Computer-Aided Design 130 (2021) 102947

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Image-Based Multiresolution Topology Optimization Using Deep Disjunctive Normal Shape Model

Vahid Keshavarzzadeh *[,1], Mitra Alirezaei [1], Tolga Tasdizen, Robert M. Kirby

Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, 84112, USA

ARTICLE INFO

Article history:
Received 23 September 2019
Received in revised form 27 July 2020
Accepted 20 September 2020

Keywords:
Topology optimization
Multiresolution analysis
Image-based segmentation
Deep neural networks

ABSTRACT

We present a machine learning framework for predicting the optimized structural topology designs using multiresolution data. Our approach primarily uses optimized designs from inexpensive coarse mesh finite element simulations for model training and generates high resolution images associated with simulation parameters that are not previously used. Our cost-efficient approach enables the designers to effectively search through possible candidate designs in situations where the design requirements rapidly change. The underlying neural network framework is based on a deep disjunctive normal shape model (DDNSM) which learns the mapping between the simulation parameters and segments of multi resolution images. Using this image-based analysis we provide a practical algorithm which enhances the predictability of the learning machine by determining a limited number of important parametric samples (i.e. samples of the simulation parameters) on which the high resolution training data is generated. We demonstrate our approach on benchmark compliance minimization problems including the 3D topology optimization where we show that the high-fidelity designs from the learning machine are close to optimal designs and can be used as effective initial guesses for the large-scale optimization problem.

© 2020 Elsevier Ltd. All rights reserved.

1. Introduction

1.1. Background

Topology optimization has been extensively used as a popular computational design tool for generating optimal structural layouts. This powerful tool enables the creation of innovative structures with complex geometry that may not be realizable with traditional approaches such as shape or size optimization [1,2]. This approach however suffers from significant computational cost especially when a detailed structural design is sought. In such cases, typically, many large scale finite element analyses need to be performed to reach to a satisfactory design which poses a significant challenge in terms of computational cost. Finding such high resolution designs in scenarios where the simulation parameters are variable is even more onerous.

In this paper, we adopt a scientific visualization/machine learning (ML) approach to overcome the computational challenges for large scale topology optimization, especially when optimal high-fidelity designs are required for variable simulation parameters. In particular, our approach utilizes multifidelity designs where mainly inexpensive low-fidelity designs in addition to judiciously chosen high-fidelity designs are used for training. Using the learning machine which encodes the relationship between the simulation parameters and the shape space we approximate optimized designs on parameters that are not previously considered in the simulations. This approach will enable the designers to efficiently traverse the simulation parameters and find the best solutions which satisfy the design objectives and criteria.

The result of this work contributes to the development of an efficient visualization tool for exploring optimized designs with variable parameters. Ideally the visualization tool is a platform that provides high resolution optimized designs interactively and this work is one stepping stone toward this objective. With such a tool, a designer could navigate through the design space interactively, assessing multiple designs qualitatively (i.e. visually) and quantitatively (through various evaluation metrics build into the tool).

In the context of engineering design, similar efforts have been devoted to embed the high-dimensional design space into semantic compact subspaces [3,4] using a manifold clustering procedure and deep generative models. The idea of visual parameter space analysis has also been explored [5,6] where visual interactive tools are introduced to navigate the data spaces or the space of quantities of interest as a function of reduced-dimensional

* Corresponding author.
E-mail addresses: vkeshava@sci.utah.edu (V. Keshavarzzadeh), mitra@sci.utah.edu (M. Alirezaei), tolga@sci.utah.edu (T. Tasdizen), kirby@sci.utah.edu (R.M. Kirby).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.cad.2020.102947
0010-4485/© 2020 Elsevier Ltd. All rights reserved.

parameter space. Similar to these efforts, several researchers have introduced visual analysis techniques for exploration of parameter-data space map in multi-criteria decision making [7], medical image analysis [8,9], geoscientific simulation models [10], decision support in flood management [11,12], inspection of fastening bolts on freight trains [13], etc. Our work in this paper can be viewed as a visual analysis technique which adopts a powerful function approximator, i.e. deep neural network to map the simulation parameters to the high-dimensional space of topology optimized designs.

Deep learning (DL) is known as a subset of ML. A large number of layers in the architecture of DL models enable them to estimate complex functions. However, there is a need for large datasets to train such complex models. In other words, DL fundamentally is neural networks that learn from large amounts of data with the use of various layers. An important class of DL, convolutional neural networks introduced in [14], are widely used in image processing. DL has also been employed in many different areas such as video processing, speech recognition, natural language processing, and computer vision [15–18].

In this paper, we present a novel deep learning approach in conjunction with the disjunctive normal shape model (DNSM) for the generic topology optimization problem. We refer to this approach as Deep DNSM which is abbreviated as DDNSM. The DNSM is a shape model that first was introduced by Ramesh et al. [19] and reformulated by Mesadi et al. [20] to be used in a Bayesian framework for image segmentation tasks. The DNSM has also been used in conjunction with a convolutional neural network by Javanmardi et al. [21] for segmenting shapes with low quality and low signal-to-noise ratio. The main idea of DNSM is to model shapes as the disjunction of convex polytopes with each polytope as a conjunction of half-spaces. Inspired by [21] we have designed a new framework that uses the DNSM on top of a fully connected deep neural network as a multiresolution network with both low resolution and high resolution designs to be trained jointly. We provide more details on both DNSM and DDNSM in Section 3.

1.2. Related work

Structural topology optimization has evolved significantly since the time it first introduced. A comprehensive review of current and future trends in this field has recently been published [22]. Among these works many researchers have addressed the problem of parametric topology optimization and topology optimization under uncertainty where the main focus is to generate robust and reliable designs [23–28]. Such problems pose a similar challenge to the one in this paper where the simulation parameters are variable. In the same vein, several research works have also attempted to tackle the problem of data-driven topology optimization and proposed novel approaches for prediction improvement in such data-driven approximations [29,30]. A number of researchers have proposed to use computational capabilities of a Graphics Processing Units (GPU) for large scale topology optimization where the main goal is to achieve optimal high resolution structures with minimal process time [31–33]. These efforts are in line with approaches which use machine learning techniques for large scale problems where powerful computational resources are required.

Machine learning techniques have achieved compelling results in data-based approximations such as image processing, pattern recognition, finance, etc. [34–36]. Complementing these efforts, novel techniques have been proposed for prediction/approximation with physics-based or simulation-based learning machines. The main goal in these works is to predict physics-based simulations which are typically governed by partial differential equations (PDE) [37,38]. This type of approaches has close connection to the topology optimization problem where the solution of the optimization problem is sought for a physical systems described with PDEs.

Topology optimization using machine learning is a relatively new research direction. As examples, authors in [39–41] use generative modeling techniques for topology optimization where they use variational auto encoders (VAE) and generative adversarial networks (GANs) to predict the optimized topology designs. A similar work [30] proposes the use of Principal Component Analysis (PCA) and a fully connected neural network to learn the mapping between loading configurations and optimal topologies. Authors in [42] use convolutional encoder–decoder architecture to solve the topology optimization problem by posing it as an image segmentation task. Finally authors in [43] introduce a *theory-driven* learning mechanism based on the optimality criteria in topology optimization for generation of near-optimal topology designs.

It is also noteworthy to mention approaches based on multiresolution analysis in the context of shape and topology optimization [44–52]. The main theme among these works is to leverage the computational efficiency of low resolution models to obtain more expensive high resolution designs. Our multiresolution approach in this paper is indeed similarly motivated by reducing the computational cost; however unlike the aforementioned references which consider a deterministic setting, our strategy is devised to incorporate parametric variations in design such as variations in load and boundary conditions [28,53]. Another important line of research which is relevant to this paper is the incorporation/representation of geometric features in topology optimization which is done via combining free-form topology optimization with embedded components/holes or representing the structure via union of geometric primitives, e.g. rectangles with straight or semicircular ends [54–69]. We however remark that our approach in this paper, DDNSM which is an effective way to represent binary images is solely used to learn from and subsequently approximate the topology optimized designs as binary images. We note that we do not use this approach for the task of topology optimization (or representing the geometry) itself; we instead use a well-known density-based approach to generate the optimized designs [70].

In this paper, we mainly focus on the computational challenges associated with the high resolution designs with variable parameters. To this end we propose an algorithm which utilizes a limited number of high resolution images corresponding to previously found optimized designs in conjunction with a large number of inexpensive low resolution optimized topology images for network training and generate a learning machine which predicts the near-optimal high resolution images for previously unseen simulation parameters. Our work contributes to the existing literature on topology optimization using machine learning by providing a novel approach which connects topology optimization with scientific visualization. In particular our work can be considered among the few recent studies which leverage deep learning for predicting near optimal designs [41,43]. These approaches are particularly useful when there is variation in design parameters and could be effective for fast exploration of optimal design manifold. We remark that in this paper we provide a systematic computational cost analysis which serves as a practical guideline to assess the learning cost in comparison with the cost of direct high resolution designs. We also remark that our deep learning approach is, in essence, different from the zero-order optimization approaches (or population-based approaches) such as genetic programming. Our approach similarly to any deep learning approach is based on the minimization of a loss function which is performed via a gradient-descent approach. As mentioned earlier we also use a gradient-based approach to

generate the training data which renders our approach entirely gradient-based [71].

The organization of this paper is as follows. In Section 2 we briefly discuss the topology optimization problem, in particular the compliance minimization. We also discuss the details of data generation for network training in this section. Section 3 discusses the details of DNSM and DDNSM approach in conjunction with an algorithm for selection of important high resolution samples. Section 4 presents numerical results on topology design of an L-bracket elastic structure, a heat sink structure, and a 3D linear elastic structure. Finally Section 5 discusses the concluding remarks.

2. Problem statement

2.1. Notation

The following notations are frequently used in this article:

- Bold faced characters denote vectors and matrices e.g. $\boldsymbol{U}$ denotes the vector of displacements.
- We use superscript to denote the resolution e.g. $\mathcal{S}^L$ denotes low resolution shape, and use subscripts to denote parametric samples e.g. $\mathcal{S}_i^L$ is a low resolution shape with sample index i. We also use subscripts to denote the dimension in the parameter space e.g. p_1 is the first parametric variable. Different uses of subscripts are clear from the context throughout the text.
- Structural topology optimization typically focuses on the response of linear elastic structures which is characterized by the following governing equation:

$$\begin{cases} \nabla \cdot \sigma(\boldsymbol{x}) + b(\boldsymbol{x}) = 0 & \forall \boldsymbol{x} \in D \\ \sigma(\boldsymbol{x}) \cdot \boldsymbol{n} = n(\boldsymbol{x}) & \forall \boldsymbol{x} \in \Gamma_N \\ u(\boldsymbol{x}) = 0 & \forall \boldsymbol{x} \in \Gamma_D \end{cases} \tag{1}$$

  where ∇ is the gradient operator, σ is the Cauchy stress tensor, $b(\boldsymbol{x})$ is the body force, $n(\boldsymbol{x})$ is the tractive force and the physical domain $D \subset \mathbb{R}^d$, $d = 2, 3$ is a bounded and Lipschitz continuous domain with two sets of Dirichlet Γ_D and Neumann Γ_N boundary conditions where $\Gamma_N \cap \Gamma_D = \emptyset$.
- The above governing equations are presented in the form of a linear elliptic partial differential equation

$$\begin{cases} -\nabla.(\mathbb{C}(\boldsymbol{x})\nabla \boldsymbol{u}(\boldsymbol{x})) = f(\boldsymbol{x}) & \forall \boldsymbol{x} \in D \\ \boldsymbol{u}(\boldsymbol{x}) = 0 & \forall \boldsymbol{x} \in \partial D \end{cases} \tag{2}$$

  where $\mathbb{C}$ is the elasticity matrix. The spatial domain is denoted by D, and ∂D denotes the Dirichlet boundary condition. We generate data for the training purpose by parameterizing e.g. the force function $f(\boldsymbol{x})$ or boundary conditions Γ_D in our examples i.e. $f(\boldsymbol{x}) \to f(\boldsymbol{x}, \boldsymbol{p})$, or $u(\boldsymbol{x}) \to u(\boldsymbol{x}, \boldsymbol{p})$ $\forall \boldsymbol{x} \in \Gamma_D$ where $\boldsymbol{p}$ denotes the parameter.
- We demonstrate our approach on three examples: design of a linear elastic structure in 2D and 3D and design of a heat sink. All examples involve solving a similar linear elliptic partial differential equation mentioned above. We solve a deterministic PDE-constrained optimization for generating each image. We use the multiresolution images corresponding to different mesh sizes to predict the highest resolution designs.
- The multiresolution analysis is indeed a biresolution procedure in this paper. In most examples we have three resolutions i.e. high, medium and low denoted by H, M and L; however the procedure and the way the loss function in deep learning is computed entail two resolutions, i.e. we consider training with either H + M dataset or H + L dataset.

2.2. Compliance minimization

A standard topology optimization problem is a constrained optimization problem which minimizes compliance subject to a volume constraint. The computational complexity lies in (1) finding the compliance which typically requires a finite element analysis and (2) the number of optimization iterations which is normally larger for larger scale problems i.e. high resolution designs. We consider density based topology optimization in which the design space is characterized with element volume fractions. The optimization problem after finite element discretization reads

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & C(\boldsymbol{\rho}) = \boldsymbol{U}^T \boldsymbol{F} \\ \text{subject to} \quad & V(\boldsymbol{\rho}) \le \bar{V} \\ & \boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{U}(\boldsymbol{\rho}) = \boldsymbol{F}(\boldsymbol{\rho}) \\ & \rho_{min} \le \boldsymbol{\rho} \le 1, \end{aligned}$$

where $\boldsymbol{K}$, $\boldsymbol{U}$ and $\boldsymbol{F}$ are the global finite element stiffness matrix, the displacement vector, and the force vector, respectively; $\boldsymbol{\rho}$ is the vector of element volume fractions, C is the compliance, V is the volume, and $0 < \rho_{min} \ll 1$ is the lower bound for the volume fractions.

In this work we only use the filter to impose a minimum length scale. The filtered volume fractions $\hat{\rho}$ are expressed via the cone kernel K_F,

$$\hat{\rho}(x_i) = \frac{\sum_{j=1}^n K_F(x_i, x_j)\rho(x_j)}{\sum_{j=1}^n K_F(x_i, x_j)}, \qquad i = 1, \ldots, n, \tag{3}$$

where

$$K_F(x_i, x_j) = \begin{cases} r_{min} - |x_i - x_j| & \text{if } |x_i - x_j| \le r_{min} \\ 0 & \text{if } |x_i - x_j| > r_{min}. \end{cases} \tag{4}$$

In these expressions r_{min} and x_i denote the filter radius and the element i centroid [72].

We use the Solid Isotropic Material with Penalization (SIMP) method to penalize intermediate volume fractions [73,74]. To this end, we compute the global stiffness matrix $\boldsymbol{K}$ by using the processed (filtered) volume fractions $\hat{\rho}$ as

$$\boldsymbol{K} = \sum_{i=1}^n \hat{\rho}_i^{\iota} \boldsymbol{K}_i, \tag{5}$$

where n is the number of elements, $\iota = 3$ is the penalization parameter and $\boldsymbol{K}_i$ is the nominal element i stiffness matrix.

In our first numerical experiment, we consider the loading as the parameter space i.e. we consider variation in these quantities in our deterministic topology optimization statement. To this end, we introduce the parameter $\boldsymbol{p}$ in loading $\boldsymbol{F}(\boldsymbol{p}), \boldsymbol{p} \in \boldsymbol{P}$ where the parameters are treated as random variables. In this example the parameters are the location of the point load p_1 and the angle of the point load p_2. To realize these variables we use the random number generator from MATLAB that we describe in the next section. We then generate a number of optimized deterministic designs associated with parametric values. We use the well-known 88-line MATLAB code for generating optimized designs [70]. In the second numerical example we consider variation in the Dirichlet boundary condition which is parameterized with two variables (associated with the coordinates of the heat sink) similarly to the first example.

2.3. Data generation

We generate three sets of data associated with the low L, medium M and high H resolutions. We use the standard square finite elements in this work and we consider the lower resolution

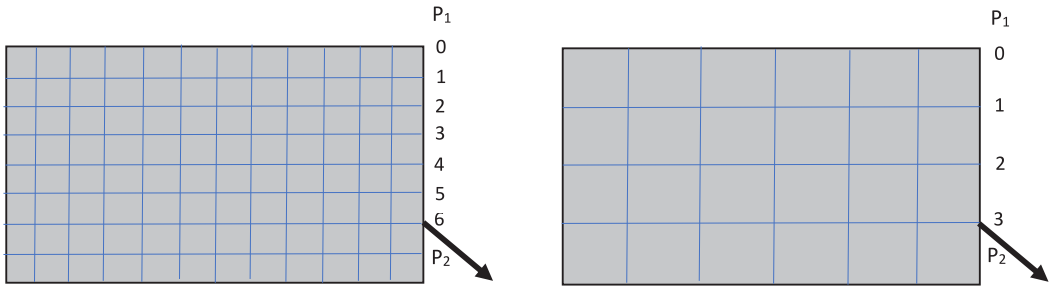

Fig. 1. Parameterized force on multiresolution finite element models.

size of the elements i.e. L and M to be 4 and 2 times of the high resolution H structure. The parameterized force $\boldsymbol{F}(\boldsymbol{p})$ for an example of medium and low resolution structures is shown in Fig. 1.

As shown in this schematic picture the load location in the high resolution structure can be simply determined from the lower resolution structure. For example in this case where the size of the coarse mesh is the double of the fine mesh, the load location in the high resolution model is the double of the low resolution model i.e. $p_1^H = 2p_1^M$. We note that the load angle remains the same between two models and we use individual fixed values for the filter radius in two structures. In one of our numerical examples where we consider the heat sink design, the parameter is the location of the heat sink. We use the same strategy as shown in Fig. 1 to determine the location of the heat sink for different resolutions. We also note that we generate optimized designs for different resolutions independently i.e. we do not use any information from each model to inform the other model during data generation.

The following steps summarize data generation for machine learning training:

- Generate three mesh sizes for low, medium and high resolutions with e.g. the size of the low resolution mesh double of the medium resolution mesh and the medium resolution mesh double of the high resolution mesh.
- Generate random parameters for load location and angle for the low resolution model. The load location for higher resolutions is determined based on its size with respect to the low resolution model. For example for a mesh with half size of the low resolution the load location is doubled as shown in Fig. 1. Assign individual filter values for each resolution. For the heat sink design use the same strategy to determine the location of the heat sink in higher resolution designs.
- For each parameter perform compliance minimization for three resolutions and find the optimal design and optimal compliance.

Our main goal in this paper is to build a predictive model which approximates the high resolution optimized design given multiresolution training data. In other words our neural network generates optimized designs for unknown parameters by learning from the images associated with pre-optimized designs in different resolutions. In the next section, we discuss the details of neural network construction as well as an algorithm for choosing informative high resolution designs which enhances the predictability of the learning machine.

3. Deep disjunctive normal shape model

In this section, we explain the Disjunctive Normal Shape Model (DNSM) and its incorporation on top of a fully connected deep neural network, termed as Deep Disjunctive Normal Shape Model (DDNSM). We also discuss details of our importance sampling strategy in this section.

