, Struct. Multidiscip. Optim. 43 (6) (2011) 767–784.
[38] B. Bourdin, Filters in topology optimization, Internat. J. Numer. Methods Engrg. 50 (9) (2001) 2143–2158.
[39] T. Zegard, G. Paulino, Bridging topology optimization and additive manufacturing, Struct. Multidiscip. Optim. 53 (1) (2016) 175–192.
[40] C. Talischi, G. Paulino, An operator splitting algorithm for Tikhonov-regularized topology optimization, Comput. Methods Appl. Mech. Engrg. 253 (2013) 599–608.
[41] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[42] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[43] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, R. Horaud, A comprehensive analysis of deep regression, 2018, ArXiv.
[44] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, ArXiv arXiv:1502.03167.
[45] R.v. Mises, On Saint Venant's principle, Bull. Amer. Math. Soc. 51 (8) (1945) 555–562.
[46] S. Timoshenko, History of Strength of Materials: With a Brief Account of the History of Theory of Elasticity and Theory of Structures, Courier Corporation, 1983.
[47] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, pp. 8024–8035.
[49] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.