Neural topology optimization
the good, the bad, and the ugly

Sanu, Suryanarayanan Manoj; Aragón, Alejandro M.; Bessa, Miguel A.

DOI
10.1007/s00158-025-04135-3

Publication date
2025

Document Version
Final published version

Published in
Structural and Multidisciplinary Optimization

Structural and Multidisciplinary Optimization (2025) 68:213
https://doi.org/10.1007/s00158-025-04135-3

RESEARCH PAPER

Neural topology optimization: the good, the bad, and the ugly

Suryanarayanan Manoj Sanu[1] · **Alejandro M. Aragón**[1] · **Miguel A. Bessa**[2]

Received: 11 February 2025 / Revised: 21 August 2025 / Accepted: 26 August 2025 / Published online: 4 October 2025

Abstract

Neural networks (NNs) hold great promise for advancing inverse design via topology optimization (TO), yet misconceptions about their application persist. This article focuses on neural topology optimization (neural TO), which leverages NNs to reparameterize the decision space and reshape the optimization landscape. While the method is still in its infancy, our analysis tools reveal critical insights into the NNs' impact on the optimization process. We demonstrate that the choice of NN architecture significantly influences the objective landscape and the optimizer's path to an optimum. Notably, NNs introduce non-convexities even in otherwise convex landscapes, potentially delaying convergence in convex problems but enhancing exploration for non-convex problems. This analysis lays the groundwork for future advancements by highlighting: (1) the potential of neural TO for non-convex problems and dedicated GPU hardware (the "good"), (2) the limitations in smooth landscapes (the "bad"), and (3) the complex challenge of selecting optimal NN architectures and hyperparameters for superior performance (the "ugly").

Keywords Topology optimization · Machine learning · Neural reparameterization · Implicit biases · Loss and objective landscapes visualization · Optimization trajectories

Finding an effective representation of the decision space is crucial in optimization, as a change of basis can transform a challenging problem into a more solvable one. Nature offers unparalleled examples of efficient representations—DNA, for instance, encodes the complexity of trillions of brain connections using just 30,000 genes (Stanley 2007). Inspired by such abstractions, this study investigates reparameterizing the design space using neural networks. Our goal is to evaluate whether this strategy, as suggested in the literature, can accelerate convergence or yield superior optima.

Responsible Editor: Joe Alexandersen.

Topical Collection: ICTAM 2024 - The centennial of ICTAM Guest Editors: J. Alexandersen, A.M. Aragón, X.S. Zhang, E. Lund, E. Wadbro, J. Kook

✉ Alejandro M. Aragón
a.m.aragon@tudelft.nl

✉ Miguel A. Bessa
miguel_bessa@brown.edu

[1] Faculty of Mechanical Engineering, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

[2] School of Engineering, Brown University, 184 Hope Street, Providence RI 02912, Rhode Island, United States

Topology optimization (TO) (Bendsøe and Kikuchi 1988) has been applied across a wide range of problems in science and engineering. It has proven effective for optimizing material layouts in complex nonlinear (Xia et al 2018; Buhl et al 2000; Fritzen et al 2016; Bendsøe et al 1996; Nakshatrala et al 2013; Chen et al 2018), transient (Min et al 1999; Le et al 2012; Yoon 2022), and multi-physics problems involving coupled differential equations (Silva and Kikuchi 1999; Yoon et al 2006; Kreissl et al 2011; Allen and Maute 2005; Yoon et al 2018; Yu et al 2019). Notably, TO has pushed design resolution boundaries to giga-scale for compliance minimization (Aage et al 2017).

Despite these successes, standard density-based TO has intrinsic limitations. First, fine mesh resolutions can lead to high computational costs. To mitigate this, researchers have explored reduced-order models (Kirsch and Papalambros 2001; Xia and Breitkopf 2014), conventional surrogate models (Raponi et al 2019; Yoshimura et al 2016), and machine learning approaches (Woldseth et al 2022). However, these methods still face challenges as TO typically relies on first-order optimizers, which may require dozens—or even hundreds—of iterations to converge. Second, TO's gradient-based search often leads to local optima, making it less effective for non-convex problems. Strategies to address this include deflation, where the objective function is modified

to steer the optimizer away from previously found solutions (Papadopoulos et al 2021), and continuation schemes, which gradually modify the convexity of the problem in hopes of escaping suboptimal local minima (Rozvany 2009). Finally, because dimensionality of the design space is tied to the mesh resolution, the scale of the optimization problem can grow arbitrarily large, leading to significant computational burden and limiting the range of feasible optimization algorithms. Thus, the way the problem is parameterized plays a critical role in overcoming these challenges.

The standard density-based parameterization, which assigns one decision variable per finite element, remains the most widely used technique in topology optimization (TO). This approach is often combined with penalization methods, such as the solid isotropic material with penalization (SIMP) (Bendsøe and Sigmund 2004) or the rational approximation of material properties (RAMP) (Stolpe and Svanberg 2001). However, alternative parameterization techniques have gained traction in recent years. For instance, geometric projection methods (Zhang et al 2016) and moving morphable components (Guo et al 2014) use simple geometric primitives as building blocks, drastically reducing the number of design variables needed to represent a structure. These reparameterization strategies decouple geometry from analysis, creating a design space that does not scale with resolution. Other reparameterization approaches explored in TO include Fourier coefficients (White et al 2018, wavelets (Poulsen 2002), B-splines (Qian 2013), element connectivities (Yoon and Kim 2005), distance fields(Biswas et al 2004), and radial basis functions (Wang and Wang 2005. While these techniques offer advantages, some lead to constrained design spaces with limited expressivity, potentially hindering the discovery of optimal structures for a given resolution. This limitation highlights the need for a general class of expressive and differentiable reparameterization strategies that can be adapted to specific optimization problems.

Neural networks (NNs) offer a compelling alternative for reparameterizing topology optimization. Unlike other reparameterization methods, NNs are highly abstract and expressive (Hornik et al 1989). Moreover, their empirical success in diverse fields, despite being optimized with simple gradient-based algorithms starting from random initialization (Danilova et al 2022), suggests they can offer a promising design space. In TO, however, NNs have primarily been used for supervised learning tasks. These include using NNs as surrogates to enhance computational efficiency—by replacing analytical sensitivities or finite element analysis (Chi et al 2021; Qian and Ye 2020)—or to predict near-optimal structures without optimization iterations (Kallioras et al 2020; Kallioras and Lagaros 2020; Xue et al 2021; Sosnovik and Oseledets 2019; Banga et al 2018). Such supervised approaches require large training datasets and have limited extrapolation capabilities i.e., they perform poorly when encountering inputs outside the training set (Woldseth et al 2022). Consequently, their use has been discouraged. In contrast, using NNs for reparameterization without supervised training—relying instead on unsupervised learning—provides an alternative approach. This strategy operates at the intersection of TO and machine learning, requiring no training data while leveraging the flexibility of NNs. Despite its potential, neural reparameterization in TO is still a nascent field.

The first study in neural reparameterization for TO, conducted by Hoyer et al (2019), utilized convolutional neural networks (CNNs). This choice was motivated by the resemblance of grid-based structures to images and the efficiency of CNNs in image processing. They reported a slight performance advantage of neural reparameterization over standard density-based TO for compliance-minimization problems. Similarly, Zhang et al (2021) adopted a CNN-based architecture and tested it on various problems, including stress-constrained compliance optimization, structural natural frequency optimization, compliant mechanism design, optimal heat conduction, and hyperelastic structure optimization. Their findings showed comparable performance to conventional TO. Departing from CNN architectures, Chandrasekhar and Suresh (2020) and Chandrasekhar and Suresh (2022) proposed fully connected neural networks to represent the density field as a continuous function of Cartesian coordinates, decoupling the geometric representation from the finite element mesh. Other studies (Deng and To 2020; Doosti et al 2021; Halle et al 2021; Zehnder et al 2021) explored similar architectures, experimenting with variations in filtering schemes, hyperparameters, and network designs. Despite these efforts, neural reparameterization remains a black box, and its utility is still uncertain. Critical aspects, such as the influence of neural architectures on optimization dynamics and their correlation with final performance, have not been thoroughly investigated. Furthermore, the lack of adherence to standard benchmarking practices for TO (Sigmund 2022) makes it challenging to assess whether reported improvements are meaningful.

This article focuses on characterizing and explaining the effects of NN reparameterization in topology optimization, which we name as "neural topology optimization" (neural TO). As a baseline for comparison, we use density-based TO with the SIMP method (Bendsøe and Kikuchi 1988) and the method of moving asymptotes (MMA) as the optimizer (Svanberg 1987). Note that other TO methodologies could be used as baseline, but the point of this work is not to claim superiority of neural TO over other TO strategies. Instead, we focus on providing the tools to effectively demonstrate both the positive and negative effects of neural TO, guiding the community toward meaningful future developments.

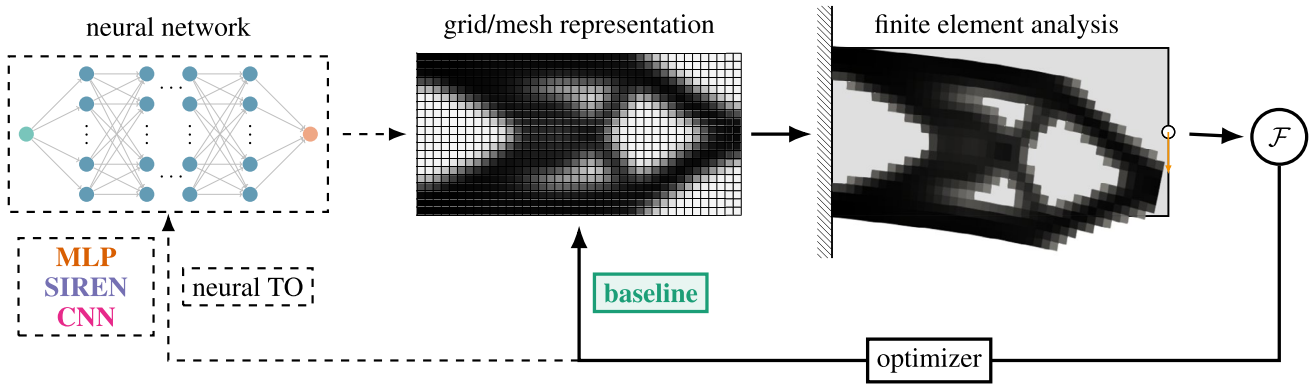


Fig. 1 Schematic of neural topology optimization (TO). Unlike standard density-based TO (baseline), an NN outputs the physical densities $\boldsymbol{\rho}$ (within the bounds [0, 1]), on which finite element analysis is performed to obtain the objective. The network parameters are updated through an optimizer to indirectly alter the density field. For the baseline, the individual "pixels" are the decision variables, while for the network, trainable parameters form the decision space. Depending on the network architecture, the output can either be the complete density field or the density at a specific location in the design domain. In the latter case, the network represents a continuous field (see Sec. B of the appendix for details about network architectures)

1 Formulation

Formally, the typical TO problem of minimizing structural compliance via a density-based method is formulated as follows (Sigmund and Maute 2013):

$$\begin{aligned} \boldsymbol{\rho}^\star = \underset{\boldsymbol{\rho}\in\mathcal{D}}{\arg\min} \quad & \mathcal{F}(\boldsymbol{U}(\boldsymbol{\rho}),\boldsymbol{\rho}) = \boldsymbol{U}^\intercal \boldsymbol{F} \equiv c, \\ \text{such that} \quad & \mathrm{g}_0(\boldsymbol{\rho}) = V = \textstyle\sum_{i=1}^{N} v_i\rho_i \leq V_0, \\ & \boldsymbol{K}\boldsymbol{U} = \boldsymbol{F}, \\ & 0 \leq \rho_i \leq 1, \quad i = \{1,\dots,N\}, \end{aligned} \tag{1}$$

where $\boldsymbol{\rho} \in \mathbb{R}^N$ is a vector of decision variables (physical density field) taken from the design space $\mathcal{D} = [0,1]^N$, N is the total number of finite elements; $\boldsymbol{U}$ is the displacement vector obtained by finite element analysis, i.e., after solving the linear system of equations $\boldsymbol{K}\boldsymbol{U} = \boldsymbol{F}$, with $\boldsymbol{K}$ and $\boldsymbol{F}$ denoting the global stiffness matrix and global force vector, respectively; c is the objective function, which gives a measure of compliance, and g_0 is the volume of material that needs to satisfy the inequality constraint, with V_0 denoting the maximum allowed volume. The subscript i denotes the corresponding element-wise quantities, and thus v_i denotes the volume of the ith element and ρ_i its density. Often, the SIMP law (Bendsøe and Sigmund 2004) is used to penalize intermediate density values (with penalty $p = 3.0$) to push the optimization toward a 0 and 1 (black-and-white) design. A density filter is usually applied to prevent the formation of artificially stiff checkerboard patterns and to enforce a mesh-independent length-scale. The method was re-implemented based on the 88-line Matlab implementation (Andreassen et al 2011) and is considered as the *baseline* in all experiments (Fig. 1).

1.1 Neural topology optimization

We define *reparameterization* as any approach where the physical density field is expressed as the output of a function h, i.e., $\boldsymbol{\rho} = h(\boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \mathbb{R}^M$ are the new decision variables[1]. As a result of reparameterization, the minimized objective and the constraint are no longer $\mathcal{F}(\boldsymbol{\rho})$ and $g_0(\boldsymbol{\rho})$ but the composition functions $\mathcal{F}\circ h(\boldsymbol{\theta})$ and $g_0\circ h(\boldsymbol{\theta})$, respectively. Therefore, when an optimizer is used to update $\boldsymbol{\theta}$, the physical density field is altered indirectly. Representing the reparameterization in this way allows for more flexibility in tailoring the decision space, as it can be both over-parameterized ($M > N$) or under-parameterized ($M < N$) when compared to the number of design parameters used in the baseline. Therefore, reparameterization decouples the finite element discretization and the representation of the density field.

Introducing a neural network (NN) to define the density distribution in a finite element mesh results in a method we refer to as "neural topology optimization" (Fig. 1). This reparameterization transforms the decision space into $\widetilde{\mathcal{D}}$, where each parameter becomes unbounded, i.e., $-\infty \leq \theta_i \leq +\infty$. NNs are parametric universal function approximators (Hornik et al 1989), in which the parameters are arranged into layers with one input layer and output layers and several hidden layers in between. The ith layer carries out a mathematical operation of the form $\sigma\left(\boldsymbol{W}_i z_i + \boldsymbol{b}_i\right)$, where z_i is the input, $\boldsymbol{W}_i$ is a matrix of weights, $\boldsymbol{b}_i$ the corresponding bias vector, and σ is a nonlinear activation

[1] Note that even filters used in density-based TO can be regarded as reparameterizing the physical densities.

function. For a network of L layers, i.e., $L-2$ hidden layers, the density is calculated recursively as

$$\boldsymbol{\rho} = \sigma\big(\boldsymbol{W}_L\sigma\big(\cdots\sigma\big(\boldsymbol{W}_2\sigma\big(\boldsymbol{W}_1\boldsymbol{z}_1+\boldsymbol{b}_1\big)\quad+\boldsymbol{b}_2\big)\cdots\big)+\boldsymbol{b}_L\big). \tag{2}$$

The reparameterization function h is represented as a NN with trainable parameters $\boldsymbol{\theta} = \{\boldsymbol{W}_1 \ldots \boldsymbol{W}_L, \boldsymbol{b}_1 \ldots \boldsymbol{b}_L\}$, i.e., $\boldsymbol{\rho} = h(\boldsymbol{\theta}, \boldsymbol{z}_1)$. Because the reparameterized decision variables are unbounded, a transformation is necessary to map the network outputs so that $\rho_i \in [0,1]$ for material interpolation and further analysis. In this paper, we chose two approaches: (1) apply a sigmoid function to the network's outputs, or (2) use a shifted-sigmoid "layer" (Hoyer et al 2019) ($\{h(\boldsymbol{\theta}) \mid 0 \le h(\boldsymbol{\theta})_i \le 1,\ g_0(h((\theta))) = V_0\}$), where the shift parameter is determined with a bisection algorithm to strictly enforce the volume constraint as well (see appendix B.1)[2]. Thus, the former strategy still requires a constrained optimizer similar to the standard approach (e.g., MMA, Svanberg (2002)) while the latter results in an unconstrained optimization, which can be solved by the common optimizers in machine learning (e.g., Adam, Kingma and Ba (2015)). Although many optimizers are available in machine learning, Adam remains the most widely used. A thorough benchmark study (Schmidt et al 2021) has shown that a well-tuned Adam optimizer performs strongly across a wide range of tasks. Consequently, Adam was chosen for our study.