3.1. DNSM representation

The DNSM [19,21] represents shapes as disjunctions of $\mathcal{N}$ convex polytopes, with each polytope as a conjunction of $\mathcal{M}$ half-spaces. Consider the characteristic function of shapes to be $f : \mathbb{R}^d \to \mathcal{B}$ where $\mathcal{B} = \{0, 1\}$. The foreground of the shape is $\Omega_f = \{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}) = 1\}$ and can be approximated as union of $\mathcal{N}$ polytopes, i.e. $\bigcup_{i=1}^{\mathcal{N}} \mathcal{P}_i$ in which $\mathcal{P}_i$ represents the i'th polytope that can be obtained as an intersection of $\mathcal{M}$ half-spaces, $\bigcap_{j=1}^{\mathcal{M}} \mathcal{H}_{ij}$. Half-spaces are represented using the perceptron function:

$$\mathcal{H}_{ij} = \begin{cases} 1, & \text{if} \quad \sum_{k=1}^{d}(w_{ijk}x_k + b_{ij}) \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where w_{ijk}, b_{ij}, and d correspond to the weights, biases, and the input dimension. We denote $\mathcal{W}_{ij} = (w_{ijk}, b_{ij})$ as the coefficients of the DNSM model that are associated with the half-space $\mathcal{H}_{ij}$ and denote $\mathcal{W}$ as the entire set of coefficients for the DNSM model i.e. $\mathcal{W}_{ij} \in \mathcal{W}$ cf. Fig. 2.

In Boolean logic, any function can be represented as a disjunction of conjunctions, which is called disjunctive normal form [75]. The disjunctive normal form of Ω_f is

$$\tilde{f}(\mathbf{x}) = \vee_{i=1}^{\mathcal{N}}\big(\wedge_{j=1}^{\mathcal{M}}\mathcal{H}_{ij}(\mathbf{x})\big). \tag{7}$$

In order to convert the disjunctive normal form into a differentiable functional form, the conjunction of binary variables is first approximated as their products, i.e.

$$d_i(\mathbf{x}) = \wedge_{j=1}^{\mathcal{M}}\mathcal{H}_{ij}(\mathbf{x}) \sim \prod_{j=1}^{\mathcal{M}}\mathcal{H}_{ij}(\mathbf{x}). \tag{8}$$

Based on De Morgan's law, the disjunctions are expressed as negation of conjunctions i.e. $\vee_{i=1}^{\mathcal{N}} d_i(\mathbf{x}) \equiv \neg \wedge_{i=1}^{\mathcal{N}} \neg d_i(\mathbf{x})$. In the context of binary variables the negation is expressed as $\neg\Phi = 1 - \Phi$. Therefore $\vee_{i=1}^{\mathcal{N}} d_i(\mathbf{x}) = 1 - \prod_{j=1}^{\mathcal{N}}(1 - d_i(\mathbf{x}))$.

The binary perceptron $\mathcal{H}_{ij}$ is then relaxed using the logistic sigmoid function:

$$\hat{\mathcal{H}}_{ij}(\mathbf{x}) = \frac{1}{1 + \exp\big(\sum_{k=1}^{d}(w_{ijk}x_k + b_{ij})\big)}. \tag{9}$$

Subsequently, the final form of the approximated characteristic function reads

$$\tilde{f}(\mathbf{x}) = 1 - \prod_{i=1}^{\mathcal{N}}(1 - \prod_{j=1}^{\mathcal{M}}\hat{\mathcal{H}}_{ij}(\mathbf{x})) \tag{10}$$

which is a differentiable function and can be utilized in gradient based optimization algorithms.

Remark 3.1. We note that the above representation allows approximation of binary valued images which makes it suitable for approximating topology optimized designs. We also note that we

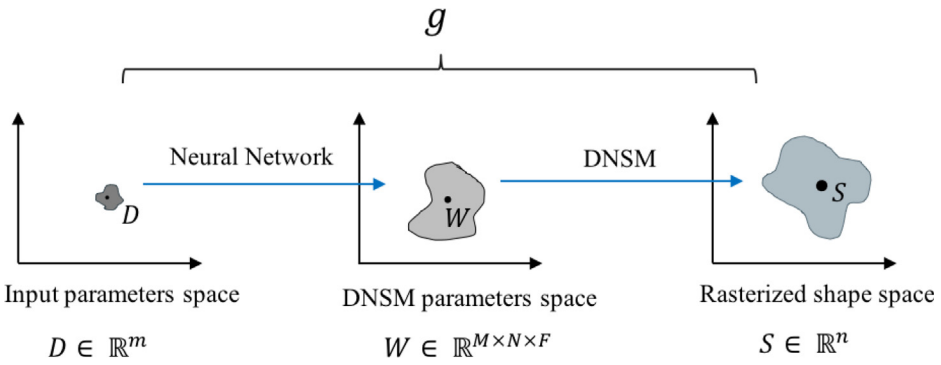

Fig. 2. Different spaces and their relationships to reconstruct a shape from the given input parameters $\mathcal{D}$. Here m corresponds to the input parameters dimension. $\mathcal{N}$ and $\mathcal{M}$ identify the number of polytopes and half-spaces, and $\mathcal{F}$ is the number of parameters for identifying a half-space in an arbitrary dimension. The (rasterized) shape $\mathcal{S}$ is represented with $n = H \times W$ pixels where H is the height, and W is the width of the shape.

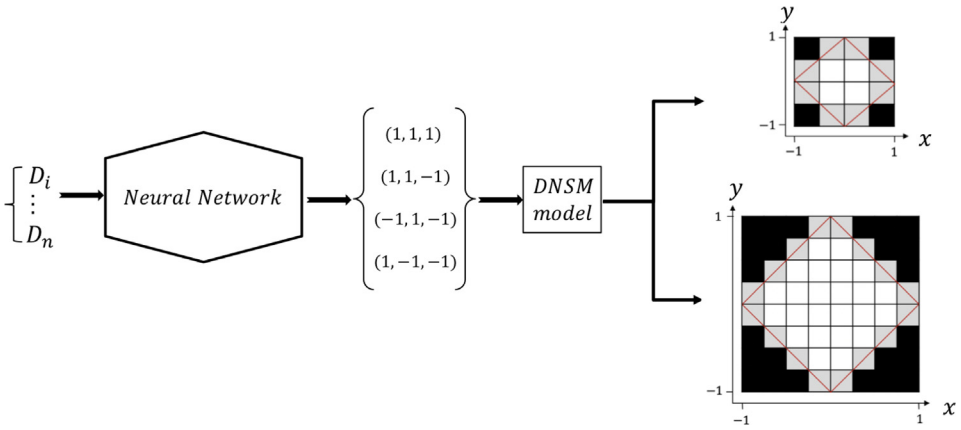

Fig. 3. Example of reconstructing a shape in two different resolutions using DNSM. In this example, $\mathcal{N} = 1$, $\mathcal{M} = 4$ where $\mathcal{N}$ is the number of polytopes (denoted by d_i cf. Eq. (8)) and $\mathcal{M}$ is the number of halfspaces (denoted by $\mathcal{H}_{ij}$ cf. Eq. (6)) . The neural network takes the design parameters, $\mathcal{D}_i$'s, as input and produces the coefficients of the half-spaces. Subsequently, DNSM takes the output of the neural network and reconstructs the shape in two resolutions. Black, white, and gray areas correspond to pixels with 0, 1, and between 0–1 values, respectively.

only use the DNSM model to represent the images, and this model is not used within the topology optimization (or the generation of datasets). The DNSM model however is used within a deep learning framework which is based on minimization of a loss function. This minimization is performed via a gradient-descent approach which requires the sensitivity of functions such as the one in Eq. (10). Notably the sensitivity computation for deep learning tasks is performed via automatic differentiation [76] which is embedded in the deep learning machinery.

3.2. DDNSM representation

The DDNSM structure is composed of the DNSM on top of a neural network as depicted in Fig. 3. Consider $g : \mathbb{R}^m \to \mathbb{R}^n$ where $n \gg m$ as a function that maps low-dimension parameters to a high-dimensional shape cf. Fig. 2.

This mapping can be done in two steps. The first step is mapping the given low-dimension parameters $\mathcal{D}$ to the coefficients $\mathcal{W}$ of the DNSM using a neural network. Given a pair of $(\mathcal{D}_i, \mathcal{S}_i)$ as a training data point, where $\mathcal{D}_i$ indicates the input parameters and $\mathcal{S}_i$ indicates the shape, the network takes $\mathcal{D}_i$ as input and produces coefficients of the DNSM as outputs ($\mathcal{W}$ in Fig. 2). The dimensionality of $\mathcal{W}$ is $\mathcal{N} \times \mathcal{M} \times \mathcal{F}$.[2] In the next step, the output of the network is passed through the DNSM and the shape can be approximated uniquely, i.e., $\tilde{\mathcal{F}}_i$. Since this method maps the representative parameters into weights of half-spaces, the model is capable of generating approximately piece-wise binary images corresponding to embedding of shapes into images.

3.3. Multiresolution network

In addition to the fundamental network previously discussed, a major advantage of the proposed method is reconstructing shapes at arbitrary resolutions. The neural network produces the half-space coefficients, and then the same shape can be reconstructed at arbitrary resolutions by the DNSM. Fig. 3 shows an example of reconstructing a shape at two different resolutions.

The network is trained in an end-to-end manner which means that the training procedure does not consider the DNSM coefficients as target; it instead uses the optimized topology shapes as target. In other words, the DNSM coefficients for topologically optimal shapes are computed within the DDNSM procedure (which can be construed as latent variables), and the shapes themselves are used for end-to-end learning. The mean squared error of the difference between the reconstructed and target shapes is used to train the network. Since the network produces outputs at two different resolutions, the loss functions corresponding to each

[2] $\mathcal{N}$ is the number of polytopes cf. Eq. (8), $\mathcal{M}$ is the number of half-spaces cf. Eq. (6), and $\mathcal{F}$ is number of parameters for identifying a half-space in an arbitrary dimension. $\mathcal{F}$ is equivalent to the number of coefficients in Eq. (6). For example, in our 2D problems for a half-space $\mathcal{H}_{ij}$ the coefficients are $(w_{ij1}, w_{ij2}, b_{ij})$ which are associated with $(x, y, 1)$; therefore $\mathcal{F} = 3$. Similarly for 3D problems $\mathcal{F} = 4$ since there are four coefficients associated with $(x, y, z, 1)$.

output are

$$l_l = \frac{1}{|\Omega_l|}\sum_{i\in\Omega_l}(\mathcal{S}_{\mathbf{i}}^{L} - \tilde{\mathcal{S}}_{\mathbf{i}}^{L})^2 \tag{11}$$

$$l_h = \frac{1}{|\Omega_h|}\sum_{i\in\Omega_h}(\mathcal{S}_{\mathbf{i}}^{H} - \tilde{\mathcal{S}}_{\mathbf{i}}^{H})^2 \tag{12}$$

where Ω_l and Ω_h are the sets of low resolution and high resolution training examples, respectively. $\mathcal{S}_i^L$ and $\tilde{\mathcal{S}}_i^L$ correspond to the low resolution target and reconstructed shapes, and $\mathcal{S}_i^H$ and $\tilde{\mathcal{S}}_i^H$ correspond to the high resolution target and reconstructed shapes. The combination of both losses is the total loss function to be minimized:

$$l_{total} = l_l + \lambda l_h \tag{13}$$

where λ is a coefficient balancing two losses.

3.4. DDNSM ingredients

In this section we provide a more detailed description of main ingredients of our deep learning framework for approximating the high-resolution designs:

- Specifying the number of polytopes $\mathcal{N}$, halfplanes $\mathcal{M}$ cf. Eq. (10) and the weight parameter λ cf. Eq. (13) in the loss function: These parameters, in general, can be treated as hyperparameters/learning parameters; however, they are user-defined in the present paper. The value of each parameter is provided in Section 4.
- Specifying the deep neural network architecture: The deep neural network (DNN) in this paper is a fully-connected feedforward network. The DNN maps simulation parameters to the coefficients of the DNSM. The numerical details of DNN, i.e. number of layers, nodes, the learning rate and the number of epochs (number of iterations in the gradient-descent approach) are provided for each numerical example in Section 4. These numerical details similarly to the previously mentioned set of parameters are user-defined and are determined such that they yield computationally efficient yet sufficiently accurate predictions.
- Providing the simulation parameters i.e. loads, boundary conditions and volume fractions as inputs to the DNN: The output of DNN is the DNSM coefficients w_{ijk}, b_{ij} cf. Eq. (9) as shown in Fig. 3. The spatial values denoted by x_k in Eq. (9) are also specified in different resolutions i.e. $\boldsymbol{x}^L$, $\boldsymbol{x}^M$ and $\boldsymbol{x}^H$. These values are discretized in the range $[-1, 1]^2$ similarly to the approach for parameterizing the load location cf. Section 2.3.
- The output of the DDNSM model (i.e. union of deep neural network and DNSM): The output is an image that can be approximated in all three resolutions. The loss function is computed according to Eqs. (11), (12), (13) i.e. the loss function is the difference between the approximated images and the actual images. It is noted that the sensitivity of the loss function with respect to the DNSM coefficients, and the sensitivity of the DNSM coefficients with respect to DNN parameters are automatically calculated via automatic differentiation.
- Once the DDNSM is trained i.e. the DNN parameters are identified, high-resolution images on unseen simulation parameters are approximated by inputting those unseen simulation parameters to DNN (which subsequently approximates the DNSM coefficients) and the high-resolution spatial values to DNSM model.

To better illustrate the manner in which images are approximated from polytopes, we show two different approximated topology optimized designs and their associated polytopes in Figs. 4 and 5. To generate these plots we have considered $\mathcal{M} = 8$ halfplanes and have varied the number of polytopes $\mathcal{N} = 5, 15, 30$. It is shown that increasing the number of polytopes, reveals more detailed features in the images; however, we have empirically found that larger $\mathcal{N}$ does not necessarily improve the quality of approximation in our datasets (i.e. does not yield better initial guesses). We provide a detailed numerical study in Section 4.2 to illustrate this point.

Generation of training samples associated with high resolution designs is costly. In the next section, we will discuss details of an algorithm which selects a set of high resolution samples Ω_h (with small size) for training.

3.5. Importance sampling for training

Reconstructing an image using the union of its polytopes enables us to identify complex shapes. Given two shapes with the same number of polytopes, the simpler shape has more polytope overlap than the more complex shape, because it is not necessary for the network to find distinct polytopes when similar ones are enough to minimize the loss. The smaller overlap in image analysis is the key idea behind our sample selection. This idea is in accordance with the sample selection strategy in the parametric space in [28] where the authors use the Euclidean distance between parametric structure responses as a measure to choose important samples.

The similarity between two polytopes, $\mathcal{P}_i, \mathcal{P}_j \in \mathbb{R}^{H\times W}$ can be calculated using the intersection over union (IoU) ratio which is expressed as

$$\text{IoU} = \frac{\mathcal{P}_i \cap \mathcal{P}_j}{\mathcal{P}_i \cup \mathcal{P}_j} \tag{14}$$

where $\mathcal{P}_i \cap \mathcal{P}_j$ and $\mathcal{P}_i \cup \mathcal{P}_j$ are obtained as

$$\begin{aligned} \mathcal{P}_i \cap \mathcal{P}_j &= \sum_{k=1}^{H}\sum_{l=1}^{W} \mathcal{P}_i^{(kl)}\mathcal{P}_j^{(kl)}, \\ \mathcal{P}_i \cup \mathcal{P}_j &= \sum_{k=1}^{H}\sum_{l=1}^{W} (\mathcal{P}_i^{(kl)} + \mathcal{P}_j^{(kl)}) - \sum_{k=1}^{H}\sum_{l=1}^{W} \mathcal{P}_i^{(kl)}\mathcal{P}_j^{(kl)}. \end{aligned} \tag{15}$$

Therefore, for a shape with $\mathcal{N}$ number of polytopes, the total overlap value is proportionate to

$$\text{IoU}_{tot} = \frac{1}{2}\sum_{i=1}^{\mathcal{N}}\sum_{j=1, j\neq i}^{\mathcal{N}} \frac{\mathcal{P}_i \cap \mathcal{P}_j}{\mathcal{P}_i \cup \mathcal{P}_j}. \tag{16}$$

Fig. 6 shows an example of polytope overlap of two shapes. The simpler shape (i.e., the square) has more polytopes overlapping it than the other one.

In order to choose where in the simulation parameter space to sample the high resolution training examples Ω_h using this method, first, the network is trained just using the low resolution shapes Ω_l to obtain the polytopes of each training sample. Then using the polytopes of training examples derived at the previous step, the IoU_{tot} ratio is computed for all training samples. Finally, the parametric samples associated with the lowest ratio are determined and high resolution designs associated with these samples are selected for training.

4. Numerical studies

4.1. L-Bracket

The L-shape structure is frequently used as a benchmark example in the structural topology optimization studies [77,78].

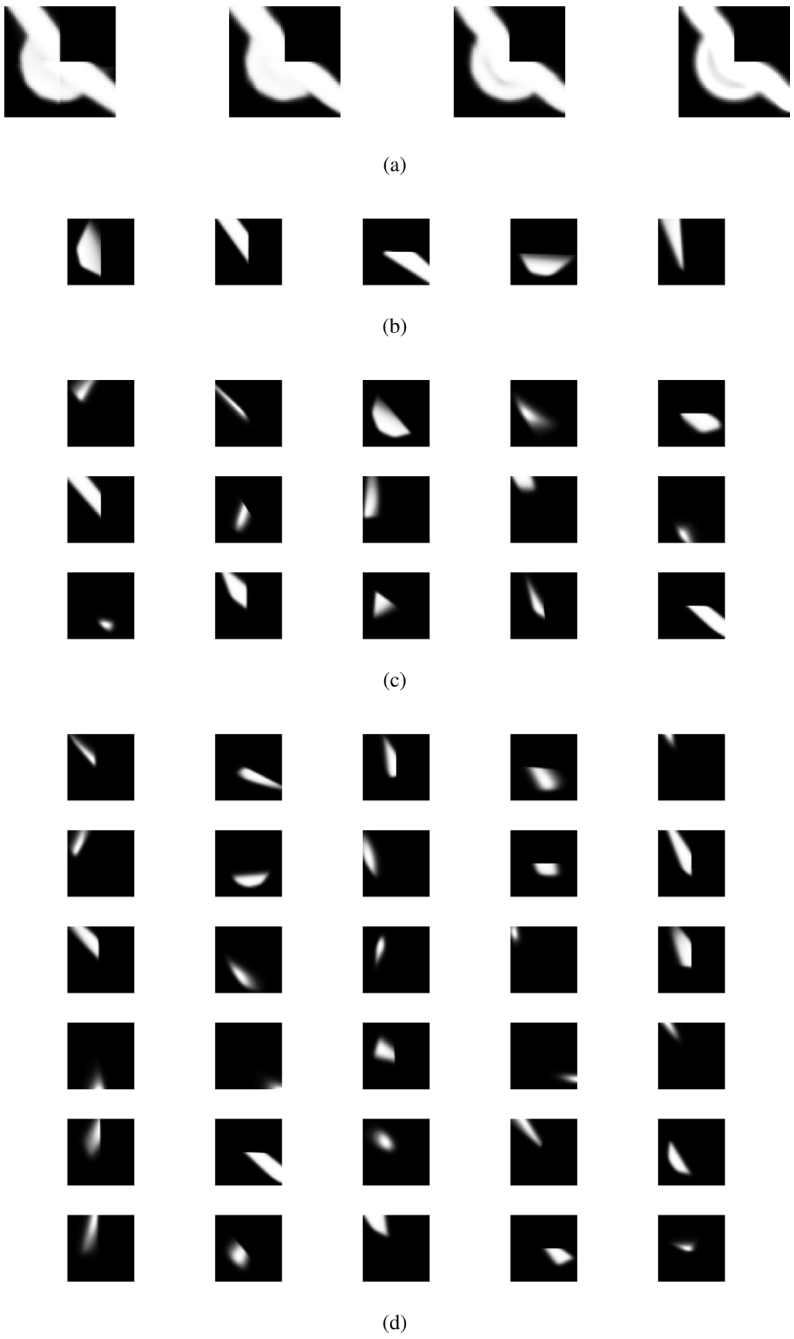

Fig. 4. (a) Reconstructed high resolution designs using $\mathcal{N} = 5,\ 15,\ 30$ polytopes (first three images) and the actual high resolution design (last image), (b) its $\mathcal{N} = 5$ associated polytopes, (c) its $\mathcal{N} = 15$ associated polytopes and (d) its $\mathcal{N} = 30$ associated polytopes obtained via DDNSM approach.