2 Experiment to illustrate the effect of reparameterization

Start by considering the 2-D stress-constrained truss optimization problem (Stolpe 2003; Kirsch 1990), shown schematically in Fig. 2a. In this problem with an applied unit load in the middle node, the objective is to minimize the mass of the structure with constraints on the stresses of the bars. The decision variables are the areas of the two bars A_1 and A_2. The mathematical formulation of this problem is included in Sec. A of the appendix.

This seemingly simple problem has a two-dimensional decision space, which is shown in Fig. 2b. The figure highlights the feasible region (white), the constraints (red and orange), and local and global optima labeled as $\boldsymbol{\rho}_l$ and $\boldsymbol{\rho}_g$, respectively[3]. As apparent from the figure, the design space for this problem is degenerate in the sense that the global minimum $\boldsymbol{\rho}_g = (1,0)$ can be reached from the feasible set only along the line $A_2 = 0$. Conversely, the local optimum, located at $\boldsymbol{\rho}_l = (0,1)$ can be reached more easily from the feasible set. It is therefore extremely challenging to find the global minimum using gradient-based optimization. This is shown by the typical trajectory followed by the MMA optimizer from a feasible starting point $\boldsymbol{\rho}_0 = (1,1)$ (see baseline trajectory in green).

To explore the effect of reparameterization, we use the smallest NN architecture consisting of only three weights (θ_1, θ_2, and θ_3) from a four-neuron setup as shown in Fig. 2c. We remove the network's bias parameters (to facilitate visualization) and nonlinearly transform the original two-dimensional decision space (A_1 and A_2) into a three-dimensional space. As a result, we show that the same optimizer (MMA) is able to reach the global optimum $\boldsymbol{\rho}_g$ from the same starting point, following the trajectory in Fig. 2d. A two-dimensional projection of this trajectory is also plotted in Fig. 2b for reference, which shows that the optimizer accesses the global optimum through the linear subspace. Thus, a well-chosen reparameterization reshapes the decision space, making new trajectories possible for the optimizer to reach the otherwise inaccessible global optimum. In this example, two factors facilitate this access. First, in Fig. 2d we see surfaces corresponding to the constraints that are periodically repeated due to the harmonic activation function chosen. As a result, there is an infinite number of global optima. However, this alone does not explain how the optimizer accesses the linear subspace (which remains degenerate, as we show in Sec. A of the appendix). The inset in Fig. 2d provides a closer view of the path to the optimum, sliced by three orthogonal planes intersecting the found global optimum (Fig. 2e). The feasible regions (white) in these planes indicate that the access path "opens up". However, this effect resembles constraint relaxation (Verbart et al 2016; Duysinx and Bendsøe 1998; Cheng and Jiang 1992), where even if the degeneracy persists, it is surrounded by regions with extremely low constraint violations, appearing nearly feasible. Additional analytical and empirical details are provided in Sec. A of the appendix.

This example illustrates the positive impact of neural reparameterization on non-trivial objective landscapes. However, it does not address the challenges of identifying such beneficial network architectures, nor does it determine whether this approach can be advantageous for high dimensional continuum TO problems such as standard compliance optimization. Additionally, practical neural networks often have orders of magnitude more parameters, complicating the analysis. Therefore, the remaining of the article will discuss a three step analysis strategy to understand the effects of NN choices in the context of neural TO, namely: 1) visualizing objective landscapes; 2) analyzing optimizer trajectories; and 3) quantifying the expressivity of NNs. We select two conventional structural compliance TO problems, namely the tensile and the Michell beam cases (see Fig. 13

[2] Note that all these operations (including the density filter) that result in physical densities are absorbed into the definition of the neural network function h.

[3] Without constraints, the problem has a trivial global optimum at (0, 0). Therefore we refer to the global optimum of the constrained problem.

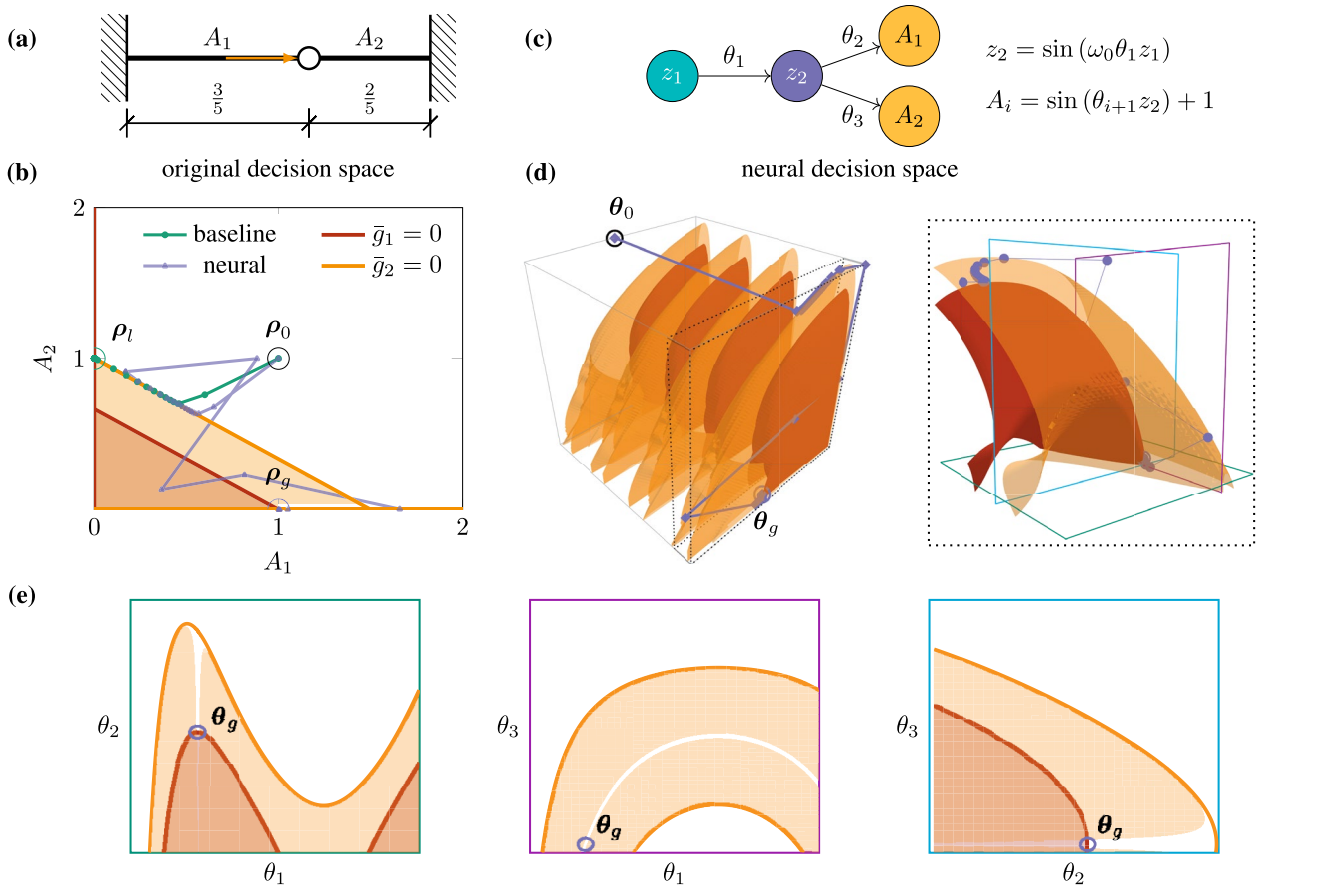


Fig. 2 Two-bar problem with stress constraints optimized using MMA, both with and without neural reparameterization: **a** Schematic showing two bars subjected to an axial load at the middle node. The objective is to minimize total mass by varying the bar areas (decision variables A_1 and A_2), with constraints on the maximum stress of each bar; **b** Original decision space, with the white area showing the feasible region of the design space (note the linear feasible subspace near (1, 0) along $A_2 = 0$), and the colored regions noting the constraint violations. Starting from a feasible point $\boldsymbol{\rho}_0 = (1, 1)$, MMA converges to the local optimum $\boldsymbol{\rho}_l = (0, 1)$. Also shown is the projected trajectory after reparameterization with an appropriate network, converging to the "singular" global minimum $\boldsymbol{\rho}_g = (1, 0)$; **c** Network used to reparameterize the problem, with a fixed input ($z_1 = 0.5$) and only 3 parameters (θ_i). The hidden neuron has a parametric sine activation function, and outputs A_1 and A_2, where ω_0 is a hyperparameter; **d** The neural decision space and the corresponding trajectory followed by the optimizer in this space. The decision space (left) consists of repeating units, and the inset (right) shows a zoomed view of the distorted constraint surfaces; **e** Three planes passing through the global optimum showing constraints, infeasible regions, and feasible paths to the solution. More details of the plots are given in SI and Sec. A of the appendix

of the appendix), which have optimized solutions with different characteristic features (e.g., coarse and fine features), to showcase the results. We also consider three representative yet different NN architectures: 1) a feedforward NN with the commonly used Leaky-ReLU activation function (MLP) (Chandrasekhar and Suresh 2020); 2) a feedforward NN with sinusoidal activation functions (SIREN) (Sitzmann et al 2020); and 3) a convolutional NN (CNN) (Hoyer et al 2019). A reader unfamiliar with these NNs is referred to Sec. B of the appendix (as well as the SI).

3 High-dimensional landscape analysis

3.1 Objective landscape visualization

In machine learning (ML) terminology, $\mathcal{F}(\boldsymbol{\rho})$ is the loss function for the NNs. Thus, for every $\boldsymbol{\theta} \in \hat{\mathcal{D}}$, we can associate a scalar value $c = \mathcal{F}(h(\boldsymbol{\theta}))$ and together, this forms the $(M + 1)$ dimensional loss landscape of the reparameterized optimization problem. High-dimensional neural loss landscapes are often visualized by perturbing a single chosen point along a single direction or along two directions (Goodfellow and Vinyals 2015; Li et al 2018). However, none of these methods allows a fair comparison of neural schemes

with the baseline. Unlike neural network parameters, the decision variables in the baseline are bounded, and thus perturbations can violate the bounds, making the loss evaluation impossible, especially near minima. More importantly, the magnitude of the perturbations is arbitrary. As a result, smaller perturbations often show convex landscapes (due to the local convexity of even non-convex functions). Thus, comparing loss landscapes without considering the length-scale of the visualization may be meaningless.

We propose a simple yet general method, building on the 1D visualization, to compare different optimization landscapes. To do so, we adopt the definition of reparameterization as a mapping from the decision space to the physical density space (i.e., $\boldsymbol{\rho} = h(\boldsymbol{\theta})$). Under this definition, neural networks are analogous to the traditional filters and projections commonly used in density-based topology optimization but with a length-scale dependent on the specific architecture (Dupuis and Jacot 2021). The procedure is detailed in Algorithm 1 in Sec. C of the appendix. First, we choose two reference points $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ in the physical density space. For this study, we chose the physical density of the baseline's converged solution as the first reference point. For the second point, we investigated two options:

1. A uniform density design, which corresponds to the commonly used starting point of the baseline; and
2. Multiple random density designs, which are commonly used as initialization points for the neural networks.

Second, for each reparameterization, we find the points in the decision space that generate these densities by solving the following optimization problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{j=1}^{N} \left(h(\boldsymbol{\theta})_j - \rho_j\right)^2, \tag{3}$$

where N is the number of finite elements[4]. By solving Eq. (3) once for each reference point and for a given reparameterization h, we obtain the corresponding decision space points $\hat{\boldsymbol{\theta}}_1$ and $\hat{\boldsymbol{\theta}}_2$. Visualization in 1-D works by interpolating between these two points, i.e., by evaluating the objective and constraint values at a series of points between them. If the two points chosen have decision variables that are within bounds, all points along the line joining them would also satisfy the bounds. Mathematically, any point on the line joining two points $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ can be represented as

$$\boldsymbol{\theta}_\alpha = \boldsymbol{\theta}_1 + \alpha\left(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\right), \quad 0 \le \alpha \le 1. \tag{4}$$

Therefore, the loss landscape can be plotted by calculating the objective $\mathcal{F}\circ h(\boldsymbol{\theta}_\alpha)$ and constraint $g_0 \circ h\left(\boldsymbol{\theta}_\alpha\right)$ values for different values of α. By keeping the density space points the same for different reparameterizations, the loss landscapes' length-scales are linked and comparisons are fair. To remove the bias introduced by the dimensionality of the design space, we chose network architectures (for MLP, SIREN, and CNN) with roughly the same number of parameters as the baseline.

Fig. 3 shows compliance on the ordinate axis as a function of linear interpolation between these two reference points, where $\alpha = 0$ and $\alpha = 1$ denote initial and final designs, respectively[5]. The first reference point refers to an initial design with uniform density (solid thick line) or random density initial designs (dashed thin lines). The second reference point is the solution obtained from the baseline for $p = 1$, known to be a convex problem (Sigmund et al 2016). Parameters are then determined to represent these points, and compliance values are evaluated for interpolated parameters $0 < \alpha < 1$. The figure also indicates if the constraint on maximum material has been violated. Fig. 3 clarifies that neural TO leads to non-convex paths that link different initialization points to the final design obtained by density-based TO, i.e., neural TO introduces "bumps" in the optimization path when linking the same initial and final designs of the baseline. This is relevant because these "bumps" impact the optimization process, as shown in the next subsection. Interestingly, the CNN architecture is less prone to introducing non-convexities than the other two architectures. While the results are shown for $p = 1$, the same holds for $p = 3$, as well as for all other TO examples we considered (see SI). For $p = 3$ non-convexities are even more pronounced.

3.2 Optimizer trajectories

Visualizing objective landscapes offers qualitative insights into the optimization process, but the linear slices observed do not depict the actual optimizer's path. These landscapes were visualized by connecting initial and final designs (decision variables) obtained through the baseline strategy (not neural TO). Hence, the second step in our analysis contrasts the actual trajectories followed by the optimizer in neural TO versus those of the baseline. This approach uses the optimizer as a probe to explore the landscape. To ensure fairness, we initiate all analyses from the same starting point. The idea is that by using the same optimizer, the differences in the trajectories can reveal how the reparameterization alone affects the landscapes. To compare the trajectories,

[4] Note that even though we have used $\boldsymbol{\theta}$ here, this method is equally applicable to visualize the baseline's landscape as well. For instance, it can be used to assess the effect of adding projection filters or making changes to SIMP's penalty, among other modifications. Furthermore, other reference points can also be chosen based on convenience.

[5] This linear slice of the landscape reflects what a line-search algorithm would encounter along this direction.