In this example, we use this type of structure to generate an image dataset. Fig. 7 shows the geometry, boundary conditions and the loading parameters that are used in this example. The domain is discretized using 64×64, 32×32 and 16×16 standard square finite elements for high, medium and low resolution optimizations. We assume a plane stress condition for the structural analysis and consider the location of load p_1 as an integer number in the range $[0, 32]$ for high resolution simulations and the angle of load p_2 in the range $[0, \pi]$. Accordingly the range of p_1 for medium and low resolution are $p_1 \in [0, 16]$ and $p_1 \in [0, 8]$ and the range for p_2 is unchanged. We use fixed filter sizes $r_{min} = 1.5, 2.5, 4.5$ and volume fraction of $v = 0.4$ for generating $n = 2000$ designs from low to high resolution for all samples in the datasets. Fig. 8 shows 5 different samples of high, medium and low resolution topology designs. As expected lower resolution images are not capable of exhibiting fine details however they still exhibit the main building blocks of the image or structure. We take advantage of their inexpensiveness and use them in conjunction with higher resolution images which contain fine details in the design to train the learning machine. It is also important to point out the scenarios where the lower resolution designs are less informative e.g. the fourth design in Fig. 8. It is apparent that the lower resolution designs do not exhibit fine details and learning from these particular instances does not necessarily result in improvement of the predictive model. However we note that our learning machine learns from a set of data points which contains more informative instances across resolutions; hence in an average sense we expect that our deep learning model can produce near optimal high resolution designs. This is confirmed by the reduction in the average number

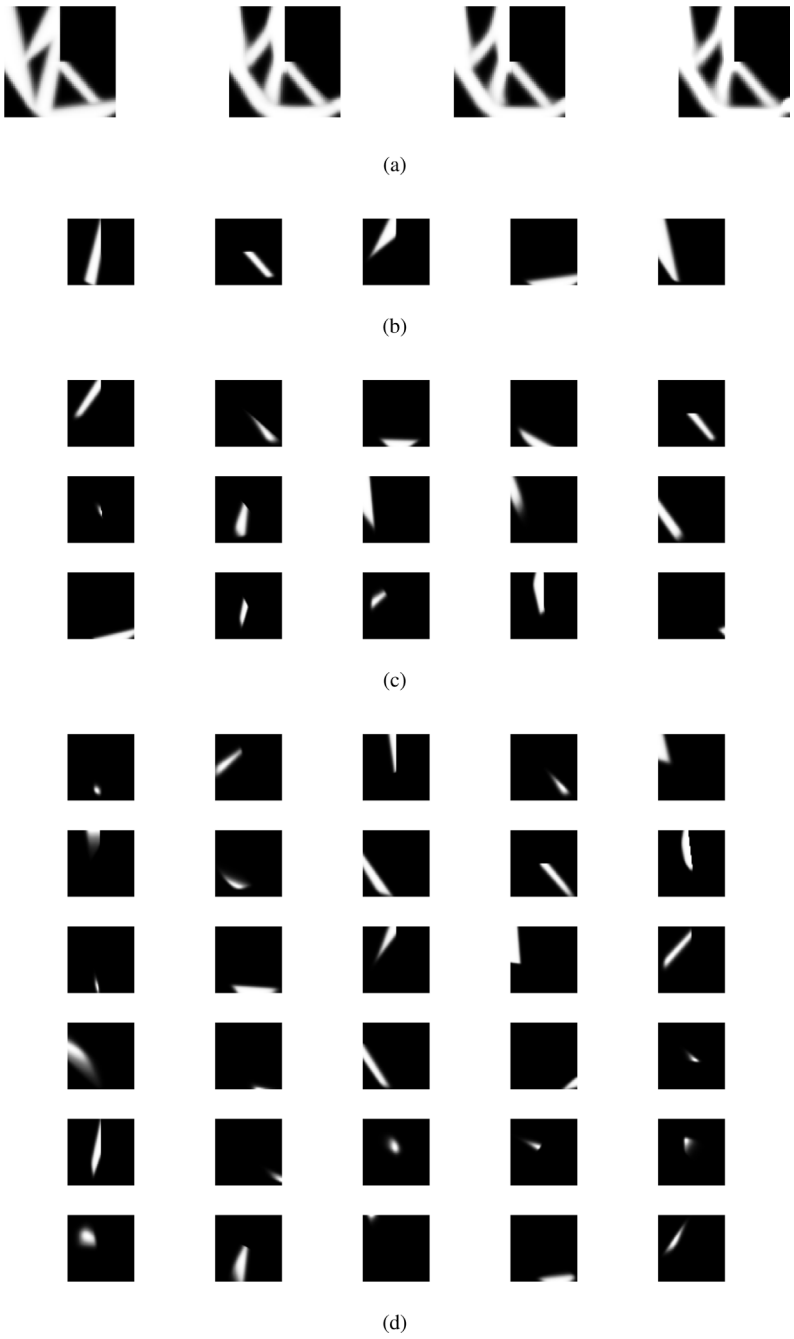

Fig. 5. (a) Reconstructed high resolution designs using $\mathcal{N} = 5,\ 15,\ 30$ polytopes (first three images) and the actual high resolution design (last image), (b) its $\mathcal{N} = 5$ associated polytopes, (c) its $\mathcal{N} = 15$ associated polytopes and (d) its $\mathcal{N} = 30$ associated polytopes obtained via DDNSM approach.

Fig. 6. Examples of two reconstructed shapes with the same number of polytopes and half-spaces, i.e. $\mathcal{N} = 2$, $\mathcal{M} = 4$. It is noted we only show the polytopes in (a), (c) that generate the reconstructed shapes in (b), (d) and the half-spaces are not shown. The shaded area shows the amount of polytope overlap. The amount of overlap of the square shape, which corresponds to the simpler shape, is larger than that of the plus sign shape.

of iterations as reported in e.g. Table 1. We also note that the importance sampling approach introduced in this paper could alleviate this issue. With importance sampling it is expected that more complex topologies are included (as high-resolution

Algorithm 1 Importance Sampling for High Resolution Shape Selection

1: $\mathcal{P}^k \leftarrow$ Train the DDNSM network using $(\mathcal{D}_k, \mathcal{S}_k^L); k \in (1, ..., |\Omega_l|)$ to obtain the polytopes $\mathcal{P}^k$ of each shape $\mathcal{S}_k^L$.
2: **for** $k = 1$ to $|\Omega_l|$ **do**
3: $\quad \text{IoU}_{tot}^k \leftarrow$ compute $\frac{1}{2}\sum_{i=1}^{\mathcal{N}}\sum_{j=1, j\neq i}^{\mathcal{N}} \frac{\mathcal{P}_i^k \cap \mathcal{P}_j^k}{\mathcal{P}_i^k \cup \mathcal{P}_j^k}$
4: **end for**
5: $\Omega_h =$ Select the first $|\Omega_h|$ indices of IoU_{tot} that correspond to the lowest ratio values.
6: Return Ω_h

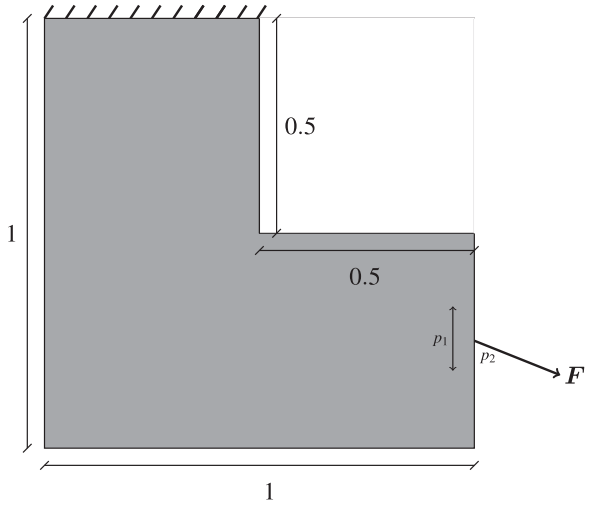

Fig. 7. Geometry, boundary conditions and load parameters for the L-Bracket structure.

Table 1
L-bracket design: Rate of success in number of iterations and compliance estimation, and average number of iterations for four datasets cf. Eq. (17).

Case	RoS in iterations %	RoS in compliance %	Average no. of Iters.
1	95.5	98.25	16.53
2	96.25	100	15.60
3	95.5	99.25	15.99
4	95.75	100	15.01

training data points) in the training dataset which subsequently improves the prediction capability of the deep network model.

Remark 4.1. We note that in our deep learning approach, lower resolution designs provide information about the general outline of the optimized structures. The fine features are captured by the high resolution images in the training set. If the minimum length scale required in the problem is relatively small, the lower resolution datasets should be generated with the smallest possible filter size (which imposes minimum length scale) such that the optimized designs show the general outline or large scale features of the structure. This may also require successive decrease of the largest mesh size until the lowest resolution designs exhibit the general outline of the structure. At the end of this section, we provide a numerical experiment which gives a quantitate estimate on the performance of the multiresolution DDNSM compared to a single high resolution DDNSM. In light of this experiment, we quantitatively assess the computational gain as well as degradation in the high resolution prediction using the multiresolution DDNSM.

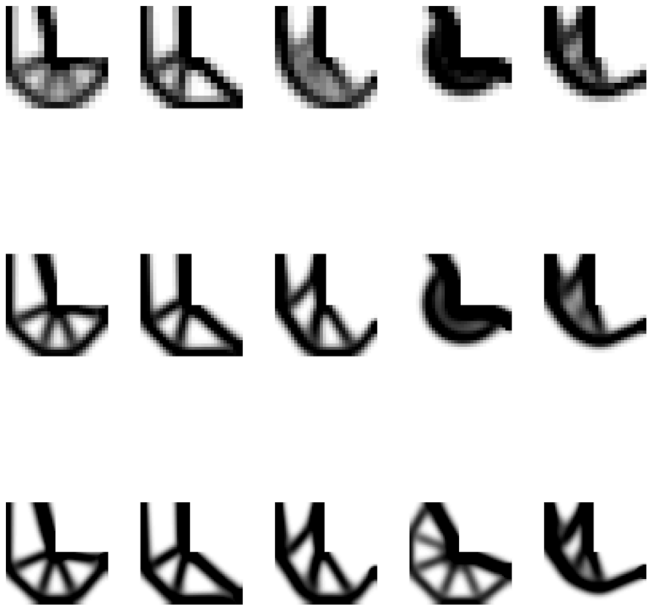

Fig. 8. Different samples of L-Bracket design in three resolutions: low 16 × 16 (top row), medium 32 × 32 (middle row) and high 64 × 64 (bottom row).

The DDNSM consists of 8 fully connected layers which takes the design parameters p_1, p_2 as input and produces the coefficients of DNSM as output. The number of nodes for 7 inner layers are $\{32, 256, 512, 1024, 2048, 2048, 1024\}$. For L-Bracket dataset, the number of DNSM coefficients (equivalent to the number of nodes in the last layer) is $15 \times 8 \times 3$ where 15 corresponds to the number of polytopes, 8 corresponds to the number of discriminants (half-spaces), and 3 corresponds to the number of parameters that identify a half-space in 2D. The number of polytopes and discriminants have been selected using the cross validation method such that the network is trained for different number of polytopes and discriminants and the ones that resulted in better reconstructions were selected. The learning rate is set to $1e{-}4$ and the number of gradient-descent iterations (epochs) is set to 250. Furthermore, λ cf. Eq. (13) is set to 0.1 for both L-Bracket and Heatsink datasets (cf. Section 4.2). We consider total of $n_{train} = 1600$ samples for training and $n_{test} = 400$ samples for testing the network model. We however vary the number of training samples to investigate the performance of the machine learning with respect to training size. To this end, we base our training on low resolution L and medium resolution M samples in addition to 20% of their size with high resolution images. In particular we consider the following scenarios:

$$
\begin{array}{lllll}
\text{Case 1}: & \text{1600 low resolution samples} & + & \text{320 high resolution samples} \\
\text{Case 2}: & \text{320 medium resolution samples} & + & \text{64 high resolution samples} \\
\text{Case 3}: & \text{800 medium resolution samples} & + & \text{160 high resolution samples} \\
\text{Case 4}: & \text{1600 medium resolution samples} & + & \text{320 high resolution samples}
\end{array}
\tag{17}
$$

It should be noted that in the first part of the experiment we select the high resolution samples randomly. Fig. 9 shows the actual high resolution (top row) and reconstructed structures associated with Cases 1 (middle row) and 4 (bottom row). It is visually apparent that the Case 4 which mainly uses the medium resolution training samples performs better compared to Case 1 which mainly uses the low resolution training samples.

To numerically investigate the performance of these cases we use the reconstructed structures as initial guesses and use the optimizer (based on the optimality criteria) [70] to find the

Fig. 9. Samples of L-Bracket design: actual high resolution (top row), reconstructed designs with dataset of Case 1 (middle row), and Case 4 (bottom row).

final structure. We count the number of iterations that each initial guess takes to generate optimal structure and we also compute the discrepancy between the final compliance and the initial compliance against the actual structure's compliance. In particular we use normalized errors $e_{C_{ini}} = |C_{ini} - C_{exc}|/C_{exc}$ and $e_{C_{fin}} = |C_{fin} - C_{exc}|/C_{exc}$ to assess the performance of the initial guesses where C_{ini}, C_{fin} and C_{exc} denote the compliance associated with the initial guess (reconstructed structure), final structure (from the initial reconstructed structure) and the actual structure.

Fig. 10 shows the number of iteration and the relative errors for Cases 1 and 4. We also consider the rate of success based on the number of iterations and proximity of compliance to the actual structure's compliance. More specifically to define the rate of success we count the number of samples that yield smaller iterations than the actual optimizer and count the number of samples that yields more accurate compliance values out of all 400 training samples. Table 1 lists the rate of success in all cases in terms of the above mentioned measures as well as the average number of iterations. The average number of iterations for the test samples with uniform initial guess is 30.1975. We observe that the Case 4 with the highest number of high resolution data points yields the lowest average number of iterations.

To investigate the applicability of this approach to other parameterizations, we generate additional datasets considering variation in the boundary condition and volume fraction in addition to the existing parametric loads. For the case of variation in volume fraction and boundary condition we consider $p_3 \in [0.3, 0.5]$ and $p_3 \in [0, 64]$ (associated with the high resolution mesh) respectively. The parameterized boundary condition (p_3) is illustrated in Fig. 11. Similarly to the previous case we generate four different datasets as follows:

$$
\begin{array}{lllll}
\text{BC-L-Low :} & \text{1600 low resolution samples} & + & \text{320 high resolution samples} \\
\text{BC-L-Medium :} & \text{1600 medium resolution samples} & + & \text{320 high resolution samples} \\
\text{VF-L-Low :} & \text{1600 low resolution samples} & + & \text{320 high resolution samples} \\
\text{VF-L-Medium :} & \text{1600 medium resolution samples} & + & \text{320 high resolution samples}
\end{array} \tag{18}
$$

Table 2
L-bracket design: Rate of success in number of iterations and compliance estimation, and average number of iterations for four datasets associated with parameterized boundary conditions and volume fractions.

Case	RoS in iterations %	RoS in compliance %	Average no. of Iters.
BC-L-Low	78.75	99.00	20.05
BC-L-Medium	88.25	99.5	14.81
VF-L-Low	80.0	99.25	24.56
VF-L-Medium	89.0	99.5	17.78

Table 3
L-bracket design: Rate of success in number of iterations and compliance estimation, and average number of iterations for two cases of low and medium resolution initial guesses.

Case	RoS in iterations %	RoS in compliance %	Average no. of Iters.
L Res. Ini. Guess	85.75	97.25	22.01
M Res. Ini. Guess	96.75	98.75	10.44

where BC-L-Low and BC-L-Medium correspond to the datasets with variation in boundary condition and load, and consist of mainly low and medium resolution samples respectively. Similarly VF-L-Low and VF-L-Medium correspond to the datasets with variation in volume fraction and load. Fig. 12 shows the reconstructed images using the above four datasets. It is also visually apparent that the reconstructed image associated with the medium resolution training data is closer to the actual image compared to the reconstructed image with low resolution training data.

To assess the performance of these training datasets quantitatively, similarly to the previous experiment we use the reconstructed images as initial guess for topology optimization. Fig. 13 shows the iteration counts and relative errors in compliance associated with BC-L-Medium and VF-L-Medium. Table 2 provides quantitative performance measures in terms of number of iterations and compliance estimation for four datasets. It is again noticeable that the training datasets associated with medium resolution designs perform better than the datasets associated with low resolution designs. It should be noted that the average number of iterations for direct high-resolution design in the cases of variations in boundary condition and volume fraction is 26.2675 and 33.7375 respectively.

To further investigate the effect of initial guess and its computational cost impact on final design we repeat the above experiment with low and medium resolution initial guesses. In other words, we generate high resolution designs by using the low and medium resolution layouts. For example if there is a solid pixel (with value 1) in the medium resolution we consider a mesh with the size of 2×2 pixels (with values 1) in the same location in high resolution. We perform this experiment for $n_{test} = 400$ test data points associated with datasets 1 and 4. The comparisons for number of iterations and compliance estimation are shown in Fig. 14 and Table 3. It is observed that the average number of iterations is reduced once the medium resolution data is used (10.44 in Table 3 versus 15.01 in Table 1) however it is the opposite for the low resolution data (22.01 in Table 3 versus 16.53 in Table 1). It should however be noted that to use the medium resolution data as initial guess, we need to generate such medium resolution designs via direct optimization. This will impact the computational cost analysis which will be discussed in detail at the end of this section.