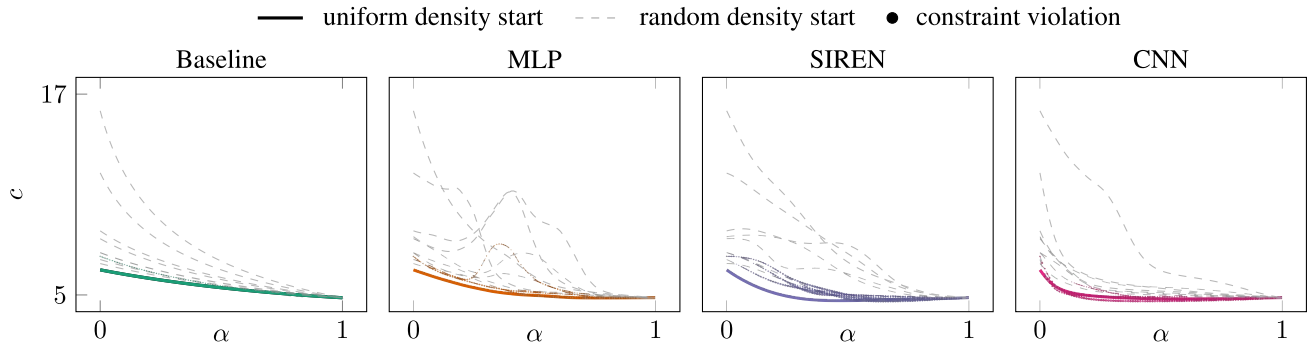


Fig. 3 The objective landscapes (interpolating between the same reference points) for different neural reparameterization methods compared against the baseline. The end point, at $\alpha = 1$, is the decision space point corresponding to the baseline solution ($\hat{\theta}^{\star}$) while the starting point is either uniform gray ($\hat{\theta}_u$, solid thick lines) or random values (denoted by multiple gray dashed thin lines). Plots are shown for Michell boundary value problem for SIMP penalty $p = 1$ (see SI for more results). Constraint violations are indicated by colored markers, with the size of the markers proportional to the violation at each point

we note that the gradient of the objective function ∇c plays an important role in how the optimizer traverses the landscape. Therefore, we plot the magnitude of the objective's gradient $\|\nabla c\|$—calculated using automatic differentiation and verified using finite difference calculations—and the gradient's direction. The latter is calculated as the angle ϕ between successive gradient vectors according to the cosine similarity, i.e.,

$$\cos\phi = \frac{\nabla c^{(i)} . \nabla c^{(i-1)}}{\left\|\nabla c^{(i)}\right\| \left\|\nabla c^{(i-1)}\right\|},$$

for iterations $i-1$ and i. In addition, we plot the optimization history, showing how the objective and constraints vary from one step of the optimizer to the next in the different landscapes. Finally, we examine the linear subspace that connects the starting and ending points of the optimization trajectory. To do this, we use Eq. (4) to interpolate between the starting point (uniform gray, which remains the same for all methods) and the solutions found in each decision space. It is important to note that such a plot does not allow comparisons among landscapes but provides information specific to the converged solution, as Eq. (3) is not used for making these plots.

We consider two different optimizers, such that the distortion of the objective landscape by neural TO is isolated from the choice of optimizer: 1) MMA (Svanberg 1987), which is common in TO literature; and 2) Adam (Kingma and Ba 2015), which is used in the NN literature. Note that hyperparameter optimization is performed for each optimizer fairly for every test case considered (see Sec. E of the appendix). Fig. 4a shows the compliance normalized by the compliance of the best design obtained by the baseline for the Michell problem with $p = 3$, as well as the volume constraint when using MMA with each of the architectures for neural TO. In essence, all cases converge to similar and feasible designs (see Fig. 4c), although usually requiring more iterations for neural TO compared to the baseline method. Fig. 4 also provides additional information on two important characteristics of the optimization process for neural TO. First, if we linearly interpolate between the initial point in the decision space to the final one, we see in Fig. 4b that the objective landscape is either non-convex or has significant constraint violations for all architectures of neural TO (and for all problems we evaluated, as can be seen in SI). Second, the evaluation of the gradient norm and angle for each iteration in Fig. 4d reveals that all neural TO strategies zigzag through the optimization path, i.e., the angle is rarely zero as it was observed for the baseline. This demonstrates that neural TO introduces non-convexities ("bumps") in the objective landscape (as discussed earlier in Fig. 3), even for otherwise convex landscapes as obtained for $p = 1$. These "bumps" perturb the way the optimizer traverses the landscape, thereby slowing down the optimization, i.e., neural TO usually requires more iterations to converge as compared to the baseline method. SI shows the same results when considering the Adam optimizer, which favors the optimization process of neural TO. While using a CNN architecture proved surprisingly competitive, the results herein seem to indicate an overall negative outcome for neural TO since they require more iterations, at least for well-behaved and nearly convex compliance optimization problems. This is an important limitation in practice.

4 Expressivity of neural TO

The analysis so far focused on NN architectures whose number of parameters is comparable to those used by the baseline. However, the design space of neural TO can be over- or under-parameterized, i.e., the number of decision variables (weights

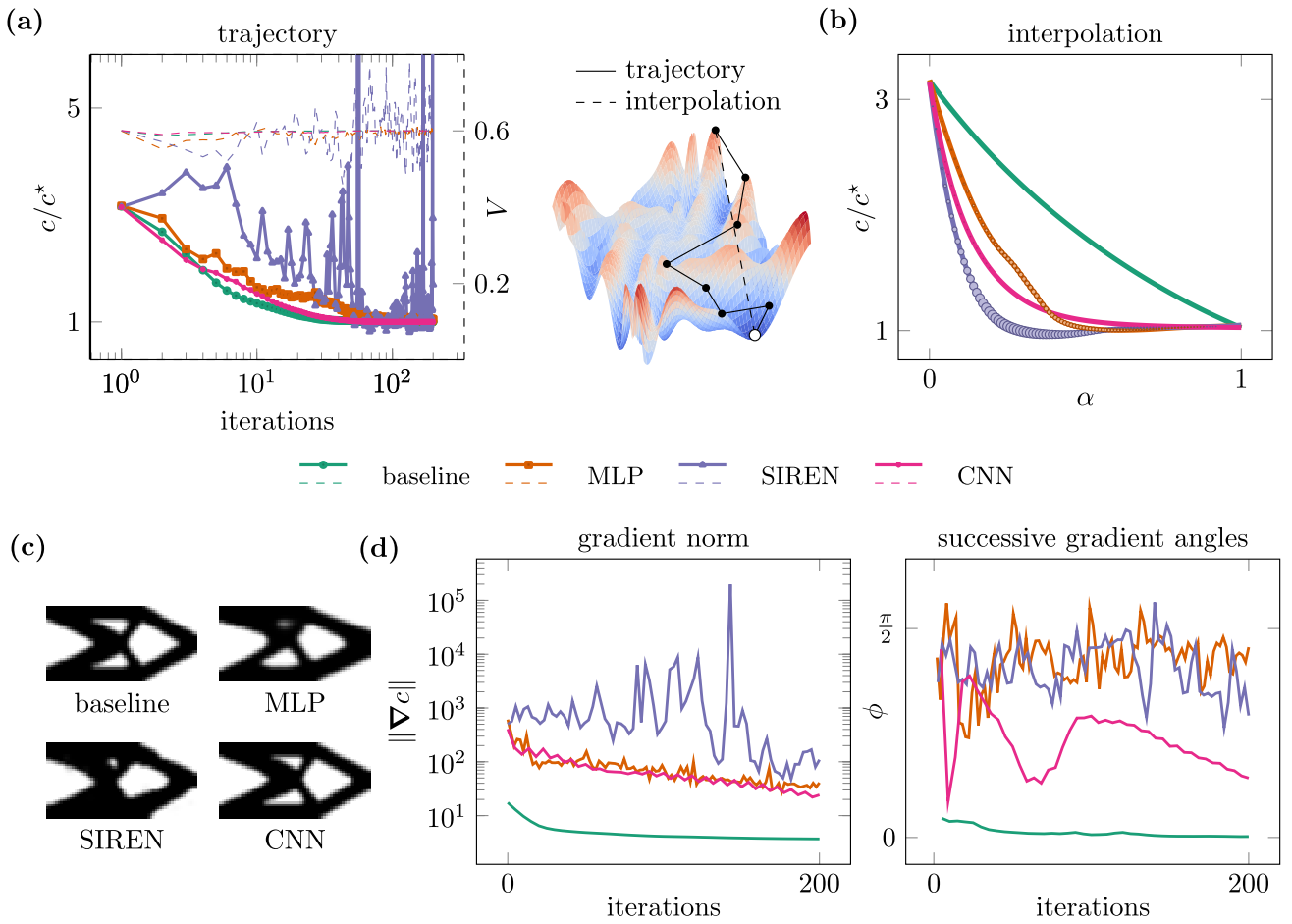


Fig. 4 Comparison of MMA's trajectory on the neural landscape against the conventional landscape for the Michell problem (with penalization $p = 3$ and target volume fraction of 60%): **a** Compliance c normalized by the baseline solution $c^\star$ (left ordinate) and volume fraction V (right dashed ordinate), as functions of the optimization iteration; **b** Normalized compliance interpolated between the initial and optimized solutions. The size of the dots indicate the amount of constraint violation; **c** Best feasible designs obtained during optimization for each method, all having similar compliance; **d** L_2-norm of the objective gradient and the angle between successive gradient vectors at each point along the optimizer's trajectory

and biases of the NN) can be higher or lower than the number of finite elements' density values. Here we investigate the effect of the decision space dimensionality, i.e., the number of trainable parameters, for the three NN architectures. If a network is not capable of representing the necessary structural features, then neural TO will not be effective because there will be designs that cannot be created (the design space becomes restricted). Conversely, even if a network is capable of representing the necessary structural features, the objective landscape's non-convexity with respect to the network parameters can make the optimization process more difficult (or easier), as demonstrated earlier.

Herein we use a simple strategy to assess the network expressivity by assuming that baseline solutions represent the ground truth (for which we used MMA with tuned hyperparameters). We then quantify the neural networks' ability to represent such topologies and use that quantity as a measure of the expressivity. We use the peak signal-to-noise ratio (PSNR), which is commonly used in computer vision literature to measure the discrepancy between a noisy (or reconstructed) image and the ground truth. The PSNR is calculated as

$$\text{PSNR} = 10 \log_{10} \frac{\text{R}^2}{\text{MSE}}, \tag{5}$$

where R is the maximum possible pixel value that we set to R = 1, and MSE refers to the mean-squared error. Therefore, a high PSNR value corresponds to a design that is visually similar to that obtained by the baseline. For instance, the lowest value in Fig. 11 is PSNR $\approx$ 33 and is visually indistinguishable.

First, we generated baseline solutions for three test cases—namely, the Messerschmitt-Bölkow-Blohm (MBB), Michell, and cantilever beams (see Fig. 13)—at the required mesh resolution, with the target volume set at $V_0 = 30\%$. Notice that the

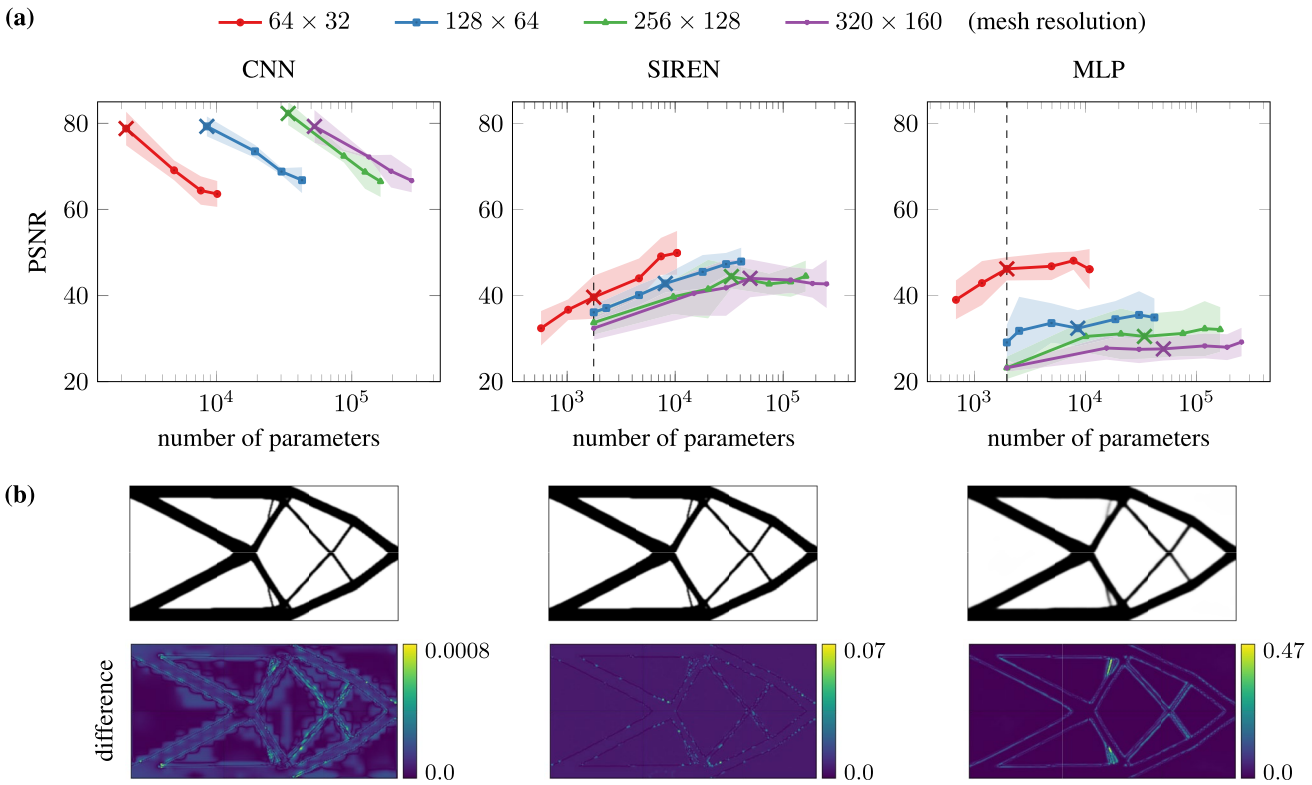


Fig. 5 **a** Peak signal-to-noise ratio (PSNR) values for NNs with different number of parameters, measured for 4 mesh resolutions (image sizes). A higher PSNR value indicates that the NN is able to accurately represent the design obtained from the baseline (taken as ground truth). Shaded region denote confidence intervals (one standard deviation) measured across several tuned hyper-parameters. The dashed vertical lines correspond to a network that has about 2000 parameters, independent of mesh resolution. Cross markers indicate architectures with parameters matching each mesh resolution, distinguishing between over-parameterized and under-parameterized regimes. **b** Final designs for the Michell beam problem and their deviations from the baseline for a 320×160 resolution for all networks corresponding to cross markers

optimized designs for these bending-dominated problems will portray fine structural features. Next, we selected a series of networks—both under- and over-parameterized—with different number of network parameters. Namely, the ratios between the network and baseline parameters are $\left\{ \frac{3}{10}, \frac{6}{10}, \frac{23}{10}, \frac{37}{10}, \frac{50}{10} \right\}$ (see SI for exact architecture details)[6]. Second, we solved Eq. (3) for each network and test case combination by substituting $\boldsymbol{\rho}$ with the target baseline solution $\boldsymbol{\rho}^\star$ and subsequently computed its corresponding PSNR value. After obtaining the PSNR values for the three test cases, the worst value was taken as the measure for that particular network at that mesh resolution. Because this measure is influenced by the optimizer and its hyperparameters, we also conducted hyperparameter tuning to maximize the PSNR value. Finally, we repeated this procedure five times with different starting points for the hyperparameter optimization to obtain PSNR values for each set of the best-identified hyperparameters. This allowed us to calculate the mean and confidence bounds. We repeated the experiment for four different mesh resolutions (64×32, 128×64, 256×128, and 320×160).

Fig. 5 presents the number of network parameters in abscissas and the peak signal-to-noise ratio (PSNR) in ordinates. PSNR is a common metric for assessing image reconstruction quality, with values above 60 dB considered high. The PSNR is calculated based on the worst fit obtained from the Messerschmitt-Bölkow-Blohm (MBB) beam, cantilever beam, and Michell beam test cases (see Fig. 13). The CNN has the highest reconstruction capabilities for any number of parameters[7]. Unsurprisingly for a reader familiar with machine learning, MLP and SIREN are less expressive than CNNs. Each curve also includes a cross symbol that separates the under-parameterized regime (to the left) from the over-parameterized regime (to the right). Noteworthy, CNNs are always over-parameterized and both SIREN and MLP lose expressivity as they become under-parameterized.

[6] We note that the CNN architecture cannot be under-parameterized when compared to the number of finite elements, thus all CNN networks are only over-parameterized.

[7] The decreasing PSNR values for increasing CNN parameters is due to two factors: 1) Training with lower floating-point precision on GPUs introduces numerical effects as the errors are already very low ($< 10^{-6}$); and 2) Allocating parameters across different types of CNN layers can impact performance.

Finally, the expressivity of MLPs drop drastically at higher mesh resolutions, where more fine features are present (Fig. 5b).

The differences in expressivity have been explained in NN literature in the context of image learning (Ulyanov et al 2017; Strümpler et al 2022). CNNs use convolutional filters that capture important features of images and introduce translation equivariance. While MLPs start training by learning low-frequency features, SIRENs do so with high-frequency features (Rahaman et al 2019). Practical TO problems often involve slender structures that are more easily reconstructed by networks able to generate high-frequency features.

Regarding the MLP and SIREN network architectures, in most published works the network size is kept constant without regard to the mesh resolution (Chandrasekhar and Suresh 2020; Deng and To 2020; Chandrasekhar and Suresh 2022). A vertical dashed line in the figure indicates the PSNR results attained by a network architecture with approximately 2000 parameters. At the highest resolution, this fixed architecture must represent a design with $320 \times 160 = 51\,200$ pixels (or densities) using approximately 2000 parameters, resulting in over a 250-fold compression and degrading the representation quality accordingly. Thus, using a network of fixed size produces simpler structures as the mesh resolution is increased. Finally, the superior performance of SIREN compared to MLP can be attributed to SIREN's ability to represent high frequencies necessary for sharp solid-void transitions, similar to square waveforms. MLP struggles with this due to its spectral bias, which causes it to fit lower-frequency signals first and capture higher frequencies very slowly.

5 Performance of neural TO

While the expressivity of a reparameterization can be advantageous for certain boundary value problems, it may prove detrimental for others, depending on the problem's characteristics. For instance, the density filter used in the baseline reduces expressivity but prevents convergence to checkerboard patterns, which are otherwise favored by the optimizer. Ultimately, the optimization dynamics dictate the effectiveness of a given reparameterization.

We employ performance profiles (Dolan and Moré 2002) to compare neural TO against the baseline, considering three boundary conditions (MBB, tensile, and bridge cases) and volume fraction constraints ranging from 10% to 60%, ensuring diverse problem features. This results in a total of 18 distinct problems (see Rojas-Labanda and Stolpe (2015) for an example of performance profiles in TO). The profiles evaluate three metrics: (1) the best objective value achieved within a fixed budget of 200 function evaluations (first column), (2) the number of iterations to convergence (second column)[8], and (3) the actual performance after thresholding gray designs into black-and-white. The abscissa represents the tolerance percentage, indicating the allowable deviation from the best solution obtained across all methods (not necessarily the baseline). The ordinate shows the percentage of problems where the performance satisfies this tolerance. Since the performance of optimizers depends on the chosen hyperparameters, we tuned them individually for each test case and for all methods (including both the baseline and neural reparameterizations) to ensure fair comparisons. Hyperparameter tuning can be computationally expensive as it involves a bi-level optimization process, where topology optimization is repeated for each set of hyperparameters. For efficiency, the hyperparameters for each problem were optimized on a coarser mesh size of 64×32 and then applied to the final optimization at a higher resolution of 576×288 (see Sec. E of the appendix for more details).

The performance profiles reveal several intriguing insights into the interaction between the optimizer and the loss landscape. First, Adam consistently outperforms MMA in finding better solutions, as evidenced in the first column of Fig. 6. This is because Adam's adaptive learning rates enable it to navigate the tortuous objective landscapes of neural TO effectively, making it a favored optimizer in machine learning. This holds true irrespective of the reparameterization chosen. Second, among the neural reparameterizations, MLPs are the least effective in finding optimal solutions. Their limited expressivity prevents them from representing thin structural members, which are critical for bending-dominated problems. Conversely, CNNs, with their high expressivity, can sometimes suffer from structural links being severed during thresholding, leading to increased compliance (third column). Interestingly, CNN-based designs often remain viable even after thresholding, especially when the optimization starts from favorable initializations. Striking a balance, SIRENs maintain their performance after thresholding due to careful tuning of their frequency hyperparameter to match the problem characteristics. Third, on average, all neural TO schemes converge more slowly than the baseline, with SIRENs being the slowest (requiring more than twice the number of iterations). This is attributed to the increased complexity of the associated loss landscapes. However, CNNs can converge on par with or even faster than the baseline, benefiting from the relatively well-behaved landscapes they generate. Finally, when focusing on the best performance, CNNs optimized with MMA emerge as strong competitors to the baseline. They achieve comparable objective values both before and after thresholding, and, in many cases, converge as quickly or faster. This finding challenges the common perception

[8] Convergence is defined as the iteration at which the objective value stabilizes, i.e., when the optimizer achieves an objective value within a percentage (here set to 5%) of its corresponding best.

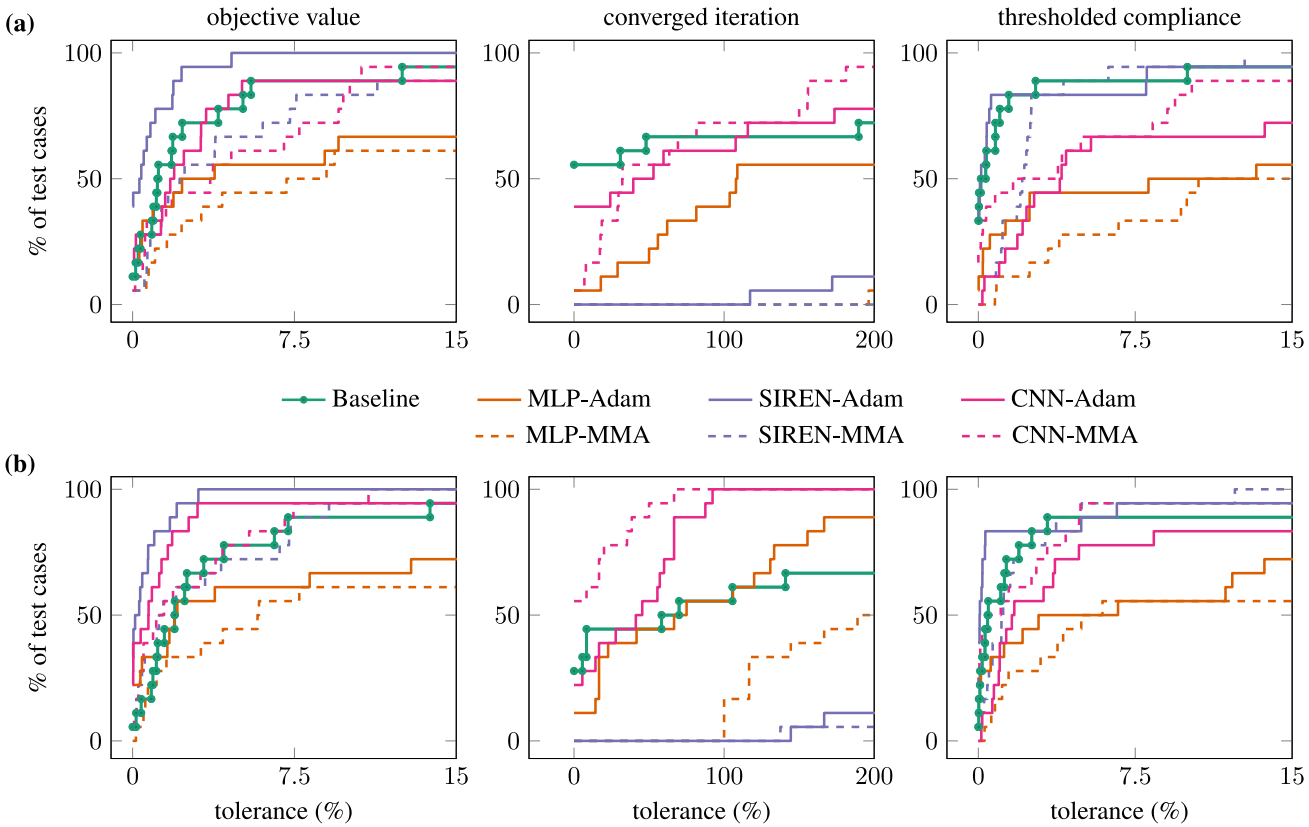


Fig. 6 Performance profiles showing **a** Median performance across six different network initializations; and **b** Best performance. For all methods, the profiles compare the best objective value attained during optimization (left column), the number of iterations to convergence (middle column), and the compliance after thresholding to black-and-white designs (right column). The plot shows the percentage of test cases (ordinates) where each method achieved results within a tolerance (abscissas) of the best-performing method for that case. Neural TO was carried out using both Adam and MMA optimizers, while the baseline method only used MMA. All networks have parameters corresponding to the a mesh resolution of 576×288 elements, and were pretrained to start with a uniform density distribution

that MMA is unsuitable for optimizing neural networks. We hypothesize that the nearly convex loss landscape associated with CNNs facilitates this synergy.

Furthermore, for the MBB, bridge, and tensile test cases considered during benchmarking, we analyzed the robustness of each method to hyperparameters, as shown in Fig. 7 using box plots. These box plots illustrate the median, 25% and 75% quartiles, and outliers, based on hyperparameter optimization results after excluding divergent cases. The baseline emerged as the most robust approach, while the performance of the SIREN was highly sensitive to hyperparameter selection. This variability in performance is also evident when considering the initialization of neural topology optimization. Despite starting each optimization run from a nearly uniform output, different runs produced designs with varying performance. These findings highlight the high sensitivity of neural TO—particularly when using randomly chosen networks—to both initialization and hyperparameter selection.

6 Outlook: going beyond compliance optimization

Future work will seek to extend neural TO beyond compliance minimization, exploring problems characterized by highly non-convex objective landscapes and more complex constraints. Two recent investigations submitted after the preprint of our article have already initiated this path (Norder et al 2025; Herrmann et al 2024). Norder et al (2025) demonstrated that neural topology optimization was advantageous in the context of Photonics by designing pentagonal crystal mirrors for lightsails, while Herrmann et al (2024) investigated the potential benefits of neural optimization for an acoustic problem. Our work focuses on structurally simpler problems but aims to provide deeper insight into when and why neural reparameterization offers advantages and disadvantages.

Although this is beyond the scope of this work, we include in this section two additional problems—summarized in

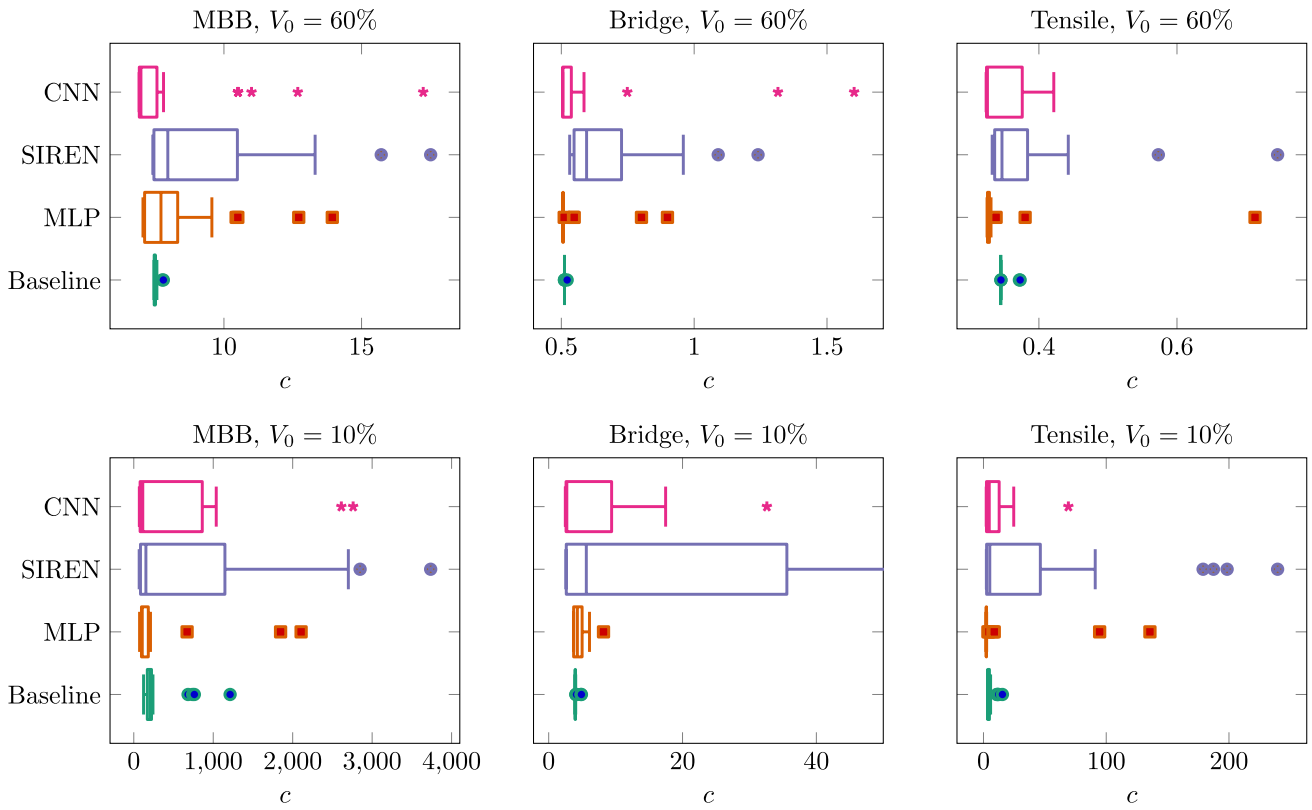


Fig. 7 Robustness to hyperparameter selection when Adam is used for optimizing the networks. The box plots show the variation in the compliance c as hyperparameters are varied for the different models. The outliers are marked while the median (middle of the box) and the 25% and 75% quartiles are shown (box's ends)

Figure 8—that may provide additional insights on the performance of neural TO: the area-to-point thermal conduction optimization, and compliant mechanism design, following the formulations in Wang et al (2010). While both problems share a similar framework with compliance minimization (the heat conduction problem is essentially analogous to it), they differ in their specific objective functions and physical parameters (see Sec. E.5). Our results in Fig. 8 show that neural TO can produce solutions that outperform the baseline (with sufficient hyperparameter tuning), while also emphasizing the challenges posed by complex objective landscapes. Notably, MLPs exhibit a persistent preference for simpler designs, underscoring their limitations in tackling these problems.

6.1 Imposing constraints

We are grateful to an anonymous reviewer for suggesting the inclusion of a brief discussion about imposing non-trivial constraints in neural TO. While a comprehensive investigation is beyond the scope of this work, we implemented an Augmented Lagrangian (AL) formulation to address the stress-constrained optimization problem, following the formulation proposed by (Verbart et al 2016) (detailed in appendix (E.5)). The aim is to minimize structural volume (V) subject to a constraint on maximum allowable stress. We adopt the standard L-shaped bracket as the benchmark problem (Fig. 13) and use the lower-bound Kreisselmeier–Steinhauser (KS) function for constraint aggregation.

To enforce the stress constraint, we employ the adaptive AL method introduced by Basir and Senocak (2022), originally developed for training physics-informed neural networks. This approach is applied consistently across all parameterizations, with hyperparameters tuned individually for each case. The resulting optimization histories, physical designs, and stress fields (from finite element analysis) are shown in Fig. 9. While the method produces realistic structures and yields comparable objective values for most parameterizations (with the exception of the MLP), exact constraint satisfaction remains challenging, requiring many optimizer iterations and extensive hyperparameter tuning. A more thorough investigation is needed to understand these limitations and to develop robust methods for incorporating complex constraints into neural topology optimization frameworks.