Next, we test the performance of the importance sampling. To this end we consider two scenarios: Case 1 which is the last case in the previous experiment i.e. 1600 M + 320 H where the high resolution samples are selected randomly and Case 2 with 1600 M

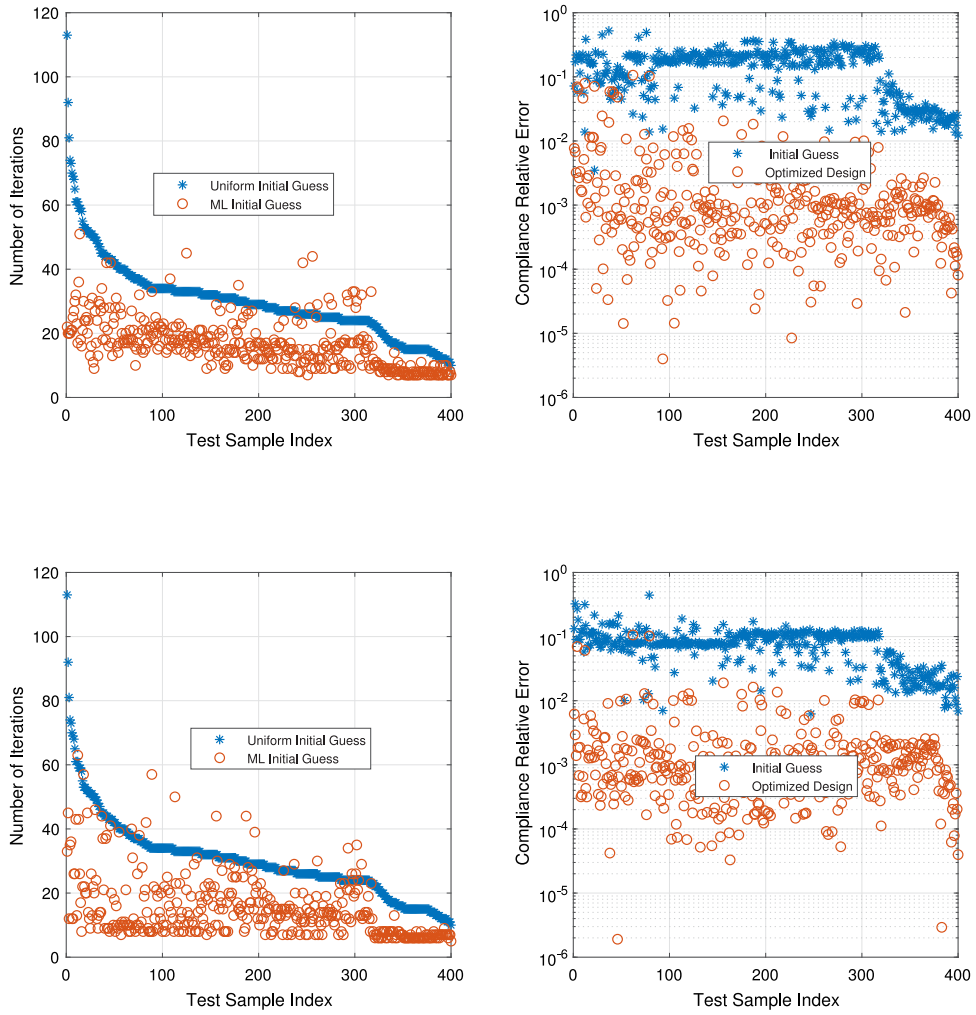

Fig. 10. L-bracket design: Iteration counts (left) and relative errors in compliance $e_{C_{ini}}, e_{C_{fin}}$ (right) for two cases: Case 1 (top row) and Case 4 (bottom row).

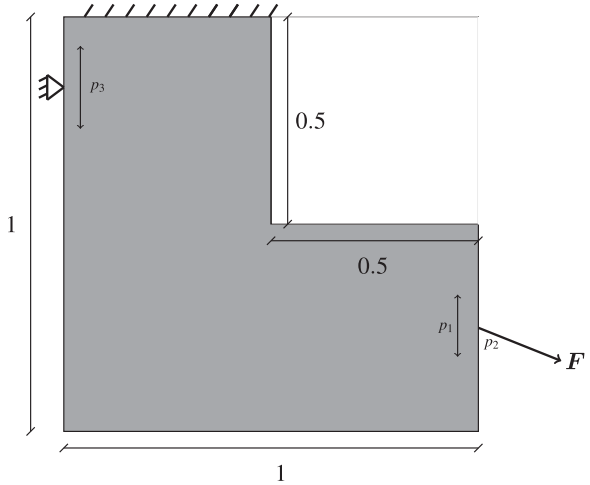

Fig. 11. Geometry and parameterized load and boundary condition for the L-Bracket structure. In this case a node with varying location along the left side of the structure is fixed in both x and y directions.

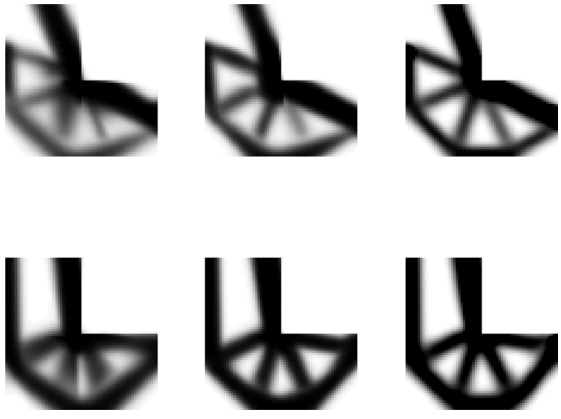

Fig. 12. Different samples of L-Bracket design associated with BC-L (top) and VF-L (bottom) datasets: reconstructed high resolution design using mostly low resolution training data (left), medium resolution training data (middle) and original high resolution design (right).

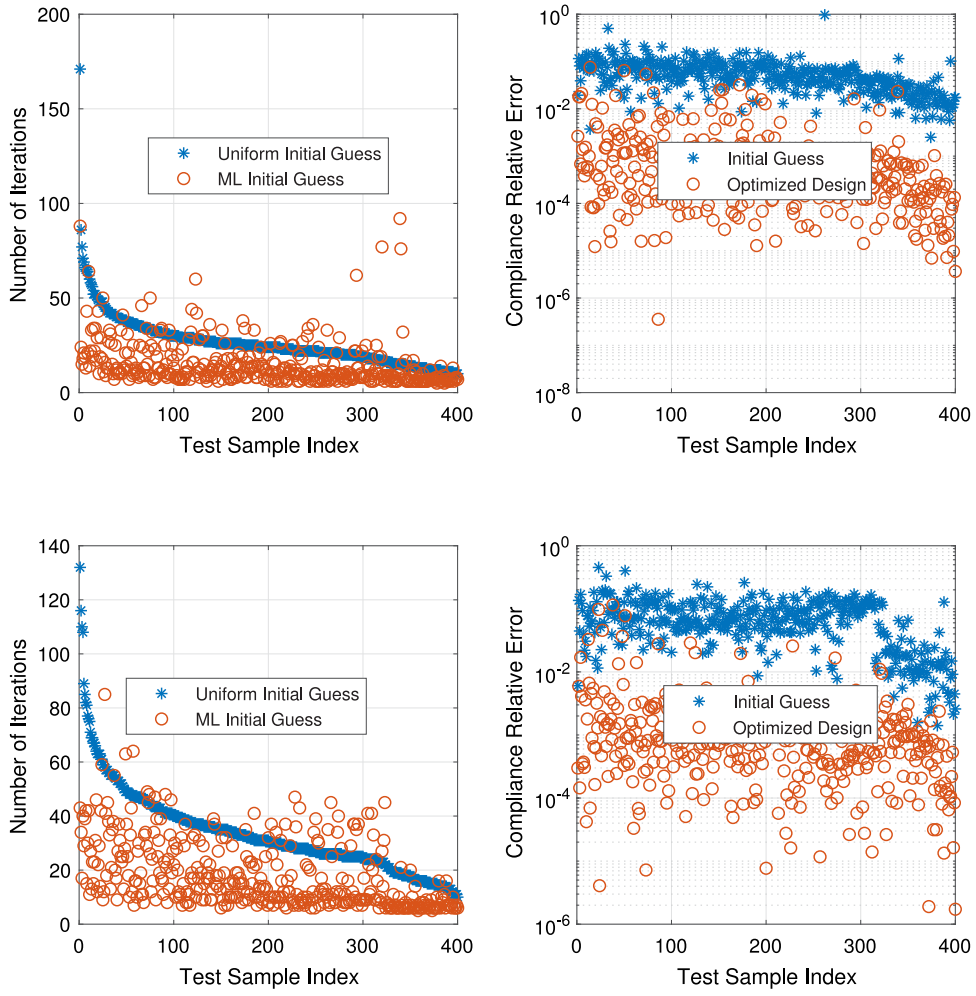

Fig. 13. L-bracket design: Iteration counts (left) and relative errors in compliance $e_{C_{ini}}, e_{C_{fin}}$ (right) for two cases: parametric boundary conditions and loading (top row) and parametric loading and volume fractions (bottom row).

+ 320 H where the high resolution samples are selected based on the strategy described in Section 3.5. More precisely, the DDNSM is first trained on medium resolution images (images with resolution 32×32). After the training, we obtain the polytopes of each training sample using the DNSM coefficients. Subsequently we select the high resolution samples using the ratio described in Section 3.5.

To illustrate the overlap ratio we show two example of the optimal structures in addition to their underlying polytopes in Fig. 15. Each shape consists of 15 polytopes. Based on importance sampling method, the more complex shape yields the less polytope overlap ratio. As seen, the simpler shape shown in (a) has larger ratio value than the more complex shape shown in (c).

Fig. 16 shows the number of iterations and the relative errors for the dataset generated using the importance sampling and Table 4 lists the rate of success in iteration and compliance prediction for two datasets i.e. random selection and important sample selection. As seen, the rate of success in iteration counts and the average number of iterations improve appreciably in the dataset which is associated with important sample selection.

Table 4
L-bracket design: Rate of success in number of iterations and compliance estimation, and average number of iterations for datasets generated using the random and importance sampling.

Case	RoS in iterations %	RoS in compliance %	Average no. of Iters.
Random	95.75	100	15.01
Importance sampling	97.25	100	14.25

Analysis of Computational Cost

It is beneficial to assess the computational savings by the machine learning approach quantitatively. To this end we assign a computational cost to each finite element analysis based on its mesh size. For a $2D$ finite element we consider the computational cost as $C_{FE} = (1/h)^2$ where h is the mesh size. We assume $h_L = 1, h_M = 0.5$ and $h_H = 0.25$; therefore the computational cost for each finite element analysis for different resolutions is $C_{FE} = 1, 4, 16$.

As an example the computational cost C_c for generating the dataset in Case 4 is $C_c = (1600 \times 4 + 320 \times 16) \times 30.1975 = 3.478e5$ where we note that 30.1975 is the average number of iterations and is assumed to be the same for medium resolution optimizations. It is expected that the medium resolution optimization takes slightly fewer number of iteration compared to the high resolution optimization however with the aforementioned assumption we estimate the computational cost of training more conservatively i.e. the training cost is smaller in practice. Since the ultimate goal is to generate the high resolution designs we

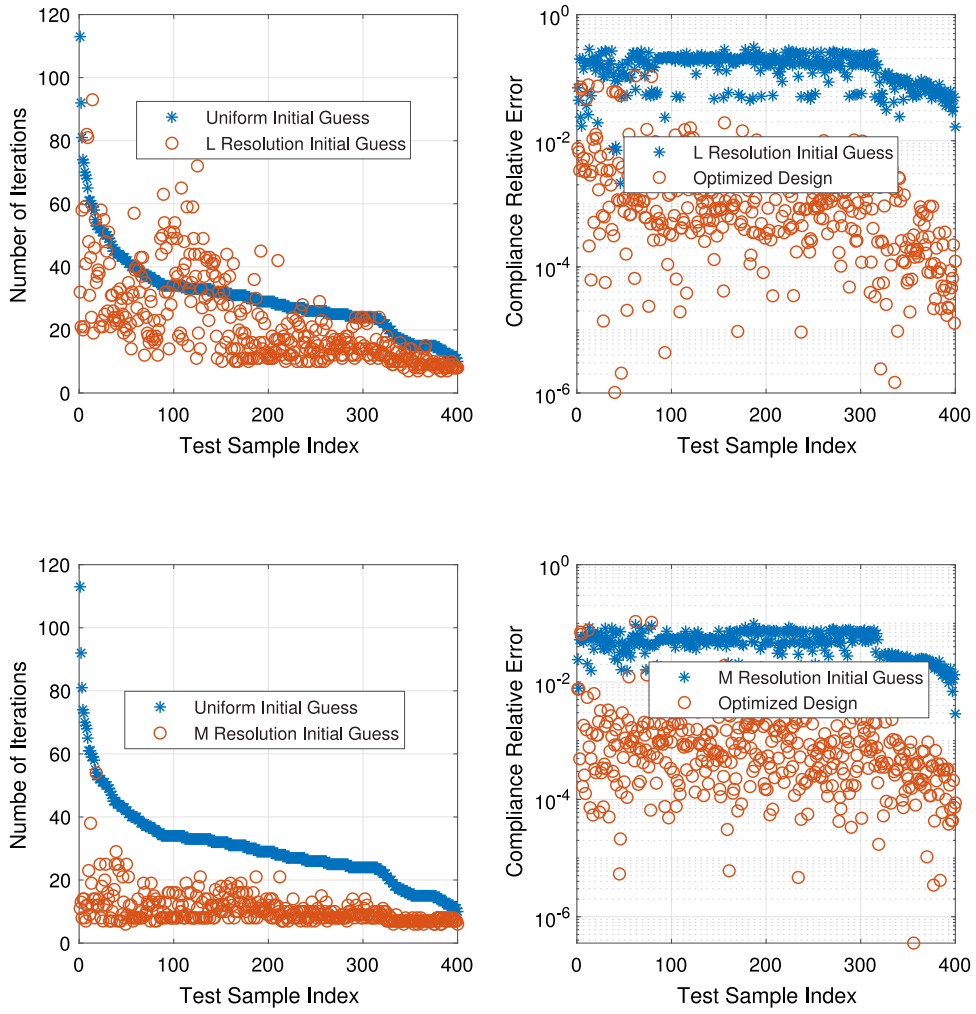

Fig. 14. L-bracket design: Iteration counts (left) and relative errors in compliance $e_{C_{ini}}, e_{C_{fin}}$ (right) for two cases of low and medium resolution initial guesses: Case 1 (top row) and Case 4 (bottom row).

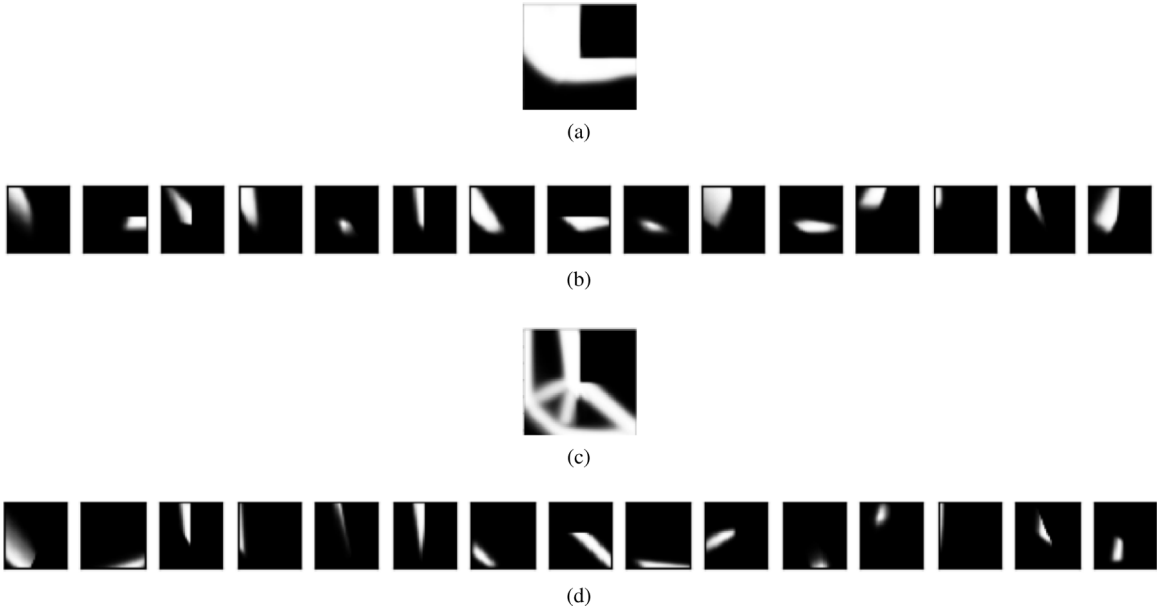

Fig. 15. (a) An optimized structure and (b) its associated polytopes with the ratio value $\text{IoU}_{tot} = 11.13$; (c) another optimized structure and (d) its associated polytopes with the ratio value $\text{IoU}_{tot} = 3.92$. In this example $\mathcal{N} = 15$, $W = 32$ and $H = 32$. As seen, the ratio value for the simpler shape shown in (a) is greater than the ratio value for the more complex shape (c).

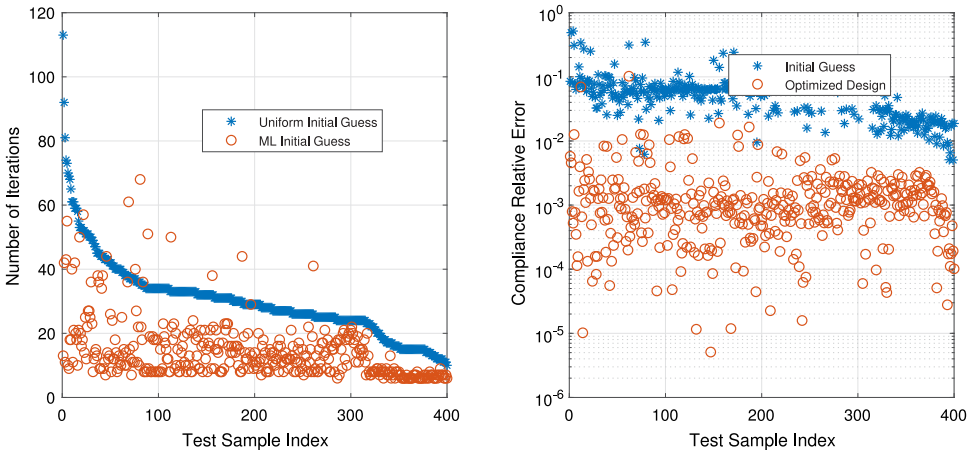

Fig. 16. L-bracket design: Iteration counts (left) and relative errors in compliance $e_{C_{ini}}, e_{C_{fin}}$ (right) for the dataset generated using the importance sampling.

solve the following equation to find the number of high resolution samples which yields the same computational cost for designing via machine learning and direct optimization:

$$3.478e5 + (14.25 \times 16)n = (30.1975 \times 16)n \tag{19}$$

The number of high resolution samples is obtained as $n = 1363$ which indicates that in the occasions when we need to find a large number of high resolution structures i.e. $n > 1363$ the proposed machine learning approach will be more cost efficient. Such situations include when the task of optimization is performed in a nested fashion i.e. the optimized topology is explored in a broader scale optimization or when human designers need to explore the space of optimal designs [43]. We perform the same analysis similarly to Eq. (19) to find the number of high resolution designs associated with different datasets that we used for training in this example. Table 5 lists these numbers of high resolution designs for different datasets. It is observed that the dataset of Case 2 yields the smallest number of high-resolution designs. This indeed implies that the cost of training data generation relative to the cost of direct high-resolution optimization, in this case is the lowest among all datasets.

It is also beneficial to assess the computational cost when using the medium and low resolution designs as shown in Table 3. Based on the results, the cost of generating a high resolution sample from these initial guesses is $C_c = 16 \times 10.44$ and $C_c = 16 \times 22.01$. On the other hand, the cost of generating these designs for each sample is $C_c = 4 \times 30.19$ and $C_c = 1 \times 30.19$ (assuming low and medium resolution designs take the same number of iterations as high resolution designs). However in both cases the total computational cost $C_C = 288,382$ is smaller than generating a high resolution design from a uniform density distribution $C_c = 16 \times 30.19 = 483$. This simple analysis indeed shows that using such low and medium resolution designs can be another viable option for initial guesses as they reduce the overall computational cost for generating high resolution designs in this example. However it is noted that the presented computational results for the number of iterations are solely relevant to this particular example. In general, there might be situations where the decrease in the number of iterations is negligible e.g. using medium resolution initial guess would result in 450 iterations instead of 500 iterations. In those situations, starting from a medium resolution design does not justify the overall cost.

Table 5
Computational cost analysis for L-bracket design: The number of high resolution designs for which the cost of building deep learning surrogate is equal to direct optimization, associated with different datasets used for training. In the situations where a larger number of high resolution designs are needed, the deep learning surrogate will be more cost efficient.

Case	Number of Hi. Res. designs	Case	Number of Hi. Res. designs
1	928	BC-L-Low	1776
2	297	BC-L-Medium	1651
3	765	VF-L-Low	1544
4	1431	VF-L-Medium	1522

We now investigate the actual computational time and assess the computational savings afforded by DDNSM. Our experiments in this paper are performed on two computers, one for data generation and performing convergence studies and the other one for performing the deep learning including training and testing. The specifications for the computational units for these two computers are (a) Intel 12 Core $i7-5930K$ @ 3.5 GHz CPU with 32 GB of RAM and (b) 2x Intel 8 Core $E5-2660$ @ 2.20 GHz processor with 64 GB of RAM and 2x Nvidia $K20$ *GPUs*. We note that these two computers are similar in terms of computational power; therefore to estimate the total time of generating optimized design images we simply add different components of computational time i.e. generating the dataset, training the network, generating images from the trained network and topology optimization using DDNSM-generated initial guesses. Another important point is that, as the problem size increases, generating the training dataset takes more time compared to network training. Therefore, asymptotically, the cost of network training will be negligible compared to the cost of generating datasets. Nevertheless for the sake of completeness, we include the cost of training and evaluating in our computational time analysis.

It is also noted that the data generation or topology optimization with SIMP is done via sequential CPU processing while the deep learning is performed on a GPU in this paper. In the following, we will compare the time associated with direct optimization with SIMP and the DDNSM approach. It should be noted that the DDNSM approach also includes the time that is associated with topology optimization using sequential CPU processing. In other words, both computational times i.e. SIMP and DDNSM times depend on the sequential CPU processing. Based on our experiments which will be reported in the following, we have found that

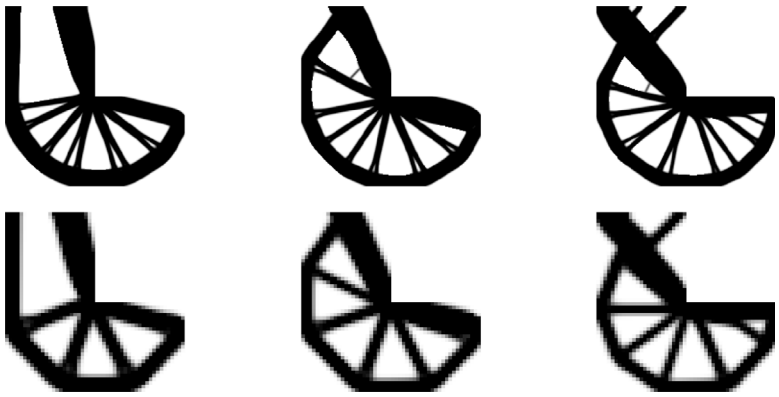

Fig. 17. L-bracket design: High (top) and low (bottom) resolution designs with 200 × 200 and 50 × 50 elements. Fine features are apparent in the high resolution designs.

the total time for SIMP is larger than DDNSM, sometimes with significant margin. Using parallel processing (i.e. converting CPU processing to GPU processing) decreases the gap between total computational times in these approaches; however the DDNSM approach remains more efficient so long as the number of test samples increases with higher rate than the number of parallel CPUs.

As an example, we provide the computational time breakdown for the dataset associated with Case 2. Different components of the computational time are as follows:

- Generating the dataset: 260.31 s
- Training the network: 1639.25 s
- Evaluating the network: 19.30 s
- Topology optimization with DDNSM initial guess: 9534.39 s

Note that the cost for evaluating the network corresponds to generating images for 10 000 high resolution designs which we use for comparison with the standard SIMP method (for the same number of 10 000 designs). Also note that the computational time for generating the dataset is rather small since it mainly consists of lower resolution designs which is the main advantage of our approach. The computational time for generating 10 000 low, medium and high resolution designs are 1103.91, 4094.16 and 20 203.75 s respectively. In Table 6 we provide the computational time for generating 10 000 high resolution designs using both methods: DDNSM and SIMP. We report two computational times, one considering every component that contributes to the total time in DDNSM and the other one considering the time for only evaluating the network and topology optimization with the DDNSM initial guesses. We also note that in a similar work [41], authors report only the evaluation time from the network which is significantly smaller compared to other components of time. As seen in this example, it takes only 19.3 s to evaluate the network 10 000 times. In other words it takes 0.002 s to generate one design with DDNSM after training while direct optimization takes 2 s. However, in this paper the setting for our method is different as we use the generated images as initial guesses to a subsequent topology optimization process. Nevertheless, from the results in Table 6, it is evident that the DDNSM approach outperforms SIMP in both cases when a large number of optimized design is needed, even for this rather small scale problem.

Table 6
L-bracket design: Computational time (in hours) for generating 10 000 high resolution L-Bracket designs.

Case/Method	DDNSM	SIMP
Computational time (with training)	3.18	5.61
Computational time (without training)	2.65	5.61

Comparison between single resolution and multiresolution DDNSM

As a final experiment in this example, we consider the comparison between a single high resolution DDNSM and a multiresolution DDNSM. The goal of this experiment is to objectively assess the trade-off between the accuracy and cost using both networks. To this end we consider a much higher resolution as the high resolution samples for training and testing the network. In particular, the H and L datasets in this experiment consist of 200 × 200 and 50 × 50 topology optimized designs. Three different samples of each dataset are shown in Fig. 17. In this example, we use small filter radii to allow the appearance of fine features in the final design. In particular we use $r_{min} = 2$ and 1.5 for high and low resolutions. Fine features are apparent in the designs shown in the first row. To train the networks we consider the following scenarios:

$$\begin{aligned} &\text{Single Resolution}: \quad 800 \text{ high resolution samples} \\ &\text{Multiresolution}: \quad 800 \text{ low resolution samples} \quad + \quad 160 \text{ high resolution samples} \end{aligned} \tag{20}$$

It is noted that in the single resolution case, we do not have the contribution of low resolution loss and the balancing coefficient in this case is $\lambda = 1$ cf. Eq. (13). The number of polytopes and discriminants is 35 and 8, the learning rate is set to $2.5e{-}4$ and the number of epochs is 650 and 250 for single resolution and multiresolution networks. We also use the importance sampling strategy for choosing the high resolution samples in the multiresolution case.

For testing the network we consider 200 samples. Fig. 18 shows the prediction from two networks. The first row shows five samples of the actual test dataset; second row shows the predictions from single resolution scenario and the last row corresponds to predictions from the multiresolution DDNSM.

It can be clearly seen that the single resolution net has been able to reveal more fine features. For example in the last image (the fifth image from left), the single resolution network has been able to recover the semi-circle in the image to some degree while in the multiresolution case the details are less apparent. However this performance increase from multiresolution to single resolution network comes with appreciable cost for training.

Before proceeding with the cost analysis, we introduce a new straight-forward measure to assess the prediction accuracy. This measure, e_{Img} is similar to the loss function that has been used in

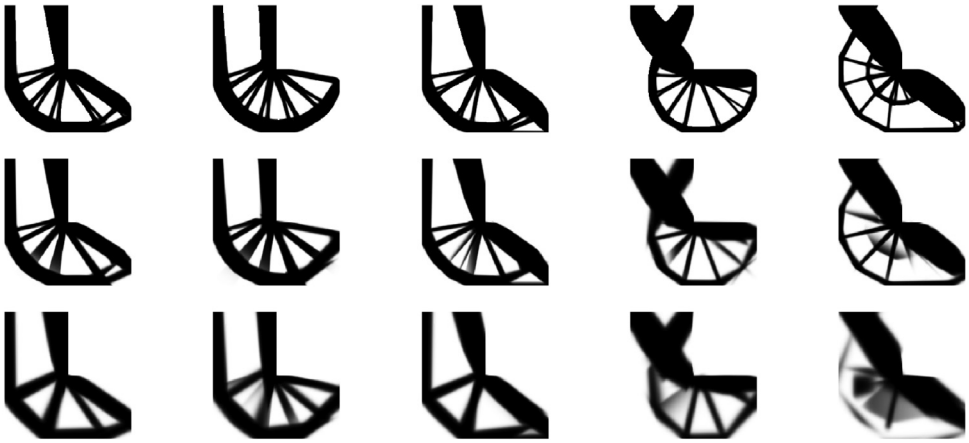

Fig. 18. L-bracket design: Actual high resolution images in the test dataset (top row), prediction from the single resolution DDNSM (middle row) and prediction from the multiresolution DDNSM (bottom row).

Table 7
L-bracket design: Rate of success in number of iterations and compliance estimation, average number of iterations, e_{lmg} and Computational time (in hours) for generating 1000 high resolution designs.

Case	RoS in Iter. %	RoS in Comp. %	Avg. no. of Iters.	e_{lmg}	Comp. Time
Single resolution	98.00	100	27.30	0.1814	24.68
Multiresolution	96.5	100	42.33	0.3566	13.09

training:

$$e_{lmg} = \frac{\|\hat{\mathcal{S}} - \mathcal{S}\|_F}{\|\mathcal{S}\|_F} \tag{21}$$

where $\mathcal{S}$ is the actual image (shape), $\hat{\mathcal{S}}$ is the predicted image and $\|.\|_F$ is the Frobenius norm. From Fig. 19 it is observed that single resolution network can almost always (except two samples) generate closer images to the actual ones. To further investigate the performance of the two networks, we repeat the same experiment similarly to previous examples and investigate the rate of success in compliance, iteration and average number of iterations. These results in addition to average of normalized errors in image prediction e_{lmg} and computational time for generating 1000 samples are shown in Table 7. The average number of iterations for designing a 200 × 200 mesh is 123.61. The computational time breakdown i.e. dataset generation, training the network, evaluating the network and subsequent topology optimization for 1000 samples for two networks are as follows: single resolution (58 640, 14 023.9, 5.11, 16 188.7) s and multiresolution (12 680, 9340, 5.17, 25 101.4) s. We emphasize that after training the network, generating 1000 high resolution images takes 5.11 and 5.17 s. This means that generating one approximate design from the neural net effectively takes a small fraction of a second, i.e. 0.005 s while direct optimization of 200 × 200 mesh takes 73 s.

From these results, it is deduced that the single resolution network outperforms the multiresolution network in terms of accuracy measures. However there is a significant difference in terms of the total computational time for these two networks. The training time i.e. the time associated with both dataset generation and training the network itself is significantly smaller (almost three times smaller) in the case of the multiresolution network. Therefore, in this case where 1000 test samples are sought the multiresolution network is significantly more efficient. It should also be noted that the single resolution network in this example has been able to appreciably decrease the number of iterations

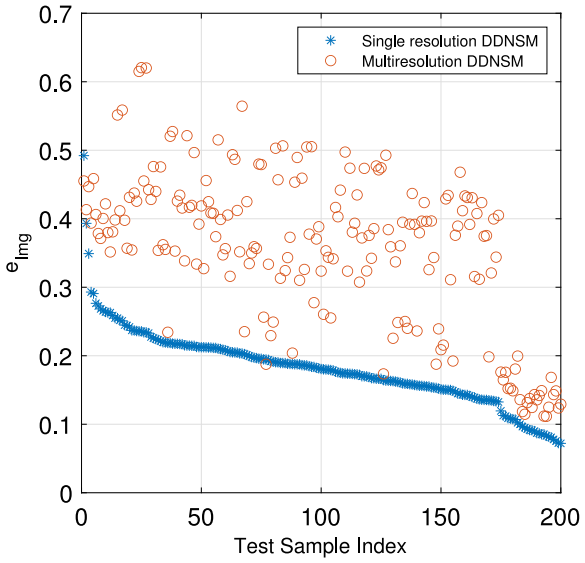

Fig. 19. L-bracket design: The normalized error in prediction of images.

(123.61 → 27.30). This implies that, according to our computational time analysis there is a breakeven point at which the total time of both networks is equal. A similar analysis to (19) for total time:

$$\underbrace{7.266e4 + 16.194 \times n}_{\text{Signle resolution time}} = \underbrace{2.202e4 + 25.106 \times n}_{\text{Multiresolution time}} \tag{22}$$

yields $n = 5683$. This means that for predicting less than 5683 optimized designs, the multiresolution network is more economical but as the number of test samples increases the single resolution network will be more efficient. In other words, if more cost is spent at the beginning for training, it would result in more efficient predictions for a large number of optimized designs $n > 5683$ which is expectable. However, if a smaller number of predictions $n < 5683$ is desired the multiresolution approach would be more suitable as it involves much smaller training cost.

4.2. Heat sink

The objective in this example is to use the multiresolution images to find the high resolution designs for a heat sink. The

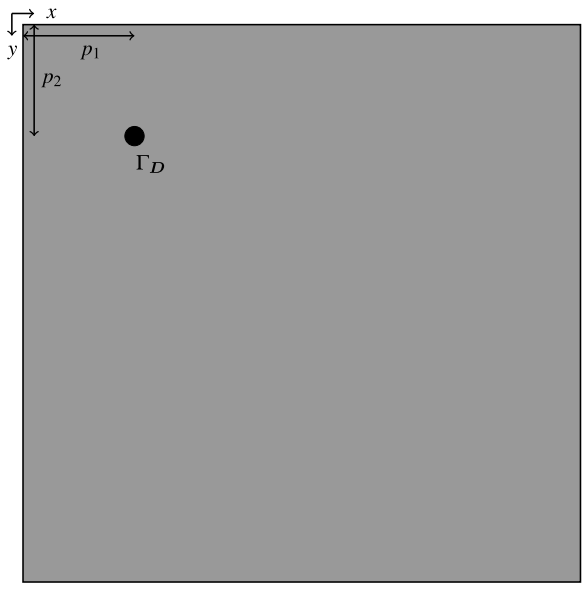

Fig. 20. Geometry and parameterized boundary conditions for the heat sink design. The temperature is zero at the boundary Γ_D.

objective in these designs is to minimize the thermal compliance [2]. The design domain shown in Fig. 20 is subject to a uniform heat and includes one heat sink with varying location. The varying location in this example is considered as the simulation parameter. Therefore the parameter space in this example includes two variables (location in x and y directions). It is noted that we consider only one nodal location as the heat sink throughout different resolutions. In other words, a point location in the low resolution domain is mapped to a point in the high resolution domain. The governing equation and boundary conditions for a steady state heat conduction problem are

$$\begin{cases} \nabla.q(\boldsymbol{x}) + b(\boldsymbol{x}) = 0 & \forall \boldsymbol{x} \in D \\ T(\boldsymbol{x}) = 0 & \forall \boldsymbol{x} \in \Gamma_D, \end{cases} \tag{23}$$

where $q(\boldsymbol{x})$ is the thermal flux and T is the temperature. The elliptic PDE associated with Eq. (23) is expressed as

$$\begin{cases} -\nabla.(\mathbb{K}(\boldsymbol{x})\nabla T(\boldsymbol{x})) = f(\boldsymbol{x}) & \forall \boldsymbol{x} \in D \\ T(\boldsymbol{x}) = 0 & \forall \boldsymbol{x} \in \partial D, \end{cases} \tag{24}$$

where $\mathbb{K}$ is the thermal conductivity matrix and $f(\boldsymbol{x})$ is the force function which is constant in all nodes in our example: $f = 0.01$. After finite element discretization, the temperature $\boldsymbol{T} \equiv T(\boldsymbol{x})$ is obtained from $\boldsymbol{KT} = \boldsymbol{F}$ where

$$\boldsymbol{K} = \int_D \mathbb{B}^T \mathbb{K} \mathbb{B} dD, \qquad \boldsymbol{F} = \int_D \mathbb{N}^T f dD \tag{25}$$

are the thermal stiffness matrix and thermal load vector, respectively, with $\mathbb{N}$ and $\mathbb{B}$ shape function matrix and its corresponding derivative matrix respectively. The thermal conductivity matrix $\mathbb{K}$ in this case is given by

$$\mathbb{K} = \mathcal{K}(\boldsymbol{\rho})\boldsymbol{I}_2, \tag{26}$$

where $\boldsymbol{I}_2$ is a 2×2 identity matrix and $\mathcal{K}(\boldsymbol{\rho})$ is the thermal conductance parameterized with respect to density $\boldsymbol{\rho}$.

Similarly to the previous examples we generate $n = 1000$ data points associated with low, medium and high resolutions. The DDNSM consists of 8 fully connected layers same as previous experiment where in this case inputs are the location in x and y directions. The number of polytopes and discriminants for this case are $\mathcal{N} = 25$ and $\mathcal{M} = 8$ respectively. We use

Table 8
Heat sink design: Rate of success in number of iterations and compliance estimation, and average number of iterations for four datasets.