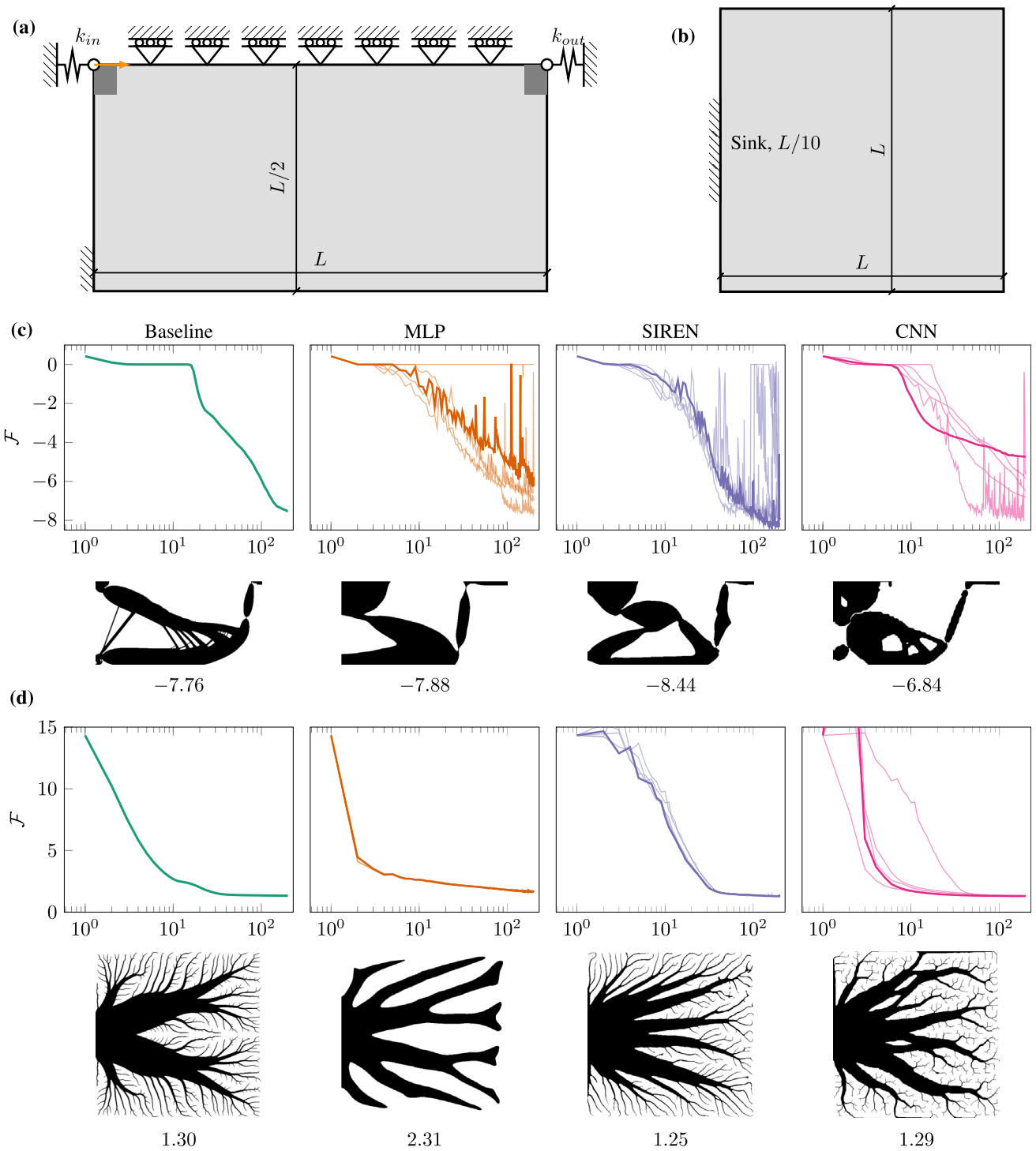


Fig. 8 Comparing neural TO's performance (optimized with Adam) against baseline on compliant mechanism design (**a**) and thermal conduction (**b**) problems. For the NNs, multiple curves correspond to optimizations from six different random starting points. Note that all networks are pretrained to generate uniform gray density field. The plots show the objective value's evolution with optimization iteration. The best thresholded designs and their objective values are also shown below the corresponding scheme. Note that hyperparameter tuning was performed for each method

7 Discussion

Neural topology optimization (neural TO) reparameterizes the decision space, replacing the elemental physical densities on the finite element mesh with the weights (and biases) of a neural network. While such change-of-variables strategies are well established in optimization, the use of neural networks introduces both opportunities and challenges, which we have sought to highlight in this work to provide insights for future research.

From a classical optimization standpoint, neural TO may seem counterintuitive. Traditional TO methods often aim to convexify the design problem to improve convergence and robustness. In contrast, introducing a neural network as a parameterization typically makes the optimization landscape more non-convex, especially in problems that are otherwise nearly convex—such as compliance minimization. We observed this across all three neural network architectures studied. The degree of distortion, however, depended strongly on the network architecture. As a result, neural TO tends to be more sensitive to initialization and hyperparameters, often requiring significantly more iterations to reach designs with comparable objective values to standard density-based methods.

The advantages of neural TO become more pronounced in highly non-convex problems, where local minima vary substantially in quality. This view is supported by the findings of Herrmann et al (2024), who had been aware of the preprint version of this article. They applied neural TO (referred to as optimization with a neural network ansatz) to a challenging acoustic design problem and found that neural

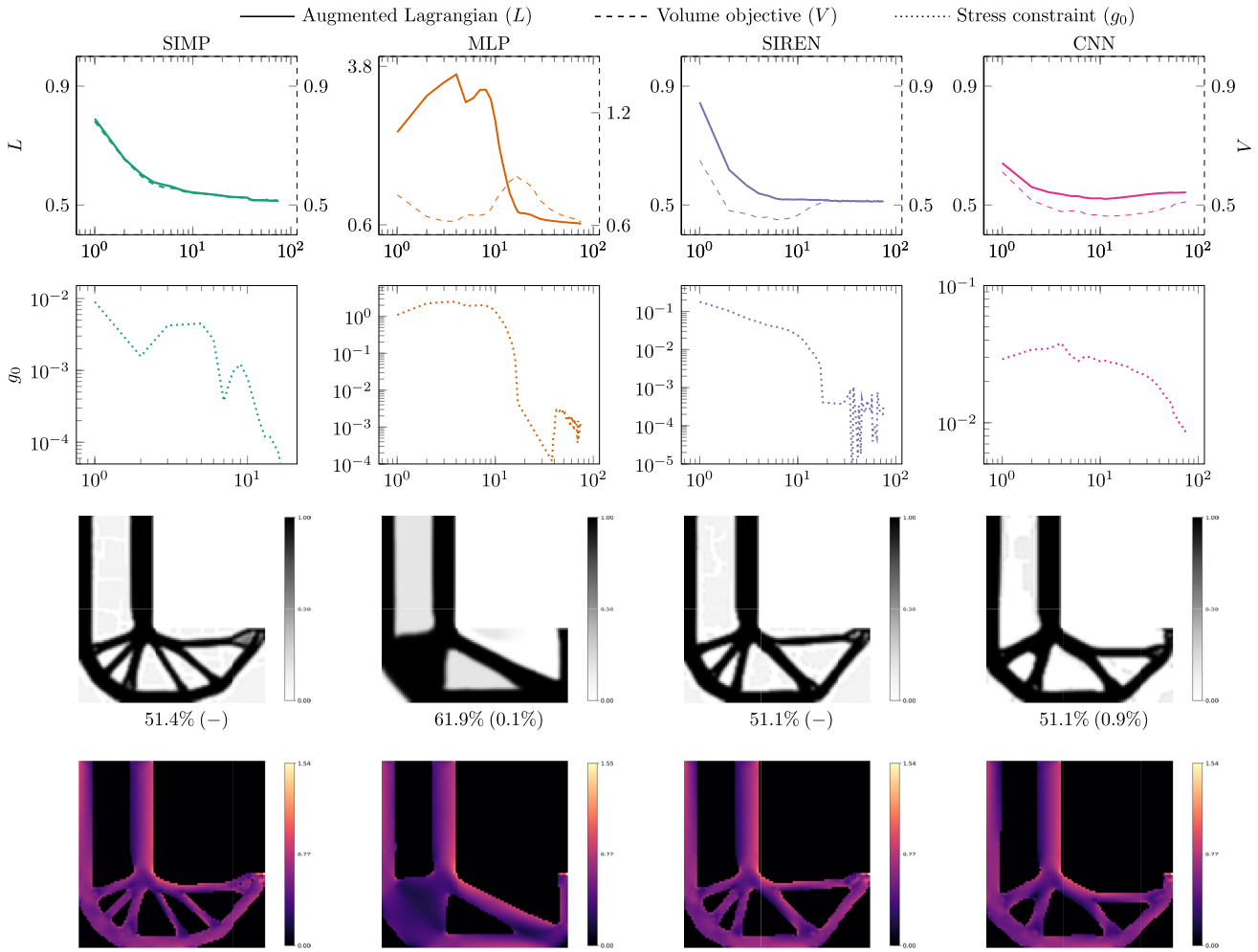


Fig. 9 Comparison of neural topology optimization and baseline methods on a stress-constrained L-shaped bracket problem. First row: evolution of the augmented Lagrangian (L, bold left axis) and objective (V, dashed, right axis) as a function of iterations; L may increase as it balances contributions from the objective and constraints. Third and fourth rows: final designs, corresponding objective values (with constraint violations), and stress distributions in regions with material density above 0.5. All methods use a modified augmented Lagrangian (Basir and Senocak 2022) with the lower-bounded KS function (Φ^{L}) (Verbart et al 2016), with hyperparameters tuned for fair comparison

parameterization can lead to better-performing designs. They conjectured that overparameterization may help escape poor local minima by reducing the likelihood that all gradient components vanish simultaneously. Here we show definite proof using the 2D truss problem that reparameterization, even when the over-parameterization is not severe, creates access paths to otherwise inaccesible minima by reshaping the landscape. The resulting landscape is determined by the inductive biases—i.e., the network's natural preference toward representing certain function classes over others. This was evident in our expressivity study: MLPs, with their limited expressivity, consistently produced smoother (low-frequency) outputs, regardless of parameter count. As a result, they tended to yield simpler structural layouts across all tested problems. In contrast, SIRENs were more flexible, capable of representing a broader range of frequencies and generating both fine and coarse features. However, the "right" bias is problem-specific. Arbitrarily chosen architectures may underperform—even with tuning—because the extent to which one can influence a network's bias by adjusting width or depth alone is limited. For example, Herrmann et al (2024) reported that using an MLP for certain analytical functions failed to produce the expected improvements. Our study provides a clear explanation for this outcome through the lens of inductive bias and expressivity.

This motivates the need for domain-informed architectures. Many of the networks used in this and related studies were not designed with TO in mind. CNNs, for instance, were originally developed for image-processing tasks, and features like residual connections were added to ease optimization in that context (Li et al 2018). Yet, CNNs in our study surprisingly performed well even when optimized using classical methods like MMA. This could be due to the structural similarity between images and the grid-based representations of TO designs. These observations suggest that neural TO could benefit significantly from architectures specifically designed for structural optimization—ones that embed properties like connectivity, load-bearing behavior, or manufacturability directly into the network. Early signs of this promise are visible in work on domain-informed reparameterizations (Both et al 2023; Heydaribeni et al 2024; Berzins et al 2024). However, their application to TO remains underexplored. Future research should aim to develop and evaluate such architectures, potentially leveraging experience from solved TO problems to guide network design or initialization. To provide insights into designing architectures, we introduced simple yet effective analysis tools, including visualization of objective landscapes, characterization of optimizer trajectories, and measurements of neural network expressivity. These tools can guide the design of both neural and density-based TO approaches, helping researchers identify which architectural choices improve convergence and solution quality.

Finally, it is important to acknowledge the limitations of this study. First, empirical performance alone cannot determine whether neural TO is fundamentally superior or inferior to conventional methods. The effectiveness of any reparameterization—neural or otherwise—depends on the specific characteristics of the optimization problem. In line with the "no free lunch" theorem (Wolpert and Macready 1997), no single approach can consistently outperform all others across every problem class. Our objective was not to establish neural TO as universally better, but rather to understand and articulate when and why certain neural architectures may offer advantages—or face limitations—within TO frameworks. Second, the problems investigated in this study are relatively small in scale. Future research should assess the scalability of neural TO, including its computational and memory costs, especially when applied to high-resolution or three-dimensional problems. Third, this work focused mostly on problems with a simple volume constraint. Reliably enforcing general constraints in neural network-based optimization is an important challenge, and the Augmented Lagrangian implementation considered in Section 6.1 needs to be investigated thoroughly in the future. Other formulations for imposing constraints could also be considered. For example, we experimented with a quadratic penalty method but found it to be highly dependent on the starting penalty value and its increment. We conjecture that neural TO is better suited for penalty or augmented Lagrangian methods because it leverages standard unconstrained optimizers, but we also showed that MMA can be used in neural TO. Still, we expect that the unconstrained optimizers that are commonly used in neural network training are better suited for neural TO. Overall, the development of effective and scalable constraint-handling techniques will be critical to advancing neural TO for more complex and realistic applications.

Appendix A Two-variable stress-constrained problem

We adopt the following problem formulation (Verbart et al 2016):

$$\begin{aligned} \boldsymbol{\rho}^\star = \underset{\boldsymbol{\rho}=(A_1,A_2)\in\mathbb{R}^2}{\arg\min} \quad & \mathcal{F}\big(A_1,A_2\big) = 0.6A_1 + 0.8A_2, \\ \text{such that} \quad & \bar{g}_i = \left(\tfrac{A_i}{2}\right) g_i \le 0, \\ & 0 \le A_i \le 2, \quad i \in \{1,2\}, \end{aligned} \tag{A1}$$

where the unscaled stress constraint is $g_i = \frac{|\sigma_i|}{\sigma_{\max}} - 1$, and $\sigma_{\max} = 1$ is the allowable stress. The objective is to find the values of A_1 and A_2 that minimize $\mathcal{F}$ according to these constraints. The global optimum is known to be at $A_1 = 1$ and $A_2 = 0$, which is a point difficult to reach with

gradient-based optimizers in this decision space (recall Fig. 2b). To solve this constrained optimization problem, we used the method of moving asymptotes (MMA Svanberg (2002)) as the optimizer, with a move limit parameter $m = 2.0$ and the asymptote initialization $a = 0.1$. We tried different values for these parameters by performing hyperparameter optimization (including other starting points) but all of them converged to the local minimum at (0, 1).

We then reformulated this problem as a neural TO example considering the smallest NN that reparameterizes the original two-dimensional decision space (A_1 and A_2) into a three-dimensional decision space ($\theta_1 = W_1$, $\theta_2 = W_2$ and $\theta_3 = W_3$) corresponding to the three weights of the NN (Fig. 2c). The NN architecture has only 4 neurons: a fixed input neuron $z_1 = 0.5$, a single hidden layer neuron with a parametric sine activation function of the form $\sin\left(\omega_0\theta_1 z_1\right)$, and two output neurons that yield A_1 and A_2. We can write this explicitly as: $A_i = \sin\left(\theta_{i+1}\sin\left(\omega_0\theta_1 z_1\right)\right) + 1$, for $i = \{1, 2\}$, where ω_0 is the frequency hyperparameter. We also use MMA as the optimizer, and its hyperparameters (including the bounds for the variables), together with the frequency parameter ω_0 were tuned using the *Tree-structured Parzen Estimator* (TPE) algorithm implemented in Optuna (Akiba et al 2019) so as to attain the global optimum for the original problem. For the results shown in the main text we used $m = 0.31$, $a = 0.1$, $\omega_0 = 88$, and the bounds on θ were set to $[-3, 3]$[9]. It is worth noting that we could find many hyperparameters that converged to the global optimum. Interestingly, for one such set of hyperparameters ($m = 0.4$, $a = 0.3$, $\omega_0 = 40$, and bounds $[-11, 11]$), the neural reparameterization converged in three iterations (see SI).

Landscape plots (Fig. 2 in main text)

For clarity here we explain how the figure in the main text was constructed. Sec. 1 of SI contains the analytical equations of the transformed constraint equations after reparameterization. It is interesting to note that each constraint now has two branches and there exists an infinite number of constraints for different choices of $c \in \mathbb{Z}$ (the set of all integers). These branches are shown in the figure with dashed and continuous lines, where lines of the same color represent the same constraint boundary. Additionally, the axes, which were also part of the constraint boundaries ($A_i = 0$), after transformation, have been represented with dotted lines. We can get equations with similar characteristics for the other planes, i.e., branching into two solutions and being infinitely periodic. However, their expressions are more cluttered, so they are not shown here for brevity but are plotted in Fig. 10a. It should be noted that the global optimum must lie on the constraint boundary and is marked on all three planes with $\boldsymbol{\theta}_g$. In Fig. 2e of the main document, these boundaries are plotted over a much smaller range and were identified empirically from data[10].

Fig. 10b reconstructs parts of the main document figure to explain how the optimizer can access the global constrained optimum in the neural space. On the left, the original decision space is shown schematically, highlighting the 1-D feasible subspace to the global optimum along $A_2 = 0$. The middle figure presents a scatter plot of a region around the optimum, where marker sizes are proportional to the constraint violation. Values below 10^{-7} are treated as feasible (markers disappear), while violations above 10^{-5} are considered completely infeasible. Between these values, marker sizes scale logarithmically to the maximum size, resulting in a nearly sharp transition at the meeting point of the two branches of $\bar{g}_2 = 0$. Here, the thin linear subspace is barely visible.

For the same plot settings (including the density of scattered points), the $\theta_3 - \theta_2$ plane in the neural space (right) shows a seemingly broader feasible region. However, the analytical plot of the constraint boundaries of the same plane (focusing on the inset at the top-right corner) shows that the only access is along the horizontal dotted line (corresponding to $A_2 = 0$)[11]. Thus, while the feasible access is mathematically still one-dimensional, the neural reparameterization induces a "constraint relaxation" (similarly to what is discussed by Verbart et al (2016)), creating a surrounding region with negligible constraint violation, thereby facilitating access to the optimum. We made the same plots by lowering the threshold for feasibility (from 10^{-7} to 10^{-10}) and observed no changes.

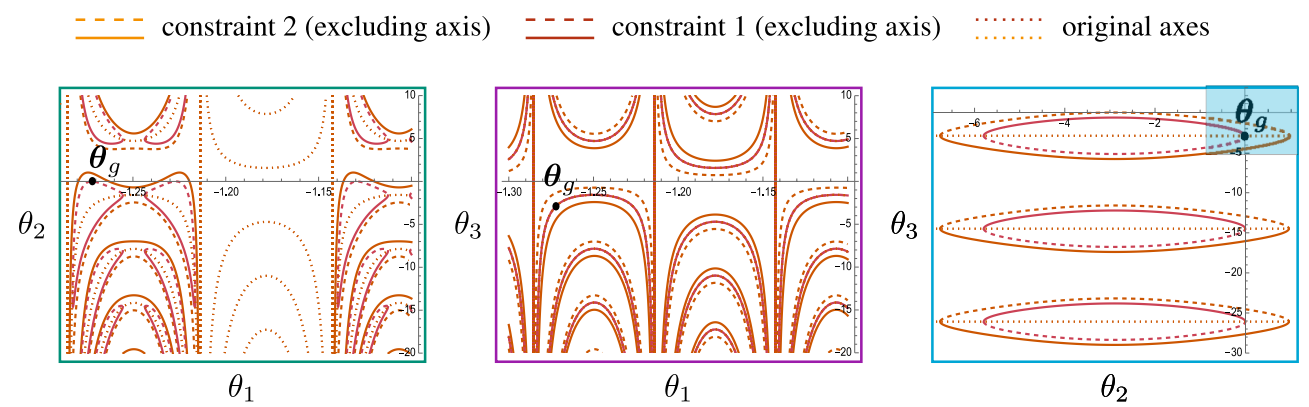
Appendix B Neural network architectures

The first neural network architecture chosen for this work is the multi-layer perceptron (MLP), where the outputs of neurons in a given layer are connected to all neurons of the next layer (see Fig. 2 of SI). Although similar fully-connected networks have been used by others (Deng and To 2020; Zehnder et al 2021; Mai et al 2022; Jeong et al 2023; Qian et al 2022), we adopt the specific structure chosen by Chandrasekhar and Suresh (2020), which has five hidden layers. Each hidden layer performs batch normalization (Ioffe and Szegedy (2015)(see Sec.2 of SI) and applies

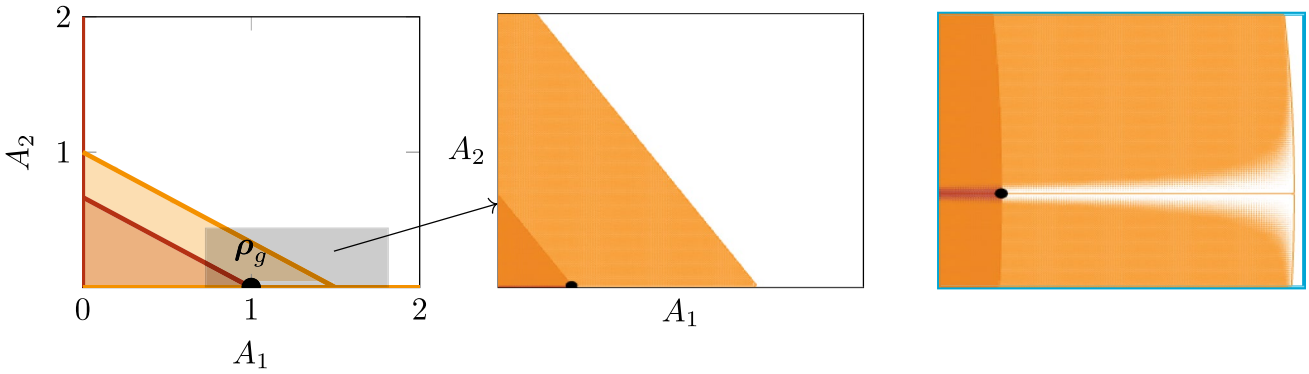
[9] MMA requires bounds on the decision variables for optimization. Since neural weights are unbounded, we have to restrict their range. The choice of bounds is an important hyperparameter.

[10] A uniform grid is constructed, and the constraint functions are evaluated at each node. The boundaries are then identified from this data using a marching cubes algorithm.

[11] The reader is reminded that the region inside the ellipses is infeasible, except for this line.



(a) Analyical solutions for the constraint boundaries



(b) Zoomed views of the original decision space (middle) and $\theta_3 - \theta_2$ plane (right)

Fig. 10 **a** Analytical constraint boundaries in the neural space at three orthogonal planes intersecting the global optimum ($\boldsymbol{\theta}^\star = (-1.272, 0, -2.901)$) found by the MMA optimizer. Dashed and continuous lines of the same color designate the two branches. Dotted lines represent the transformed equations $A_i = 0$; **b** Illustration of how the original 1-D degenerate subspace to the global minimum is transformed in the neural space. The original decision space is shown on the left (the degenerate subspace has been enlarged for clarity), while the middle figure provides a zoomed-in view of the degenerate subspace (actual scatter plot). The figure on the right shows the same subspace in the neural space. Both plots have the same density of points, and the marker sizes are proportional to the constraint violation

Leaky-ReLU nonlinear activation $\sigma = \max(0.01\boldsymbol{l}, \boldsymbol{l})$, where the max function is applied element-wise.

Although MLPs are universal approximators, they suffer from spectral bias and struggle to represent high-frequency features. To address this, either input coordinates can be projected using random Fourier features (RFFs) or sinusoidal activation functions can be used. Chandrasekhar and Suresh (2022) used RFFs for length-scale control in topology optimization. Using the same fully connected architecture but with a sine activation function results in SIREN networks (Sitzmann et al 2020). In our study, we chose SIRENs for their structural similarity to MLPs and their memory efficiency. This choice does not impact generality, as SIRENs have been shown to be equivalent to RFF-MLPs (Benbarka et al 2022). A hidden layer of a SIREN takes the input and applies the transformation $\sin\left(\omega_0(\boldsymbol{W}\boldsymbol{l} + \boldsymbol{b})\right)$, where ω_0 is a hyperparameter that dictates the highest frequencies that can be represented. The coordinates of the centers of the finite elements $\boldsymbol{x}_i$ are given as inputs to the network and their corresponding density values $\rho(\boldsymbol{x}_i)$ are retrieved from the network. For a 2-D finite element mesh, this mapping can be written as $\rho_i = h(\boldsymbol{\theta}; \boldsymbol{x}_i) : \mathbb{R}^2 \to \mathbb{R}$.

Finally, we also consider a convolutional NN (CNN), an architecture widely used for image processing. Specifically, for this study we use the decoder-type architecture adopted by Hoyer et al (2019) and by Zhang et al (2021), which takes as input a vector $\boldsymbol{z} \in \mathbb{R}^n$ to generate the physical density field $\boldsymbol{\rho}$. The input vector—whose dimension n is an architectural choice, often chosen such that it is less than the number of image pixels—is treated as a trainable parameter and is randomly initialized; the CNN's mapping $\boldsymbol{\rho} = h(\boldsymbol{\theta}; \boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R}^N$, yields the full density field of the finite element discretization. Here, $N = N_x \times N_y$, with N_x and N_y denoting the number of finite elements along their

respective Cartesian directions. Unlike MLPs and SIRENs, the CNN architecture (see Fig. 2 of SI) depends on the mesh resolution, i.e., once the architecture is chosen, the resulting design is fixed in size. Therefore, to generate a larger resolution design, the number of CNN parameters has to be scaled up (similarly to the baseline). As opposed to the architecture used by Hoyer et al (2019); Zhang et al (2021) that had 5 hidden layers, we limit the number of hidden layers to two. Additionally, in experiments designed to match the number of parameters of the CNN architecture to that of the baseline, we set the number of convolutional channels as well as the dimension of z to 1, and used only three convolution filters: two in the first hidden layer and one in the last.

Other choices were made to make fair comparisons: We do not use continuation schemes for the SIMP penalty factor. Furthermore, although NNs become more expressive either through increasing the number of hidden layers (depth) (Lu et al 2017) or the number of neurons per layer (width) (Hornik et al 1989), we chose the latter and fixed the depth for all experiments. All networks were implemented in JAX (Bradbury et al 2018) using the Haiku library (Hennigan et al 2020), within our in-house topology optimization library.

B.1 Volume constraint enforcement

Most publications that use reparameterization (e.g., Chandrasekhar and Suresh (2020, 2022) among others) either use quadratic penalty or augmented Lagrangian methods to convert the constrained optimization problem to an unconstrained one, thereby enabling common optimizers in machine learning to be used. However, these introduce additional hyperparameters, such as the initial penalty value, its increment magnitude per iteration, and the criterion used to increase the penalty value. Setting these hyperparameters appropriately can be challenging. Instead, we either use MMA as the optimizer for the neural TO—for which enforcing the volume constraint is straightforward—or when using Adam as the optimizer we enforce the volume constraint through the shifted-sigmoid strategy (Hoyer et al 2019). The latter applies a parametric-sigmoidal transformation on the outputs of the network at each iteration of the optimization process. Briefly, we use the following sigmoid function:

$$\rho_i = \frac{1}{1 + \exp^{(\tilde{\rho}_i - b(\tilde{\boldsymbol{\rho}}, V_0))}},$$

where the output of the network (i.e., $\tilde{\boldsymbol{\rho}}$) is transformed into physical densities bounded between 0 and 1. The scalar parameter b shifts the output to ensure the volume constraint is enforced exactly at every iteration, i.e., $g_0(\boldsymbol{\rho}) = V_0$. This converts the original constrained problem into an unconstrained one, allowing the use of common ML optimizers. The value of b depends on the output and the required volume fraction, and it is determined using a bisection algorithm (within an error of 10^{-12}). For the neural networks, we obtained smoother results when this operation was carried out after the density filtering. We note that this technique is similar to the volume-preserving thresholding filter of Xu et al (2009), except that the smoothness of the projection is not controllable. However, we found our approach to be more robust, as the bisection algorithm successfully found roots for all tested cases without failure.

Appendix C Landscape analysis

Algorithm 1 Loss landscape visualization using linear interpolation

Input: Reparameterization maps: $\mathcal{H} = \{h_1, \ldots, h_n\}$, reference points in the physical density space $\boldsymbol{\rho}_1, \boldsymbol{\rho}_2$
Output: Sets $\boldsymbol{\mathcal{F}}$ (function values) and $\boldsymbol{\mathcal{G}}$ (constraint values)

– *Obtain decision space points*
for $h_i \in \mathcal{H}$ **do**
  $\hat{\boldsymbol{\theta}}_1 \leftarrow \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \|h_i(\boldsymbol{\theta}) - \boldsymbol{\rho}_1\|^2$
  – *Obtain first point in decision space*
  $\hat{\boldsymbol{\theta}}_2 \leftarrow \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \|h_i(\boldsymbol{\theta}) - \boldsymbol{\rho}_2\|^2$
  – *Obtain second point in decision space*
  – *Interpolate between points*
  for $0 < \alpha < 1$ **do**
    $\boldsymbol{\theta}_\alpha \leftarrow \hat{\boldsymbol{\theta}}_1 + \alpha\left(\hat{\boldsymbol{\theta}}_2 - \hat{\boldsymbol{\theta}}_1\right)$
    – *Obtain intermediate parameter point*
    $\boldsymbol{\mathcal{F}} \leftarrow \boldsymbol{\mathcal{F}} \cup \left(\alpha, \mathcal{F} \circ h_i\left(\hat{\boldsymbol{\theta}}_\alpha\right)\right)$
    – *Evaluate function value and store pair in set*
    $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{G}} \cup \left(\alpha, g_0 \circ h_i\left(\hat{\boldsymbol{\theta}}_\alpha\right)\right)$
    – *Evaluate constraint value and store pair in set*
return $\{\boldsymbol{\mathcal{F}}, \boldsymbol{\mathcal{G}}\}$

Table 1 Maximum mean square error during optimization (Eq. (3)) for loss landscape visualization

	Baseline	MLP	SIREN	CNN
Tensile ($p = 1$)	6.6×10^{-8}	5.6×10^{-5}	1.4×10^{-5}	3.9×10^{-7}
Michell ($p = 1$)	9.6×10^{-8}	7.3×10^{-5}	3.3×10^{-5}	2.5×10^{-5}
Tensile ($p = 3$)	7.2×10^{-8}	5.6×10^{-5}	3.4×10^{-5}	3.9×10^{-7}
Michell ($p = 3$)	1.6×10^{-7}	5.6×10^{-5}	1×10^{-5}	4.2×10^{-4}

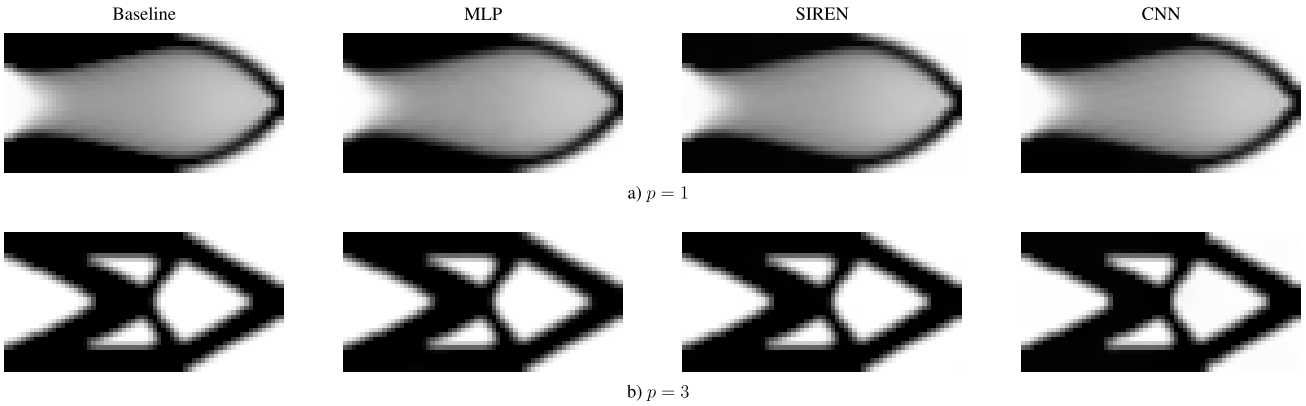


Fig. 11 The physical densities obtained for the Michell test case after solving Eq. (3) for the reference point (corresponding to the solution of the baseline)

C.1 Visualization

The algorithm for generating the 1-D loss landscape plots is detailed in Alg. 1. Additionally, the errors obtained after solving the optimization problem in Eq. (3) using the L-BFGS optimizer are presented in Table 1, while the designs produced by the network after optimization are illustrated in Fig. 11. Additional plots for more cases are provided in the SI for brevity.