Case	RoS in iterations %	RoS in compliance %	Average no. of Iters.
1	57.0	100	240.93
2	80.0	100	190.20
3	94.0	100	152.43
4	95.5	100	132.32

$n_{train} = 800$ samples for training and $n_{test} = 200$ for testing the network. Different resolutions in this examples i.e. H , M and L are 128×128, 64×64 and 32×32 meshes respectively where we again use standard square finite elements. Fig. 21 shows different samples of optimal designs in three resolutions. We consider similar scenarios to the previous example to test the performance of the machine learning algorithm. In particular we consider the following four datasets:

$$\begin{array}{lllll} \text{Case 1}: & \text{800 low resolution samples} & + & \text{160 high resolution samples} \\ \text{Case 2}: & \text{160 medium resolution samples} & + & \text{32 high resolution samples} \\ \text{Case 3}: & \text{400 medium resolution samples} & + & \text{80 high resolution samples} \\ \text{Case 4}: & \text{800 medium resolution samples} & + & \text{160 high resolution samples} \end{array} \tag{27}$$

The reconstructed images for three arbitrary samples are shown in Fig. 22. It is apparent that the reconstructed images in bottom row associated with Case 4 exhibit more details compared to the middle row associated with Case 1. The performance of these two datasets are tested via the number of iterations and relative error in compliance. Fig. 23 shows these metrics for two datasets. Again it is apparent that the dataset of Case 4 which uses 100% of the medium resolution data outperforms the Case 1 dataset which uses low resolution data. Table 8 lists the performance of the above mentioned four datasets. As expected as the number or resolution of training data points increases the performance improves. Also the average number of iterations without using the image-based machine learning is 243.32 and using e.g. the dataset of Case 4 reduces this time to almost half i.e. 132.32 iterations. To be more precise we perform the same calculations for computational cost as previous example. The computational cost for generating the dataset of Case 4 is $C_c = (800 \times 4 + 160 \times 16) \times 243.32 = 1.4015e6$. Using the following equation

$$1.4015e6 + (132.32 \times 16)n = (243.32 \times 16)n \tag{28}$$

we find $n = 789$ which again indicates that it is beneficial to use the machine learning strategy when finding $n > 789$ high resolution heat sink designs is desired.

Similar to our previous example, we also perform the computational time analysis. In this example, the computational time for generating 10 000 low, medium and high resolution designs are 1906.3, 20 526.05, 121 968.54 s respectively. The computational time for generating the dataset of Case 4, the network training, network evaluation and topology optimization with DDNSM initial guesses for 10 000 samples are 3593.58, 6068.00, 32.70 and 66 327.78 s. Table 9 lists the computational time for both methods. As expected, DDNSM outperforms standard SIMP. It is also apparent that, in this example which involves a larger scale problem compared to previous one, the difference in computational time is more significant. We also again emphasize that generating a heat sink topology optimized design with 128×128 elements takes 12.2 s using SIMP while generating an approximate design takes only 0.003 s with the trained network.

In this example we also repeat the experiment for importance sampling. To this end, we consider the dataset of Case 2 and perform network learning by using the high resolution training data selected randomly and via the proposed algorithm. Fig. 24

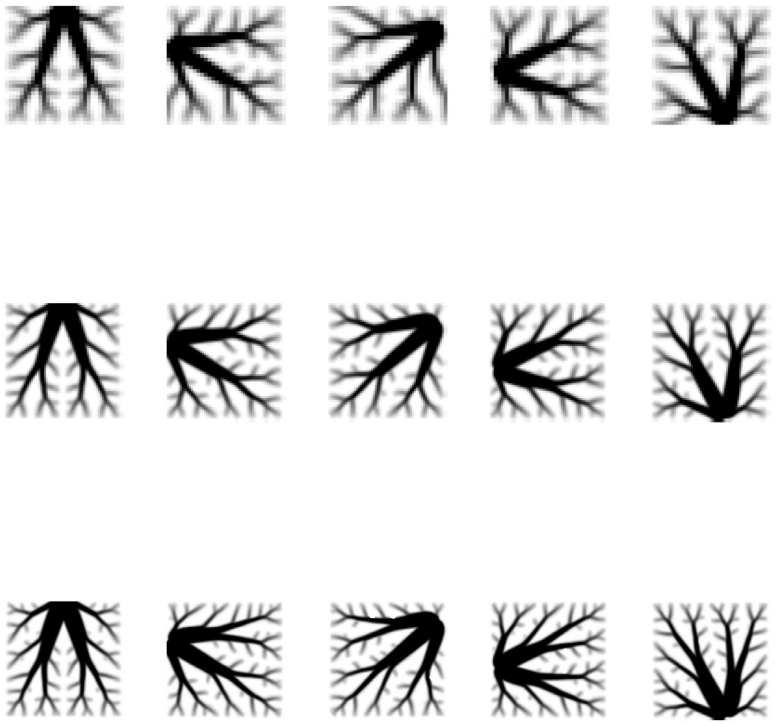

Fig. 21. Different samples of heat sink design in three resolutions: low 32 × 32 (top row), medium 64 × 64 (middle row) and high 128 × 128 (bottom row).

Table 9
L-bracket design: Computational time (in hours) for generating 10 000 high resolution heat sink designs.

Case/Method	DDNSM	SIMP
Computational time (with training)	21.11	33.88
Computational time (without training)	18.43	33.88

Table 10
Heat sink design: rate of success in number of iterations and compliance estimation, and average number of iterations for datasets generated using the random and importance sampling.

Case	RoS in iterations %	RoS in compliance %	Average no. of Iters.
Random	80.0	100	190.20
Importance sampling	88.0	100	186.62

shows the reconstructed images using two datasets and Fig. 25 shows the performance of these datasets in terms of iteration counts and compliance relative errors. It is observed that the performance of the selected samples in this example is slightly improved since the initial guesses are almost similar in both cases. We also expect that the prediction in this example is more challenging compared to the previous example as the optimal topologies in this case exhibit more details. This is evident from the number of polytopes used in these two examples: the first example uses $\mathcal{N} = 15$ polytopes while the second examples uses $\mathcal{N} = 25$ polytopes. The improvement in prediction is shown via numerics in Table 10 where the average number of iterations is improved from 190 to 186.

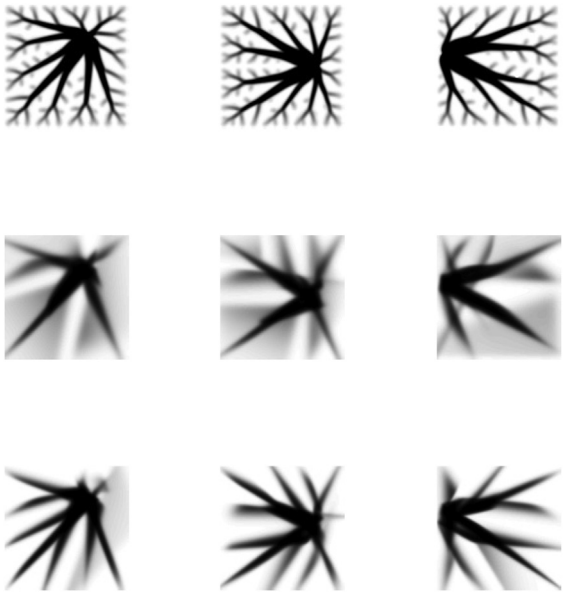

Fig. 22. Samples of heat sink design: actual high resolution (top row), reconstructed designs with dataset of Case 1 (middle row), and Case 4 (bottom row).

To investigate the effect of number of polytopes on the approximation more precisely we increase the number of polytopes in the DDNSM framework. In particular, we reconstruct the designs associated with Case 3 with $\mathcal{N} = 50,\ 100,\ 200$. Fig. 26

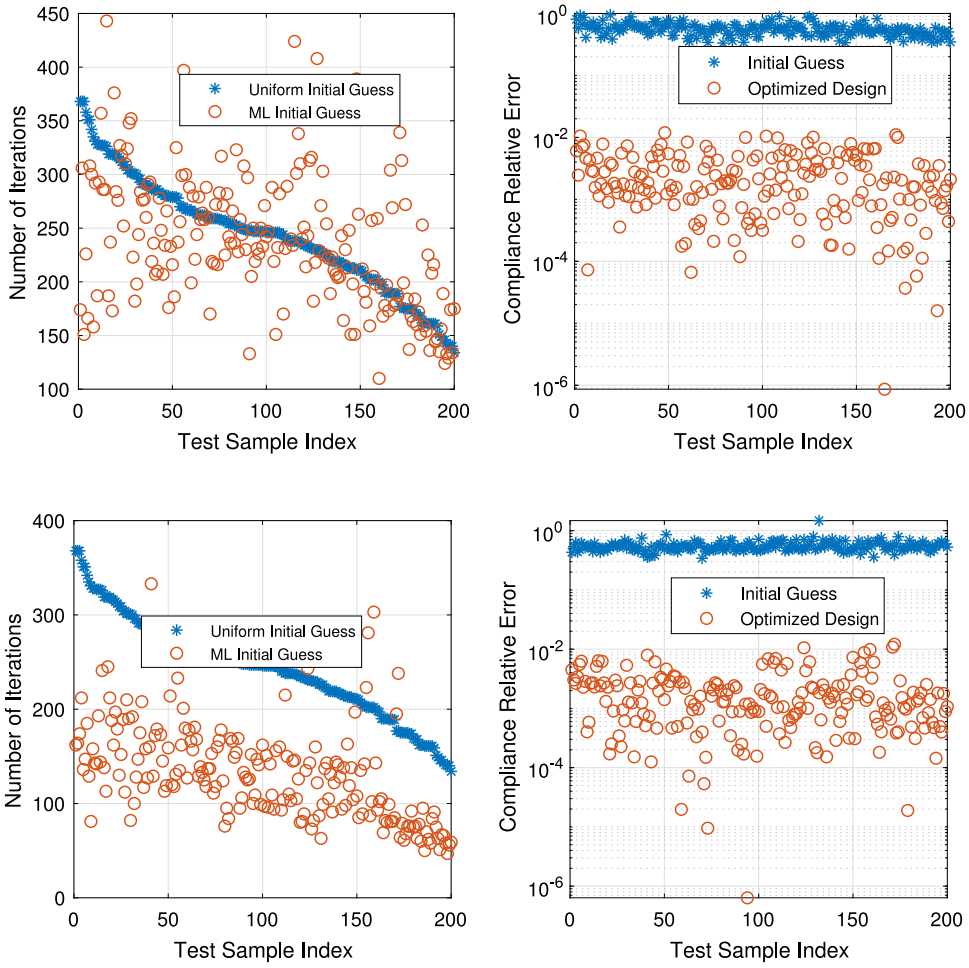

Fig. 23. Heat sink design: Iteration counts (left) and relative errors in compliance $e_{C_{ini}}, e_{C_{fin}}$ (right) for two cases: Case 1 (top row) and Case 4 (bottom row).

Table 11
Heat sink design: rate of success in number of iterations and compliance estimation, and average number of iterations for datasets generated using different number of polytopes: $\mathcal{N} = 25,\ 50,\ 100,\ 200$.

$\mathcal{N}$	RoS in iterations %	RoS in compliance %	Average no. of Iters.
25	94.0	100	152.43
50	93.5	100	158.95
100	92.0	100	157.29
200	86.0	100	179.36

shows an example of reconstructed high resolution designs corresponding to different number of polytopes and Table 11 lists the performance of different reconstructions in terms of the number of iterations and compliance estimation. From this experiment, it is observed that higher number of polytopes does not necessarily result in improved approximation. In particular we observe that increasing the number of polytopes does not improve the reconstruction of fine details of the heat sink design as it is evident from Fig. 26. We thus deem $\mathcal{N} = 25$ to be sufficient for our DDNSM approximation.

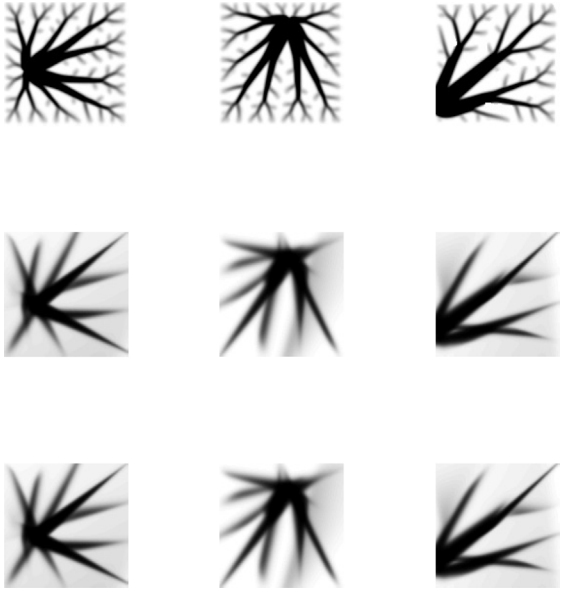

Fig. 24. Samples of heat sink design: actual high resolution (top row), reconstructed designs with dataset of Case 2 with random sampling (middle row), and Case 2 with importance sampling (bottom row).

4.3. 3D topology optimization

In this section we consider the problem of 3D topology optimization to show the effectiveness of our approach. It is well-known that there is a significant difference between 2D and 3D topology optimization problems in terms of cost. However, our DDNSM approach can be easily and systematically generalized to

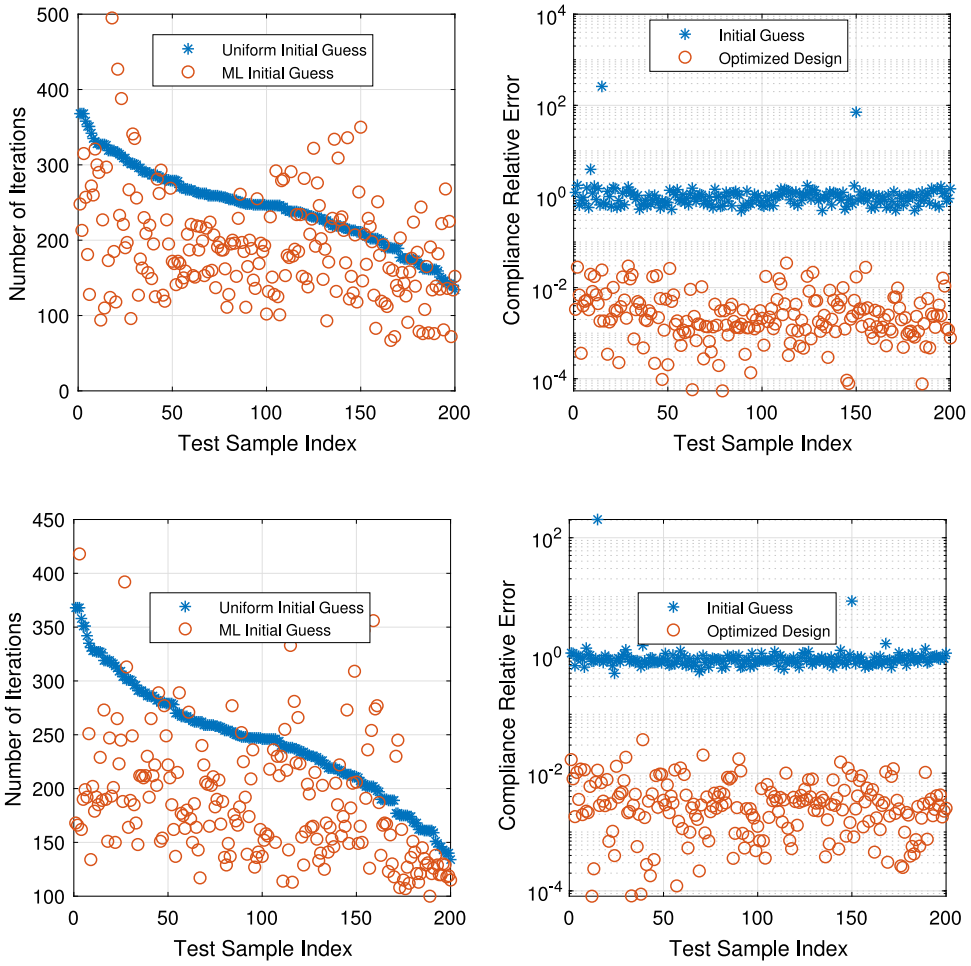

Fig. 25. Heat sink design: Iteration counts (left) and relative errors in compliance $e_{C_{ini}}$, $e_{C_{fin}}$ (right) for the dataset generated using the random (top) and importance sampling (bottom).

Fig. 26. An arbitrary reconstructed sample of heat sink design associated with number of polytopes $\mathcal{N} = 25,\ 50,\ 100,\ 200$ (from left to right).

3D images, and hence we use it as an effective tool to predict 3D optimized designs.

The design domain, boundary conditions and loading parameters are shown in Fig. 27. We make changes to the underlying code [79] for 3D topology optimization with varying parameters and use the visualization tool inside this code to visualize the optimized designs as well as the polytopes which are parts of our DDNSM approach. To generate training datasets, we consider two resolutions in this example: high and low resolutions. The high and low resolution FEAs consist of $40 \times 20 \times 4$ and $20 \times 10 \times 2$ brick elements with 12 915 and 2079 degrees of freedom, respectively. Similar to the first example, we set the Poisson's ratio ν to 0.3. We also consider the location of load p_1 as an integer number $p_1 \in [0, 20]$ for high resolution (similarly $p_1 \in [0, 10]$ for low resolution) and the angle of load as a uniformly distributed continuous variable $p_2 \in [0, \pi/2]$. The filter size for both resolutions are fixed $r_{min} = 1.3,\ 2.6$ and the volume

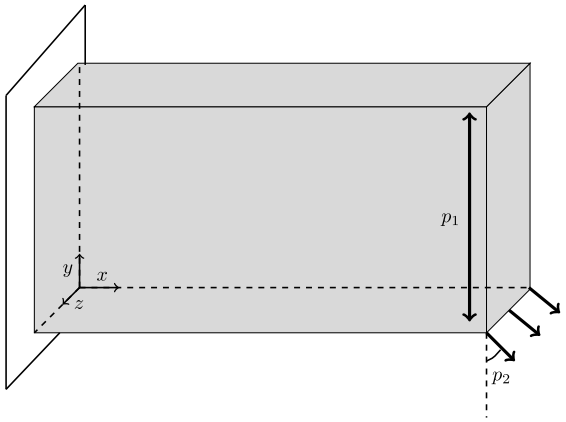

Fig. 27. Geometry, boundary conditions and load parameters for the 3D cantilever beam.

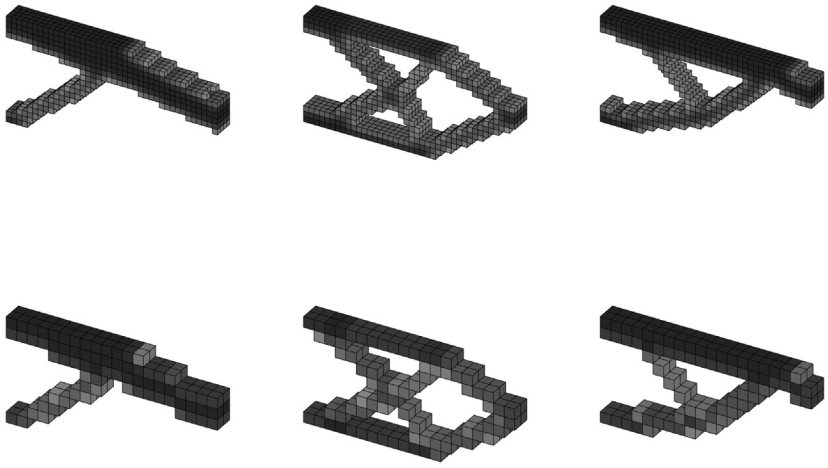

Fig. 28. 3D design: Different samples of 3D design in two resolutions: high $40 \times 20 \times 4$ (top row), and low $20 \times 10 \times 2$ (bottom row).