C.2 Trajectory analysis

Fig. 12 presents the metrics analyzed along the optimizer's trajectory for all models, test problems (tensile and Michell), and optimizers (MMA and Adam).

Appendix D Expressivity study

The results of this expressivity study are summarized in Fig. 5 of the main text. We also considered the CNN used by Hoyer et al (2019), that even though is over-parameterized by approximately 120 times, it achieved similar PSNR values (71, 74, 75, and 74 for all four resolutions) [12]. SI contains information on the exact architectures used in this study.

Appendix E Topology optimization details

For all the boundary value problems considered in this study (see Fig. 13), Young's moduli were set at 10^{-9} for the void and 10 for the material, and Poisson's ratio was set to 0.3; plane stress conditions were assumed. For solving the finite element discrete system of equations, a direct LU solver was used. We use the Python implementation of MMA from Deetman (2024).

E.1 Initialization

The baseline is initialized with a uniform density field, where each pixel is set to the target volume fraction. To be consistent with the baseline and to start the optimization from a feasible point, we trained the NNs' parameters to generate a uniform gray density distribution before starting topology optimization (Zhang et al 2021). This can be achieved by solving Eq. (3), by setting $\rho_i = V_0$. This pretraining was performed with 300 iterations of Adam (Kingma and Ba 2015) (with a default learning rate of 0.001), yielding errors lower than 10^{-4}. Noteworthy, the cost of this operation is negligible since neither the expensive finite element analysis nor the adjoint analysis is conducted.

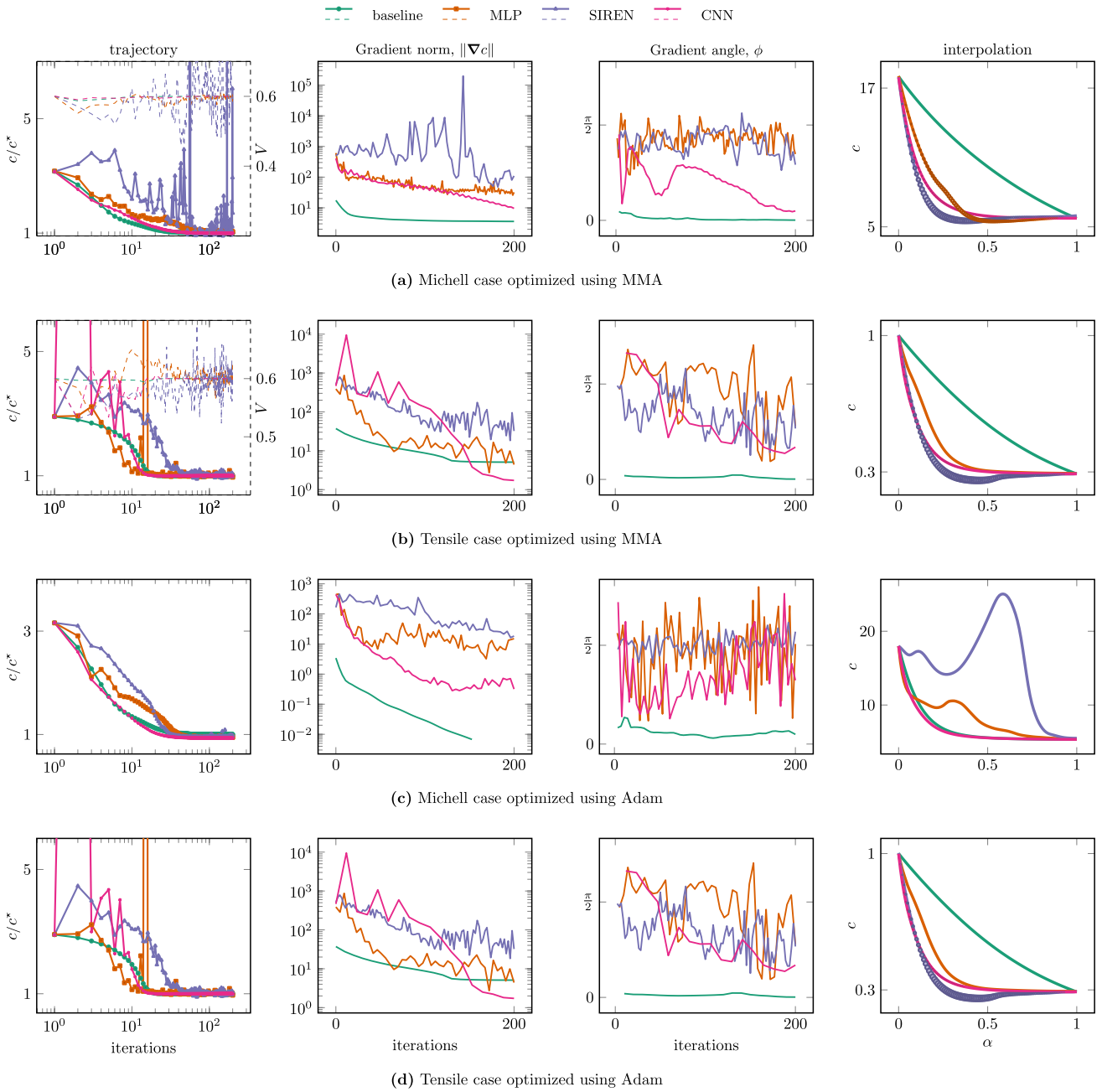
E.2 Thresholding designs

To obtain black-and-white designs—i.e., density values of either 0 or 1 for all finite elements—we use the algorithm described by Sigmund (Sigmund 2022). Briefly, if the number of elements/pixels is N, then the design is flattened and sorted in descending order based on the densities. Then, the number of pixels to be set to black N_p is obtained by

$$N_p = \frac{N\left(V_0 - 0.001\right)}{1 - 0.001}, \tag{E2}$$

where V_0 is the target volume fraction. Then, the discrete design is obtained by setting the first N_p values to 1 and all others to 0.001 (Sigmund 2022). If the volume fraction changes slightly, a new compliance value can be calculated

[12] These results are not shown in Fig. 5 because the number of parameters is much larger than those of the other CNN networks considered.



(a) Michell case optimized using MMA

(b) Tensile case optimized using MMA

(c) Michell case optimized using Adam

(d) Tensile case optimized using Adam

Fig. 12 Comparison of optimizer's trajectory on the neural landscape against the conventional landscape (with penalization $p = 3$ and target volume fraction of 60%). The first column shows the compliance c normalized by the baseline solution $c^\star$ and volume fraction V, as functions of the optimization iteration; the last column is the normalized compliance along interpolation between initial and final designs. The size of the dots indicate the amount of constraint violation; the L_2 -norm of the objective gradient and the angle between successive gradient vectors at each point along the optimizer's trajectory are shown in the middle two columns

as $c_{\text{new}} = c_{\text{th}} \times \frac{V_{\text{th}}}{V_0}$, where c_{th} is the compliance value of the thresholded design with a volume fraction V_{th}, which (slightly) violates the target volume fraction V_0. Typical designs after thresholding are shown in SI.

E.3 Hyperparameter tuning

We tuned the hyperparameters only at the lowest resolution (64×32) by performing 60 iterations with the optimizer, and used the hyperparameters that minimized the compliance for the optimization at higher resolutions. The hyperparameters chosen for the different methods were:

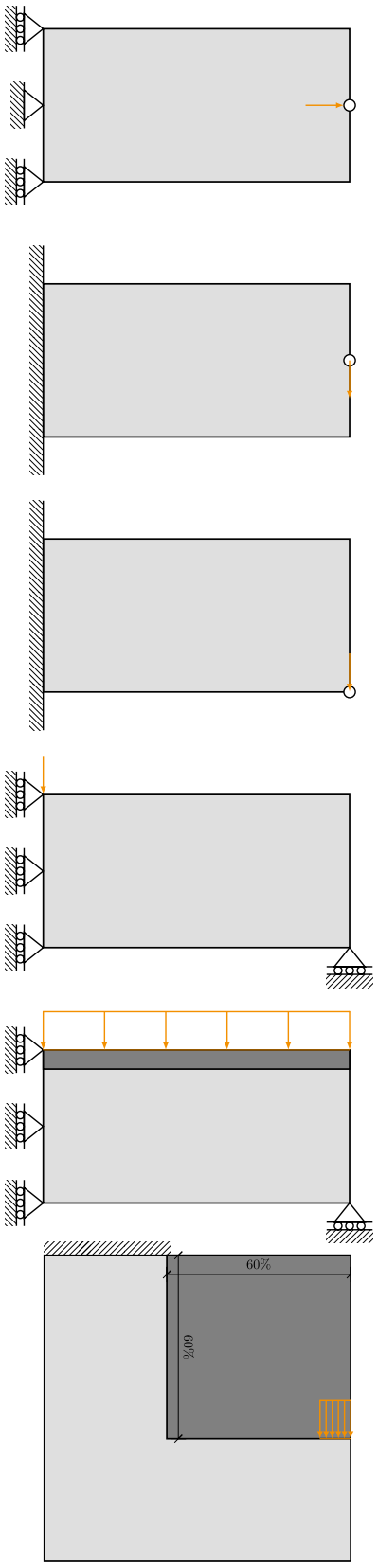


Fig. 13 Different boundary value problems used in this study. All loads are distributed (not concentrated) across a finite length. For the bridge case, the top few elements have been designated as a non-design domain with a constant material density of 1.0. For the L-shaped bracket, a square non-design region spanning 60% of the design area was set to have a constant material density of 0

1. MMA
   - Move limit m;
   - Asymptote initialization a;
   - Bounds on decision variables b (only for training neural networks)
2. Adam
   - Learning rate η;
   - Gradient clipping value g_c (i.e., scaling of the gradient vector if its norm is above this value).

In addition to these optimizer-specific hyperparameters, SIREN has an additional hyperparameter ω_0, which controls the frequencies that can be learned. All hyperparameters were tuned with the Optuna package (Akiba et al 2019) using the TPE algorithm. For the test cases used in the landscape visualization, the best hyperparameter values were obtained after 25 outer iterations.

E.4 Performance profiles

Performance profiles (Dolan and Moré 2002) are used for statistically comparing s solvers (or methods) on t test cases. To construct a performance profile, the performance ratio for solver i on case j is defined as

$$r_{ij} = \frac{M_{ij}}{\min(M_{1j}, M_{2j}, \cdots, M_{nj})}, \qquad M_{ij} \geq 0,$$

where M is any scalar metric of interest (e.g., best objective value, converged iteration, or compliance of the thresholded design) that is to be compared. This ratio indicates the performance relative to the best solver for that particular case. Next, the solver is treated as a "winner" for a particular case if its performance is within a tolerance of the best solver according to the following function:

$$k(r_{ij}, \tau) = \begin{cases} 1 & r_{ij} \leq \tau, \\ 0 & \text{otherwise}, \end{cases} \tag{E3}$$

where $\tau \geq 1$ is the tolerance factor. If $\tau = 1$, the allowable error is 0%, and only one solver is allowed to be the winner for a given test case. With increasing tolerance (from $1 \rightarrow \infty$), multiple solvers may qualify as winners. The performance profile is then the evolution of the percentage of test cases where the solver is a winner as the tolerance

is relaxed. Thus, the performance profile for the ith solver is given by

$$p_i(\tau) = \frac{\sum_j k(r_{ij}, \tau)}{m}, \tag{E4}$$

which denotes the probability that the solver's performance is within a factor τ of the best possible performance for all test cases. For all plots showing performance profiles, the allowed tolerance (in %) is used as the abscissas instead of τ.

E.5 Extension to other problems

Section 6 included two additional problems where we tested neural TO: thermal conduction optimization, and compliant mechanism design. Here, we provide additional details for replication of the results, and we also include another example involving stress-constrained volume minimization in appendix (E.5). The first two problems are structurally similar to compliance minimization in that they involve volume constraints, which are relatively straightforward to enforce. In contrast, the stress-constrained formulation introduces significant challenges, which we discuss in greater detail in appendix (E.5).

E.5.1 Volume-constrained problems: thermal compliance, and compliant mechanism design

For the thermal conduction problem, the goal is to optimize the material distribution to efficiently dissipate heat from a design domain to a sink. The material conductivity is set to 1.0, while the void or non-material regions have a conductivity of 0.001. The objective function $\mathcal{F}$ is expressed as $\mathcal{F} = \boldsymbol{P}^\intercal \boldsymbol{U}$, where $\boldsymbol{P} = \boldsymbol{F}$ represents the unit thermal load distributed throughout the domain. For this problem we set the target volume to $V_0 = 30\%$.

For the compliant mechanism design problem, the objective is to construct a mechanism where the force applied at the top-left node of the domain yields the maximum negative displacement at the top-right node. Here, the input spring stiffness (k_{in}) and input force are both set to 1, while the output spring stiffness (k_{out}) is set to 0.001. All entries in vector $\boldsymbol{P}$ are zeros except for the one corresponding to the output degree of freedom (the top-right node), which is set to one. The target volume is set to $V_0 = 40\%$.

We follow the same procedure as with compliance minimization, i.e., pretraining to uniform density initialization, hyperparameter tuning at 64×32 resolution, and testing at 544×272.[13] The results, as well as schematics of the boundary conditions, are shown in Fig. 8. We only used Adam as the optimizer since it was better than MMA for the compliance-minimization problem.

E.5.2 Stress-constrained optimization

We consider the following optimization problem from Verbart et al (2016):

$$\begin{aligned} \boldsymbol{\rho}^\star = \underset{\boldsymbol{\rho}\in\mathcal{D}}{\arg\min} \quad & V = \frac{1}{V_0}\sum_{i=1}^{N} v_i \rho_i, \\ \text{such that} \quad & \mathrm{g}_0(\boldsymbol{\rho}) = \Psi(\bar{\mathrm{g}}_i) \le 0, \\ & \boldsymbol{K}\boldsymbol{U}(\boldsymbol{\rho}) = \boldsymbol{F}, \\ & 0 \le \rho_i \le 1, \\ & i = \{1, \dots, N\}, \end{aligned} \tag{E5}$$

where the objective is to minimize the structural mass, and Ψ is an aggregation function that combines the element-wise local stress constraint values $\bar{\mathrm{g}}_i$ into a single global constraint. Each local constraint is defined as $\bar{\mathrm{g}}_i = \rho_i\left(\frac{\sigma_i}{\sigma_{max}} - 1\right)$, where σ_i denotes the von Mises stress at the centroid of the ith element. Note that the stress is computed assuming the full (solid) Young's modulus, rather than interpolated stiffness, to better reflect physical fidelity, i.e., we use the microscopic stress definition instead of the homogenized stress. We set the allowable stress limit to $\sigma_{\max} = 0.75$ with the force applied being 2.0[14], distributed over 5 nodes. The choice of aggregation function Ψ significantly influences the optimization results. We consider the lower-bounded Kreisselmeier–Steinhauser (KS) function (Ψ^{L}) (Verbart et al 2016):

$$\Psi^{\mathrm{L}}(\bar{\mathrm{g}}_i) = \frac{1}{P}\ln\left(\frac{1}{N}\sum_{i=1}^{N}\exp(P\bar{\mathrm{g}}_i)\right), \tag{E6}$$

where the parameter P controls the smoothness and closeness to the true maximum function—larger values of P result in a tighter approximation. We set $P = 10$ in all our experiments. The square domain was discretized with a $N = 96 \times 96$ regular mesh, with each element having unit dimensions. All other settings were kept similar to Verbart et al (2016).