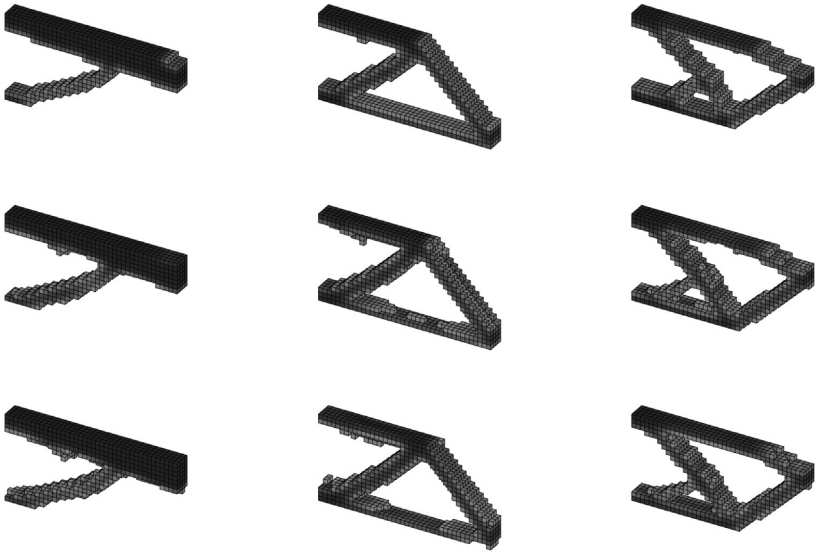

Fig. 29. Samples of 3D design: actual high resolution (top row), reconstructed designs with dataset of Case 3 (middle row), and Case 6 (bottom row).

fraction is $v = 0.35$. Similarly to the second example, we generate $n = 1000$ optimized designs for both resolutions and use 20% of the dataset for testing. Fig. 28 shows different samples of high and low resolutions.

In this example, we use the same DDNSM parameters and deep net architecture. I.e. the deep net consists of 8 fully connected layers with the number of nodes for 7 inner layers as $\{32, 256, 512, 1024, 2048, 2048, 1024\}$. There is a difference in the number of DNSM coefficients since the problem is 3D, therefore the number of nodes in the last layer is $15 \times 8 \times 4$. The learning rate is again set to $1e{-}4$, the number of gradient-descent iterations (epochs) is set to 250 and we set $\lambda = 1$ cf. Eq. (13).

We consider three cases associated with different training sizes as follows:

$$\begin{array}{lllll} \text{Case 1}: & \text{800 low resolution samples} & + & \text{80 high resolution samples} \\ \text{Case 2}: & \text{800 low resolution samples} & + & \text{160 high resolution samples} \\ \text{Case 3}: & \text{800 low resolution samples} & + & \text{240 high resolution samples} \end{array} \tag{29}$$

The first three cases above are experimented with random sampling. We consider additional three cases, Cases 4,5 and 6, with importance sampling. Before proceeding with numerical convergence study, it is beneficial to see the prediction of DDNSM visually. Fig. 29 shows three different optimized designs (within the test set) and their DDNSM predictions using datasets with random and importance sampling. It is interesting to see that the DDNSM approach in this case, i.e. the 3D designs, generates images that are appreciably close to the actual design. Indeed, the predicted designs are very close to the actual design such that they could be used without further processing. This statement is verified in the numerical results shown in Fig. 31 (right pane). As can be seen in this dataset, the initial guesses yield very close compliances to the actual compliances as the quantities for convergence of compliance with initial guesses are $e_{C_{ini}} \leq 0.1$ almost consistently throughout the test cases.

In this example, we also show the polytopes as the building block of images for better illustration of the approach. Fig. 30 shows an arbitrary optimized design and its $\mathcal{N} = 5$ and $\mathcal{N} = 15$

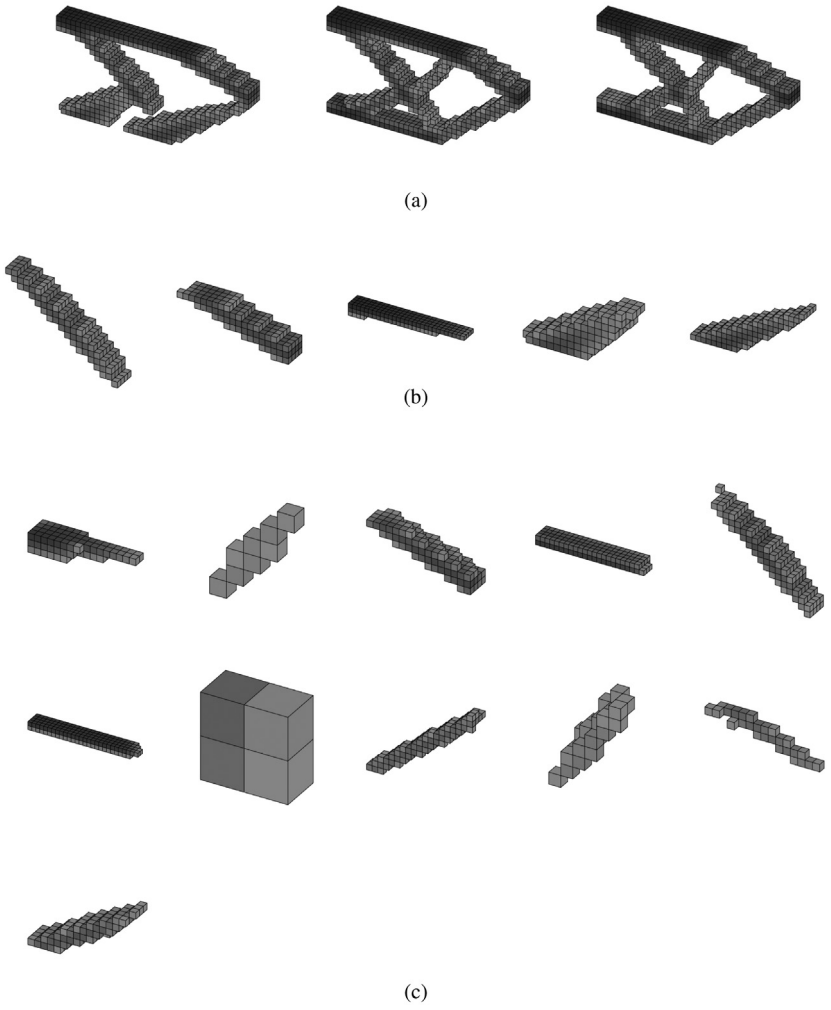

Fig. 30. 3D design: (a) Reconstructed design using $\mathcal{N} = 5$ and $\mathcal{N} = 15$ polytopes (left and middle) and the actual design (right), (b) $\mathcal{N} = 5$ and (c) $\mathcal{N} = 15$ associated polytopes obtained via DDNSM approach. Note that 4 out of 15 polytopes are obtained extremely small, hence they do not appear as images.

Table 12
3D design: rate of success in number of iterations and compliance estimation, and average number of iterations for different cases.

Case	RoS in iterations %	RoS in compliance %	Average no. of Iters.
1	85.5	92.0	81.82
2	87.5	95.0	77.64
3	87.0	88.0	79.65
5	84.5	83.5	79.44
4	85.5	84.0	78.55
6	83.5	83.0	76.28

polytopes. In some cases the identified polytopes are extremely small which are shown as blank. Nevertheless the union of the polytopes results in the reconstructed image. It is also observed that $\mathcal{N} = 5$ is not sufficient for reconstructing the actual design as the reconstructed image exhibits disconnected areas in the structure but $\mathcal{N} = 15$ results in almost identical image to the actual design.

Similar to previous examples, we numerically investigate the performance of each training case in terms of number of iterations and compliance estimation. Convergence results for six cases are listed in Table 12. The convergence results are shown via Fig. 31 for Cases 3 and 6, associated with random and importance sampling. Based on the results in Table 12, it is observed that the datasets with importance sampling perform slightly better in terms of number of iterations, however in both cases of random and importance sampling the DDNSM can noticeably reduce the iteration count which has significant impact on computational cost. As mentioned earlier, it is also observed that the initial guesses obtained with DDNSM in this 3D example, are very close to the actual optimized designs both visually and numerically (for compliance estimation).

In the next part we provide analysis of computational cost both in terms of function evaluation and computation time which again demonstrates the appreciable improvement provided by DDNSM in comparison with the standard topology optimization approach.

Analysis of Computational Cost

Similar to previous examples, we first investigate the performance of our DDNSM approach with respect to function evaluations. The average number of iterations for this 3D example is 152.08. Based on the results in Table 12 the number of iterations is reduced to almost half by using the initial guesses generated with DDNSM. In particular in Case 6, the average number of iterations is obtained as 76.28. The computational cost for generating this case is $C_c = (800 \times 1 + 240 \times 8) \times 152.08 = 4.1365e5$. Note

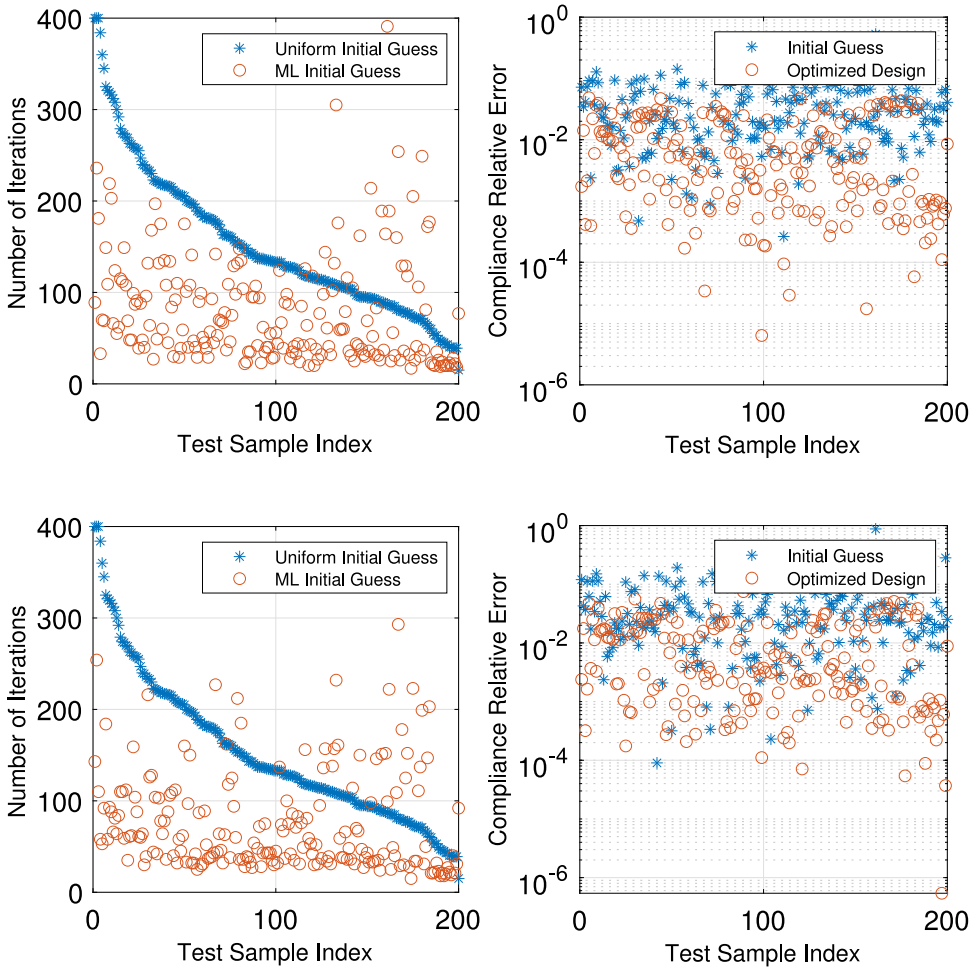

Fig. 31. 3D design: Iteration counts (left) and relative errors in compliance $e_{C_{ini}}$, $e_{C_{fin}}$ (right) for two cases: Case 3 (top row) and Case 6 (bottom row).

that in this case, performing one low resolution FEA is considered as a unit cost and since the problem is 3D, the high resolution cost (with mesh size halved) is $2^3 = 8$ times larger. Solving the following equation

$$4.1365e5 + (76.28 \times 8)n = (152.08 \times 8)n \tag{30}$$

yields $n = 85$. Based on this result, it is beneficial to use DDNSM strategy when one needs to find $n > 85$ high resolution 3D designs. This low number indeed shows the significant advantage of DDNSM surrogate over standard topology optimization approach (SIMP) for finding a large number of optimized designs. Next, we estimate the computational saving in terms of computational time.

In this example, the computational time for generating 10 000 low and high resolution designs are 11 091.35 and 331 213.11 s respectively. Note that generating a large number of high resolution designs in this case takes significant amount of time. The computational time for generating the dataset of Case 6, the network training, network evaluation and topology optimization with DDNSM initial guess for 10 000 samples are 8836.42, 3237.20, 19.30 and 166 129.24 s. Also note that in this example, training the network takes significantly less time compared to dataset generation. Once the network is trained, generating one design takes 0.002 s while direct 3D optimization takes 33 s. Table 13 lists the computational time for DDNSM and SIMP. Using DDNSM, the computational time has been reduced to almost half. It should be noted that, these estimations are obtained by considering the DDNSM-generated images as initial guesses to a subsequent topology optimization process. While in this example the predicted images are almost close to the optimized designs, they could be used without further processing. In that case, the computational time for finding 10 000 high resolution designs via DDNSM is obtained by only considering training and evaluation time. This amounts to 3.35 h as listed in the last row of the table which is indeed a significant reduction compared to the SIMP approach.

Table 13
3D design: Computational time (in hours) for generating 10 000 high resolution 3D topology optimized designs.

Case/Method	DDNSM	SIMP
Computational time (with training)	49.50	92.00
Computational time (without training)	46.15	92.00
Computational time (without post processing)	3.35	92.00

As an extension to this example, we consider a case where the loading is varying in the z direction. To this end we increase the number of elements in this direction to allow the topology optimization solver generates 3D designs that are non-symmetric in the z direction. In this way we can test the capability of the DDNSM approach to capture the internal structures within topology optimized designs. In particular, we consider high and low resolution meshes with $40 \times 20 \times 10$ and $20 \times 10 \times 5$ elements. The loading in the z direction is shown in Fig. 32.

Fig. 33 shows three different high and low resolution designs. It is apparent that there is considerable variation in the z direction

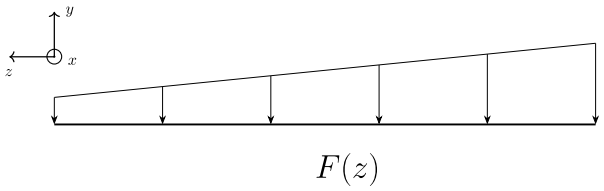

Fig. 32. 3D design: Non-uniform load in the z direction. The load varies linearly in the [0.5, 1.5] range.

Table 14
3D design: rate of success in number of iterations and compliance estimation, and average number of iterations for the optimized designs with loading shown in Fig. 32.

Case	RoS in iterations %	RoS in compliance %	Average no. of Iters.
1	88.0	91.0	94.21
2	86.5	88.5	97.05
3	87.5	90.0	87.59
5	85.0	87.5	100.66
4	86.5	91.5	90.67
6	88.5	90.5	89.01

in the optimized designs. Similarly to the previous experiment we consider six cases cf. Eq. (29) including both random and importance sampling.

Fig. 34 shows the prediction of DDNSM for Cases 3 and 6. It is observed that the DDNSM is capable of representing the 3D images and exhibiting internal structures in this experiment as well. We also notice the better performance of the importance sampling compared to the random sampling in the first predicted image. A part of the image has not been recovered by the random sampling however the importance sampling has yielded a close image to the actual one.

We also perform the iteration counts and compliance estimation for the six datasets. The results are listed in Table 14. The average number of iteration for this experiment (with larger number of elements) is 241.30. It is again observed that using DDNSM the number of iterations is considerably reduced.

To be precise, we again compute the total computational time for DDNSM and SIMP for a large number of high resolution designs which we consider as 1000 designs in this experiment. The computational time for generating the dataset with 1000 high and low resolution designs is 311 950.65 and 7391.85 s respectively. The required time for generating the dataset associated with e.g. Case 6 is 80 781.63 s. The training time for the neural network is 5628.81 and the time for evaluating the network for 1000 designs is 3.206 s. Again note that evaluating one design from the trained network only takes 0.003 s (versus 312 s or 5 min with direct 3D optimization in this case) which enables the utilization of this approach as an interactive tool for exploration of optimized designs.

Finally performing the subsequent topology optimization for predictions of Case 6 takes 115 071.39 s. The total computational times for both approaches are listed in Table 15 which again demonstrates the significant computational saving using DDNSM.

5. Conclusion

We presented a machine learning approach to generate high resolution topology designs from multiresolution images. Our approach used a novel scientific visualization method based on deep disjunctive normal shape model (DDNSM). The learning process involves simulation parameters and multiresolution images associated with each simulation parameter. Learning the map between the parameters and multiresolution images, the DDNSM generates near optimal high resolution designs on unknown simulation parameters. To overcome the computational challenges in training data we provided a systematic algorithm with an image-based quantitative measure to choose important high resolution designs which have more impact on the predictability of the learning approach. We showed our approach on three numerical examples associated with an L-shape elastic beam, a heat sink and a 3D linear elastic structure. We showed that the predicted high resolution designs can be effectively used as initial guesses for design scenarios where the simulation parameters are varying. In all cases we provided a computational cost analysis which serves as a guideline to assess the effectiveness of our machine learning approach compared to direct high resolution optimization. Via the computationally-intensive 3D topology optimization example, it is shown that the DDNSM predictive model reduces the computational time to almost half when compared with the standard topology optimization approach.

Some remarks on the limitations and advantages of our approach are in order:

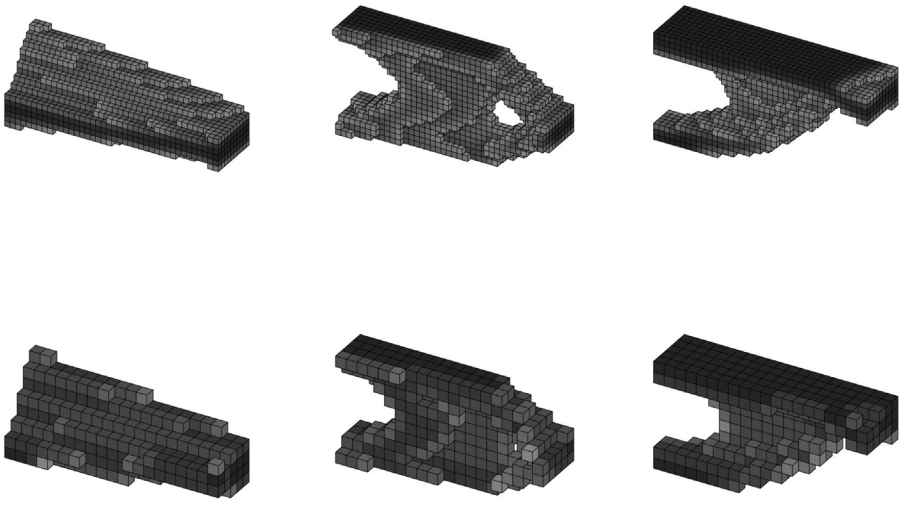

Fig. 33. 3D design: Different samples of 3D design in two resolutions: high 40 × 20 × 10 (top row), and low 20 × 10 × 5 (bottom row).

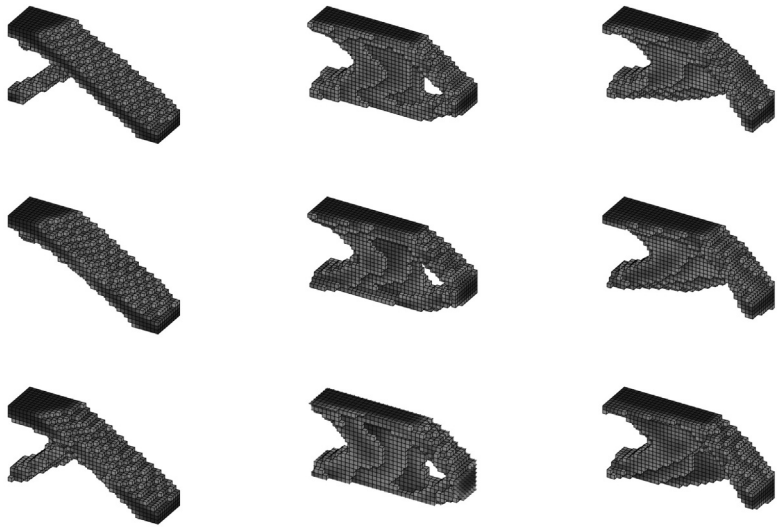

Fig. 34. Samples of 3D design with varying load in the z direction: actual high resolution (top row), reconstructed designs with dataset of Case 3 (middle row), and Case 6 (bottom row).

Table 15
3D design: Computational time (in hours) for generating 1000 high resolution 3D topology optimized designs associated with the loading shown in Fig. 32.

Case/Method	DDNSM	SIMP
Computational time	55.96	86.65

- Limitation: The success of our multiresolution approach is inherently dependent on the consistency of images across resolutions. The DDNSM approach interpolates between images that are provided to the net and if a particular high resolution image is not provided to the net, most likely images in the vicinity of that particular image cannot be successfully recovered. To alleviate this issue we have introduced a quantifiable measure (as a part of DDNSM algorithm) which effectively estimates the complexity of each image cf. Eq. (16). Images with small total overlap value exhibit finer features and are more likely to cover the entire high dimensional space of high-resolution images and therefore are suitable to be included in the training set.
- Limitation: The training cost is indeed significant in this paper as the approach takes optimized designs which themselves are the result of an optimization problem. Based on the results in the first numerical example where the single resolution DDNSM is compared with the multiresolution DDNSM, the single resolution DDNSM approach is more successful in recovering the fine features as all training points are high resolution images however that procedure is significantly costly compared to the multiresolution approach. The multiresolution approach is introduced to mitigate the computational burden of the training process. While not every fine feature is reproduced in multiresolution DDNSM, we have empirically found that the predicted images could be used as effective initial guesses to a subsequent topology optimization process.
- Limitation: The formulation of the loss function is based on the linear combination of losses in two resolutions with a balancing coefficient cf. Eq. (13). In general, the balancing coefficient is obtained from a cross-validation study similarly to many statistical analysis problems. We believe this is one pragmatic way to formulate this multiresolution problem which has yielded reasonable results however this formulation might not be the only way or the most mathematically appealing way to incorporate multiresolution data into the deep learning framework. Further investigation into the formulation of loss function or the architecture of DDNSM can potentially yield more improved predictions.
- Advantage: The method and its associated formulation can systematically accommodate multiresolution images. In DDNSM each image is decomposed to its geometric primitives (polytopes) and those geometric primitives can be found/computed in any resolution. Such geometric decomposition and multiresolution representation cannot be achieved with popular convolutional neural networks [14] which are extensively used in image processing.
- Advantage: The method can be equally effective in representing 3D images. The generalization from 2D to 3D (or even higher if needed) is trivial for DDNSM however there is a significant difference between 2D and 3D topology optimization problems in terms of both implementation and computational cost. We believe DDNSM is significantly advantageous for representing 3D images especially those images that are obtained with significant cost such as topology optimized designs.

Finally as future directions we plan to first develop a framework for topology optimization with physics informed neural networks (PINNs) [80] which could potentially address the fundamental issue of training cost in this paper. PINNs solve partial differential equations which describe physics problems via neural network machinery and can potentially be used for the topology optimization problem. As a next step, the PINNs framework for topology optimization can be integrated with DDNSM as a monolithic framework for image-based parametric topology optimization. Such framework, if achieved, will be free from finite element analysis and time consuming mesh generation. Another extension to this work is the incorporation of well-established geometric regularization steps in topology optimization such as heaviside projection method [81] into the DDNSM formulation to promote the recovery of sharp black–white designs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was sponsored by The Defense Advanced Research Projects Agency, USA TRADES Award HR0011-17-2-0016.

References

[1] Bendsøe M, Kikuchi N. Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Engrg 1988;71(2):197–224.
[2] Bendsøe M, Sigmund O. Topology optimization: Theory, methods and applications. Springer; 2003.
[3] Chen W, Fuge M, Chazan J. Design manifolds capture the intrinsic complexity and dimension of design spaces. J Mech Des 2017;139(5). (03).
[4] Burnap A, Liu Y, Pan Y, Lee H, Gonzalez R, Papalambros PY. Estimating and exploring the product form design space using deep generative models. In: ASME 2016 international design engineering technical conferences and computers and information in engineering conference.
[5] Torsney-Weir T, Saad A, Moller T, Hege H, Weber B, Verbavatz J, et al. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. IEEE Trans Vis Comput Graphics 2011;17(12):1892–901.
[6] Sedlmair M, Heinzl C, Bruckner S, Piringer H, Moller T. Visual parameter space analysis: A conceptual framework. IEEE Trans Vis Comput Graphics 2014;20(12):2161–70.
[7] Pajer S, Streit M, Torsney-Weir T, Spechtenhauser F, Möller T, Piringer H. Weightlifter: Visual weight space exploration for multi-criteria decision making. IEEE Trans Vis Comput Graphics 2017;23(1):611–20.
[8] Weber B, Greenan G, Prohaska S, Baum D, Hege HC, Muller-Reichert T, et al. Automated tracing of microtubules in electron tomograms of plastic embedded samples of Caenorhabditis elegans embryos. J Struct Biol 2012;178(2):129–38.
[9] Schultz T, Kindlmann GL. Open-box spectral clustering: Applications to medical image analysis. IEEE Trans Vis Comput Graphics 2013;19(12):2100–8.
[10] Unger A, Schulte S, Klemann V, Dransch D. A visual analysis concept for the validation of geoscientific simulation models. IEEE Trans Vis Comput Graphics 2012;18(12):2216–25.
[11] Waser J, Konev A, Sadransky B, Horvath Z, Ribicic H, Carnecky R, et al. Many plans: Multidimensional ensembles for visual decision support in flood management. Comput Graph Forum 2014;33(3):281–90.
[12] Konev A, Waser J, Sadransky B, Cornel D, Perdigão RAP, Horváth Z, et al. Run watchers: Automatic simulation-based decision support in flood management. IEEE Trans Vis Comput Graphics 2014;20(12):1873–82.
[13] Liu L, Zhou F, He Y. Automated status inspection of fastening bolts on freight trains using a machine vision approach. Proc Inst Mech Eng F 2016;230(7):1629–41.
[14] LeCun Y, Bengio Y, et al. Convolutional networks for images, speech, and time series. Handb Brain Theory Neural Netw 1995;3361(10):1995.
[15] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436.
[16] Schmidhuber J. Deep learning in neural networks: An overview. Neural Netw 2015;61:85–117.
[17] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012, p. 1097–105.
[18] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014, arXiv preprint arXiv:1409.1556.
[19] Ramesh N, Mesadi F, Cetin M, Tasdizen T. Disjunctive normal shape models. In: 2015 IEEE 12th international symposium on biomedical imaging (ISBI). 2015, p. 1535–9.
[20] Mesadi F, Erdil E, Cetin M, Tasdizen T. Image segmentation using disjunctive normal bayesian shape and appearance models. IEEE Trans Med Imaging 2017;37(1):293–305.
[21] Javanmardi M, Bigolin Lanfredi R, Cetin M, Tasdizen T. Image segmentation by deep learning of disjunctive normal shape model shape representation. In: The IEEE conference on computer vision and pattern recognition (CVPR) workshops. 2018.
[22] Liu J, Gaynor AT, Chen S, Kang Z, Suresh K, Takezawa A, et al. Current and future trends in topology optimization for additive manufacturing. Struct Multidiscip Optim 2018;57(6):2457–83.
[23] Martinez-Frutos J, Herrero-Perez D, Kessler M, Periago F. Risk-averse structural topology optimization under random fields using stochastic expansion methods. Comput Methods Appl Mech Engrg 2018;330:180–206.
[24] Keshavarzzadeh V, Fernandez F, Tortorelli DA. Topology optimization under uncertainty via non-intrusive polynomial chaos expansion. Comput Methods Appl Mech Engrg 2017;318:120–47.
[25] Keshavarzzadeh V, Meidani H, Tortorelli DA. Gradient based design optimization under uncertainty via stochastic expansion methods. Comput Methods Appl Mech Engrg 2016;306:47–76.
[26] De Gournay F, Allaire G, Jouve F. Shape and topology optimization of the robust compliance via the level set method. ESAIM: COCV 2008;1:43–70.
[27] Bendsøe M, Kikuchi N. Robust shape optimization of continuous structures via the level set method. Comput Methods Appl Mech Engrg 2016;305:271–91.
[28] Keshavarzzadeh V, Kirby RM, Narayan A. Parametric topology optimization with multiresolution finite element models. Internat J Numer Methods Engrg 2019;119(7):567–89.
[29] Bobby S, Spence SMJ, Kareem A. Data-driven performance-based topology optimization of uncertain wind-excited tall buildings. Struct Multidiscip Optim 2016;54(6):1379–402.
[30] Ulu E, Zhang R, Yumer ME, Kara LB. A data-driven investigation and estimation of optimal topologies under variable loading configurations. 2014, p. 387–99.
[31] Martinez-Frutos J, Martinez-Castejon PJ, Herrero-Perez D. Efficient topology optimization using gpu computing with multilevel granularity. Adv Eng Softw 2017;106:47–62.
[32] Martinez-Frutos J, Herrero-Perez D. Large-scale robust topology optimization using multi-gpu systems. Comput Methods Appl Mech Engrg 2016;311:393–414.
[33] Challis VJ, Roberts AP, Grotowski JF. High resolution topology optimization using graphics processing units (gpus). Struct Multidiscip Optim 2014;49(2):315–25.
[34] Jain AK, Duin RPW, Mao J. Statistical pattern recognition: a review. IEEE Trans Pattern Anal Mach Intell 2000;22(1):4–37.
[35] van der Maaten L, Hinton G. Visualizing data using t-sne. J Mach Learn Res 2008;9(85):2579–605.
[36] Jordan MI, Mitchell TM. Machine learning: Trends, perspectives, and prospects. Science 2015;349(6245):255–60.
[37] Raissi M, Perdikaris P, Karniadakis GE. Machine learning of linear differential equations using gaussian processes. J Comput Phys 2017;348:683–93.
[38] Raissi M, Perdikaris P, Karniadakis G. Numerical gaussian processes for time-dependent and nonlinear partial differential equations. SIAM J Sci Comput 2018;40(1):A172–98.
[39] Yu Y, Hur T, Jung J. Deep learning for topology optimization design, Korea Advanced Atomic Research Institute. arXiv preprint: https://arxiv.org/pdf/1801.05463.pdf.
[40] Oh S, Jung Y, Kim S, Lee I, Kang N. Deep generative design: Integration of topology optimization and generative models. J Mech Des 2019.
[41] Li B, Huang C, Li X, Zheng S, Hong J. Non-iterative structural topology optimization using deep learning. Comput Aided Des 2019;115:172–80.
[42] Sosnovik I, Oseledets I. Neural networks for topology optimization, University of Amsterdam, Amsterdam, The Netherlands; Skolkovo Institute of Science and Technology, Moscow, Russia; Institute of Numerical Mathematics RAS, Moscow, Russia. arXiv preprint: https://arxiv.org/abs/1709.09578.
[43] Cang R, Yao H, Ren Y. One-shot generation of near-optimal topology through theory-driven machine learning. Comput Aided Des 2019;109:12–21.
[44] Gupta DK, van Keulen F, Langelaar M. Design and analysis adaptivity in multiresolution topology optimization. Int J Numer Methods Eng https://doi.org/10.1002/nme.6217.
[45] Bandara K, Rüberg T, Cirak F. Shape optimisation with multiresolution subdivision surfaces and immersed finite elements. Comput Methods Appl Mech Engrg 2016;300:510–39.
[46] Nguyen TH, Paulino GH, Song J, Le CH. A computational paradigm for multiresolution topology optimization (mtop). Struct Multidiscip Optim 2018;41:525–39.
[47] Lieu QX, Lee J. Multiresolution topology optimization using isogeometric analysis. Internat J Numer Methods Engrg 2017;112(13):2025–47.
[48] Bandara K, Cirak F, Of G, Steinbach O, Zapletal J. Boundary element based multiresolution shape optimisation in electrostatics. J Comput Phys 2015;297:584–98.
[49] Kim YY, Yoon GH. Multi-resolution multi-scale topology optimization — a new paradigm. Int J Solids Struct 2000;37(39):5529–59.
[50] Filipov ET, Chun J, Paulino GH, Song J. Polygonal multiresolution topology optimization (polymtop) for structural dynamics. Struct Multidiscip Optim 2016;53:673–94.
[51] Lieu QX, Lee J. A multi-resolution approach for multi-material topology optimization based on isogeometric analysis. Comput Methods Appl Mech Engrg 2017;323:272–302.
[52] Groen JP, Langelaar M, Sigmund O, Ruess M. Higher-order multi-resolution topology optimization using the finite cell method. Internat J Numer Methods Engrg 2017;110(10):903–20.
[53] De S, Maute K, Doostan A. Bi-fidelity stochastic gradient descent for structural optimization under uncertainty. 2019, arXiv:1911.10420 [math.OC].
[54] Zhang W, Zhao L, Gao T, Cai S. Topology optimization with closed b-splines and boolean operations. Comput Methods Appl Mech Engrg 2017;315:652–70.

[55] Qian Z, Ananthasuresh GK. Optimal embedding of rigid objects in the topology design of structures. Mech Based Des Struct Mach 2004;32(2):165–93.
[56] Chen J, Shapiro V, Suresh K, Tsukanov I. Shape optimization with topological changes and parametric control. Internat J Numer Methods Engrg 2007;71(3):313–46.
[57] Xia L, Zhu J, Zhang W. Sensitivity analysis with the modified heaviside function for the optimal layout design of multi-component systems. Comput Methods Appl Mech Engrg 2012;241–244:142–54.
[58] Zhang W, Xia L, Zhu J, Zhang Q. Some recent advances in the integrated layout design of multicomponent systems. J Mech Des 2011;133(10).
[59] Zhang W, Zhong W, Guo X. Explicit layout control in optimal design of structural systems with multiple embedding components. Comput Methods Appl Mech Engrg 2015;290:290–313.
[60] Zhu J, Zhang W, Beckers P, Chen Y, Guo Z. Simultaneous design of components layout and supporting structures using coupled shape and topology optimization technique. Struct Multidiscip Optim 2008;36:29–41.
[61] Zhou M, Wang MY. Engineering feature design for level set based structural optimization. Comput Aided Des 2013;45(12):1524–37.
[62] Guo X, Zhang W, Zhong W. Doing topology optimization explicitly and geometrically—A new moving morphable components based framework. J Appl Mech 2014;81(8):081009.
[63] Guo X, Zhang W, Zhang J, Yuan J. Explicit structural topology optimization based on moving morphable components (mmc) with curved skeletons. Comput Methods Appl Mech Engrg 2016;310:711–48.
[64] Zhang W, Yuan J, Zhang J, Guo X. A new topology optimization approach based on moving morphable components (mmc) and the ersatz material model. Struct Multidiscip Optim 2016;53(6):1243–60.
[65] Zhang W, Li D, Yuan J, Song J, Guo X. A new three-dimensional topology optimization method based on moving morphable components (mmcs). Comput Mech 2017;59:647–65.
[66] Bell B, Norato J, Tortorelli D. A geometry projection method for continuum-based topology optimization of structures, arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2012-5485.
[67] Norato J, Bell B, Tortorelli D. A geometry projection method for continuum-based topology optimization with discrete elements. Comput Methods Appl Mech Engrg 2015;293:306–27.
[68] Zhang S, Norato JA, Gain AL, Lyu N. A geometry projection method for the topology optimization of plate structures. Struct Multidiscip Optim 2016;54:1173–90.
[69] Norato JA. Topology optimization with supershapes. Struct Multidiscip Optim 2018;58:415–34.
[70] Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O. Efficient topology optimization in matlab using 88 lines of code. Struct Multidiscip Optim 2011;43(1):1–16.
[71] Sigmund O. On the usefulness of non-gradient approaches in topology optimization. Struct Multidiscip Optim 2011;43:589–96.
[72] Bruns T, Tortorelli D. Topology optimization of non-linear elastic structures and compliant mechanisms. Comput Methods Appl Mech Engrg 2001;190(26–27):3443–59.
[73] Bendsøe M. Optimal shape design as a material distribution problem. Struct Optim 1989;1(4):193–202.
[74] Bendsøe M, Sigmund O. Material interpolation schemes in topology optimization. Arch Appl Mech 1999;69(9):635–54.
[75] Hazewinkel M. Encyclopaedia of mathematics: C an updated and annotated translation of the soviet mathematical encyclopaedia, Vol. 2. Springer Science Business Media; 2013.
[76] Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: a survey. 2018, arXiv:1502.05767 [cs.SC].
[77] Le C, Norato J, Bruns T, Ha C, Tortorelli D. Stress-based topology optimization for continua. Struct Multidiscip Optim 2010;41(4):605–20.
[78] James KA, Waisman H. Failure mitigation in optimal topology design using a coupled nonlinear continuum damage model. Comput Methods Appl Mech Engrg 2014;268:614–31.
[79] Liu Kai, Tovar Andres. An efficient 3d topology optimization code written in MATLAB. Struct Multidiscip Optim 2014;50:1175–96.
[80] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[81] Guest J, Prevost JH, Belytschko T. Achieving minimum length scale in topology optimization using nodal design variables and projection functions. Internat J Numer Methods Engrg 2004;61(2):238–54.