As MMA has shown limited robustness even for enforcing simple volume constraints in neural TO, we adopt an alternative strategy: the augmented Lagrangian (AL) method (see Byrd et al (1994) for further details). The augmented Lagrangian function is defined as

$$\mathcal{L} = V + \lambda \mathrm{g}_0 + \mu \mathrm{g}_0^2, \tag{E7}$$

[13] Since the thermal problem has a square domain, we use the same number of elements for x and y directions i.e., we test at 544×544 mesh resolution

[14] All values in standard units

Table 2 Hyperparameters used for the stress-constrained L-shaped bracket problem

Parameterization	ALM			Adam	
	α	γ	n_i	η	g_c
Baseline	0.6	0.21	61	0.01	×
MLP	0.7	0.04	51	6.0×10^{-4}	0.01
SIREN	0.6	0.16	61	3.0×10^{-4}	0.01
CNN	0.99	0.02	31	7.7×10^{-3}	0.1

α and β are two hyperparameters of the modified AL method, behaving similarly to a learning rate and momentum term for the outer updates (i.e., updating the Lagrange multipliers). At each multiplier update, the inner problem is minimized using Adam, requiring n_i iterations with learning rate η. To ensure stability, gradients of the inner problem are clipped by their norm if above g_c before being passed to Adam

where λ and μ are dual variables with the same dimensionality as the number of constraints. The AL method consists of two nested updates: in the inner loop, the design variables are updated for fixed dual variables to minimize the AL function; in the outer loop, the dual variables are updated based on constraint violations. Initially, constraint violations are penalized lightly, but the penalization increases with infeasibility. As a result, the AL objective may increase during early iterations. Once constraints become feasible (or active, in the case of equality constraints), their contribution to the AL vanishes.

In our study, we employ the Adam optimizer for the inner updates, with gradient clipping applied: if the gradient norm exceeds a predefined threshold, it is rescaled to this threshold, a standard practice in neural network training to enhance stability. To further improve the robustness of AL in the context of neural networks, we adopt the adaptive dual update scheme proposed in Basir and Senocak (2022), governed by two tunable hyperparameters α and γ . The parameter α introduces a momentum-like effect by applying an exponentially weighted average to the squared constraint term. This running history is then combined with the hyperparameter γ, which plays a similar role to a learning rate, to update the penalty (μ). Finally, as in the standard AL method, the Lagrange multiplier (λ) is updated using the calculated penalty.

Hyperparameters for the outer optimization (e.g., α and γ), Adam's learning rate, the number of inner steps, and the gradient clipping threshold are tuned individually for each method; the tuned values are shown in Table 2. To evaluate constraint-handling capability, we consider an L-shaped bracket under stress constraints and optimize both the standard SIMP parameterization and three neural parameterizations using the modified AL framework.

While the AL method offers an alternative to MMA, it comes with its own limitations. As formulated, the AL method is primarily suited for equality constraints. Although slack variable formulations could extend its applicability to inequality constraints, such extensions require further research. Additionally, the AL method typically requires a large number of iterations for convergence, since each outer iteration involves an inner optimization loop of 50-60 steps. The method is also highly sensitive to hyperparameter choices, making it less robust in practical settings.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00158-025-04135-3.

Acknowledgements M.A.B. acknowledges the generous support by the Office of Naval Research through Grant No. N00014-21-1-2670. All authors sincerely thank Prof. Fred van Keulen, Prof. Mathijs Langelaar, Dr. Stijn Koppen, and Dr. Dirk Munro from the Mechanical Engineering faculty at TU Delft for the fruitful discussions and valuable feedback provided. S.M.S. appreciates the efforts of the Bessa research group members for proofreading and pointing out corrections, with special thanks to Gaweł Kus, Igor Kuszczak, and Shunyu Yin for their discussions regarding topology optimization.

Author contributions Alejandro Marcos Aragón and Miguel Anibal Bessa contributed to the study's conception and design. Material preparation, data collection and analysis were performed by Suryanarayanan Manoj Sanu. The first draft of the manuscript was written by Suryanarayanan Manoj Sanu and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data Availibility All key experimental details are provided in the article, appendix, and supplementary information. Codes and raw data for replication are available upon request by contacting the corresponding authors via email.

Declarations

Competing Interests The authors declare no competing interests (beyond the disclosed funding) relevant to this article.

Replication of results All details pertaining to the conducted experiments are given in the appendix and the supplementary information document provided alongside the paper.

References

Aage N, Andreassen E, Lazarov BS et al (2017) Giga-voxel computational morphogenesis for structural design. Nature 550(7674):84–86. https://doi.org/10.1038/nature23911

Akiba T, Sano S, Yanase T, et al (2019) Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2623–2631

Allen M, Maute K (2005) Reliability-based shape optimization of structures undergoing fluid-structure interaction phenomena. Comput Methods Appl Mech Eng 194(30–33):3472–3495. https://doi.org/10.1016/j.cma.2004.12.028

Andreassen E, Clausen A, Schevenels M et al (2011) Efficient topology optimization in matlab using 88 lines of code. Struct Multidiscip Optim 43:1–16. https://doi.org/10.1007/s00158-010-0594-7

Banga S, Gehani H, Bhilare S, et al (2018) 3d topology optimization using convolutional neural networks. arXiv preprint arXiv:1808.07440

Basir S, Senocak I (2022) Physics and equality constrained artificial neural networks: Application to forward and inverse problems with multi-fidelity data fusion. J Comput Phys 463:111301. https://doi.org/10.1016/j.jcp.2022.111301, https://www.sciencedirect.com/science/article/pii/S0021999122003631

Benbarka N, Höfer T, Zell A, et al (2022) Seeing implicit neural representations as fourier series. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp 2041–2050

Bendsøe MP, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng 71:197–224. https://doi.org/10.1016/0045-7825(88)90086-2

Bendsøe MP, Sigmund O (2004) Topology Optimization. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-05086-6

Bendsøe MP, Guedes JM, Plaxton S et al (1996) Optimization of structure and material properties for solids composed of softening material. Int J Solids Struct 33(12):1799–1813. https://doi.org/10.1016/0020-7683(95)00121-2

Berzins A, Radler A, Volkmann E, et al (2024) Geometry-informed neural networks. arXiv:2402.14009

Biswas A, Shapiro V, Tsukanov I (2004) Heterogeneous material modeling with distance fields. Computer Aided Geometric Design 21(3):215–242. https://doi.org/10.1016/j.cagd.2003.08.002

Both C, Dehmamy N, Yu R et al (2023) Accelerating network layouts using graph neural networks. Nat Commun 14(1):1560. https://doi.org/10.1038/s41467-023-37189-2

Bradbury J, Frostig R, Hawkins P, et al (2018) JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax

Buhl T, Pedersen C, Sigmund O (2000) Stiffness design of geometrically nonlinear structures using topology optimization. Struct Multidiscip Optim 19(2):93–104. https://doi.org/10.1007/s001580050089

Byrd RH, Nocedal J, Schnabel RB (1994) Representations of quasi-newton matrices and their use in limited memory methods. Math Program 63(1–3):129–156

Chandrasekhar A, Suresh K (2020) Tounn: Topology optimization using neural networks. Struct Multidiscip Optim 63(3):1135–1149. https://doi.org/10.1007/s00158-020-02748-4

Chandrasekhar A, Suresh K (2022) Approximate length scale filter in topology optimization using fourier enhanced neural networks. Comput Aided Des 150(C). https://doi.org/10.1016/j.cad.2022.103277

Chen W, Xia L, Yang J et al (2018) Optimal microstructures of elastoplastic cellular materials under various macroscopic strains. Mech Mater 118:120–132

Cheng G, Jiang Z (1992) Study on topology optimization with stress constraints. Eng Optim 20(2):129–148. https://doi.org/10.1080/03052159208941276

Chi H, Zhang Y, Tang TLE et al (2021) Universal machine learning for topology optimization. Comput Methods Appl Mech Eng 375. https://doi.org/10.1016/j.cma.2019.112739

Danilova M, Dvurechensky P, Gasnikov A, et al (2022) Recent Theoretical Advances in Non-Convex Optimization, Springer International Publishing, pp 79–163. https://doi.org/10.1007/978-3-031-00832-0_3

Deetman A (2024) Gcmma-mma-python: Python code of the method of moving asymptotes

Deng H, To AC (2020) Topology optimization based on deep representation learning (drl) for compliance and stress-constrained design. Comput Mech 66(2):449–469. https://doi.org/10.1007/s00466-020-01859-5

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Math Program 91(2):201–213. https://doi.org/10.1007/s101070100263

Doosti N, Panetta J, Babaei V (2021) Topology optimization via frequency tuning of neural design representations. In: Proceedings of the 6th Annual ACM Symposium on Computational Fabrication. Association for Computing Machinery, New York, NY, USA, SCF '21, https://doi.org/10.1145/3485114.3485124

Dupuis B, Jacot A (2021) Dnn-based topology optimisation: Spatial invariance and neural tangent kernel. Adv Neural Inf Process Syst 34:27659–27669

Duysinx P, Bendsøe MP (1998) Topology optimization of continuum structures with local stress constraints. Int J Numer Meth Eng 43(8):1453–1478

Fritzen F, Xia L, Leuschner M et al (2016) Topology optimization of multiscale elastoviscoplastic structures. Int J Numer Meth Eng 106(6):430–453

Goodfellow IJ, Vinyals O (2015) Qualitatively characterizing neural network optimization problems. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, arXiv:abs/1412.6544

Guo X, Zhang W, Zhong W (2014) Doing topology optimization explicitly and geometrically—a new moving morphable components based framework. Journal of Applied Mechanics 81(8). https://doi.org/10.1115/1.4027609

Halle A, Campanile LF, Hasse A (2021) An artificial intelligence-assisted design method for topology optimization without pre-optimized training data. Appl Sci 11(19):9041. https://doi.org/10.3390/app11199041

Hennigan T, Cai T, Norman T, et al (2020) Haiku: Sonnet for JAX. http://github.com/deepmind/dm-haiku

Herrmann L, Sigmund O, Li VM, et al (2024) On neural networks for generating better local optima in topology optimization. Structural and Multidisciplinary Optimization 67(11). https://doi.org/10.1007/s00158-024-03908-6

Heydaribeni N, Zhan X, Zhang R et al (2024) Distributed constrained combinatorial optimization leveraging hypergraph neural networks. Nature Machine Intelligence 6(6):664–672. https://doi.org/10.1038/s42256-024-00833-7

Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Hoyer S, Sohl-Dickstein J, Greydanus S (2019) Neural reparameterization improves structural optimization. In: NeurIPS 2019 Workshop on Solving Inverse Problems with Deep Networks

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pmlr, pp 448–456

Jeong H, Batuwatta-Gamage C, Bai J et al (2023) A complete physics-informed neural network-based framework for structural topology optimization. Comput Methods Appl Mech Eng 417:116401. https://doi.org/10.1016/j.cma.2023.116401

Kallioras NA, Lagaros ND (2020) Dl-scale: a novel deep learning-based model order upscaling scheme for solving topology optimization problems. Neural Comput Appl 33(12):7125–7144. https://doi.org/10.1007/s00521-020-05480-8

Kallioras NA, Kazakis G, Lagaros ND (2020) Accelerated topology optimization by means of deep learning. Struct Multidiscip Optim 62(3):1185–1212. https://doi.org/10.1007/s00158-020-02545-z

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, arXiv:abs/1412.6980

Kirsch U (1990) On singular topologies in optimum structural design. Structural Optimization 2(3):133–142. https://doi.org/10.1007/bf01836562

Kirsch U, Papalambros P (2001) Structural reanalysis for topological modifications - a unified approach. Struct Multidiscip Optim 21(5):333–344. https://doi.org/10.1007/s001580100112

Kreissl S, Pingen G, Maute K (2011) Topology optimization for unsteady flow. Int J Numer Meth Eng 87(13):1229–1253. https://doi.org/10.1002/nme.3151

Le C, Bruns TE, Tortorelli DA (2012) Material microstructure optimization for linear elastodynamic energy wave management. J Mech Phys Solids 60(2):351–378. https://doi.org/10.1016/j.jmps.2011.09.002

Li H, Xu Z, Taylor G, et al (2018) Visualizing the loss landscape of neural nets. Advances in neural information processing systems 31

Lu Z, Pu H, Wang F, et al (2017) The expressive power of neural networks: A view from the width. Advances in neural information processing systems 30

Mai HT, Lieu QX, Kang J et al (2022) A novel deep unsupervised learning-based framework for optimization of truss structures. Engineering with Computers 39(4):2585–2608. https://doi.org/10.1007/s00366-022-01636-3

Min S, Kikuchi N, Park YC et al (1999) Optimal topology design of structures under dynamic loads. Structural Optimization 17(2–3):208–218. https://doi.org/10.1007/bf01195945

Nakshatrala PB, Tortorelli DA, Nakshatrala K (2013) Nonlinear structural design using multiscale topology optimization. part 1: Static formulation. Comput Methods Appl Mech Eng 261:167–176

Norder L, Yin S, de Jong MH et al (2025) Pentagonal photonic crystal mirrors: scalable lightsails with enhanced acceleration via neural topology optimization. Nat Commun 16(1):2753

Papadopoulos IPA, Farrell PE, Surowiec TM (2021) Computing multiple solutions of topology optimization problems. SIAM J Sci Comput 43(3):A1555–A1582

Poulsen TA (2002) Topology optimization in wavelet space. Int J Numer Meth Eng 53:567–582. https://doi.org/10.1002/NME.285

Qian C, Ye W (2020) Accelerating gradient-based topology optimization design with dual-model artificial neural networks. Struct Multidiscip Optim 63(4):1687–1707. https://doi.org/10.1007/s00158-020-02770-6

Qian W, Xu Y, Li H (2022) A topology description function-enhanced neural network for topology optimization. Computer-Aided Civil and Infrastructure Engineering 38(8):1020–1040. https://doi.org/10.1111/mice.12933

Qian X (2013) Topology optimization in b-spline space. Comput Methods Appl Mech Eng 265:15–35. https://doi.org/10.1016/j.cma.2013.06.001

Rahaman N, Baratin A, Arpit D, et al (2019) On the spectral bias of neural networks. In: International conference on machine learning, PMLR, pp 5301–5310

Raponi E, Bujny M, Olhofer M et al (2019) Kriging-assisted topology optimization of crash structures. Comput Methods Appl Mech Eng 348:730–752. https://doi.org/10.1016/j.cma.2019.02.002

Rojas-Labanda S, Stolpe M (2015) Benchmarking optimization solvers for structural topology optimization. Struct Multidiscip Optim 52(3):527–547. https://doi.org/10.1007/s00158-015-1250-z

Rozvany GIN (2009) A critical review of established methods of structural topology optimization. Struct Multidiscip Optim 37(3):217–237. https://doi.org/10.1007/s00158-007-0217-0

Schmidt RM, Schneider F, Hennig P (2021) Descending through a crowded valley-benchmarking deep learning optimizers. In: International Conference on Machine Learning, PMLR, pp 9367–9376

Sigmund O (2022) On benchmarking and good scientific practise in topology optimization. Struct Multidiscip Optim 65:315. https://doi.org/10.1007/s00158-022-03427-2

Sigmund O, Maute K (2013) Topology optimization approaches: A comparative review. Struct Multidiscip Optim 48:1031–1055. https://doi.org/10.1007/s00158-013-0978-6

Sigmund O, Aage N, Andreassen E (2016) On the (non-)optimality of michell structures. Struct Multidiscip Optim 54(2):361–373. https://doi.org/10.1007/s00158-016-1420-7

Silva ECN, Kikuchi N (1999) Design of piezoelectric transducers using topology optimization. Smart Mater Struct 8(3):350–364. https://doi.org/10.1088/0964-1726/8/3/307

Sitzmann V, Martel JN, Bergman AW, et al (2020) Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems 2020-December. https://doi.org/10.48550/arxiv.2006.09661

Sosnovik I, Oseledets I (2019) Neural networks for topology optimization. Russ J Numer Anal Math Model 34(4):215–223

Stanley KO (2007) Compositional pattern producing networks: A novel abstraction of development. Genet Program Evolvable Mach 8(2):131–162. https://doi.org/10.1007/s10710-007-9028-8

Stolpe M (2003) On Models and Methods for Global Optimization of Structural Topology. Matematik, Stockholm

Stolpe M, Svanberg K (2001) An alternative interpolation scheme for minimum compliance topology optimization. Struct Multidiscip Optim 22(2):116–124. https://doi.org/10.1007/s001580100129

Strümpler Y, Postels J, Yang R, et al (2022) Implicit neural representations for image compression. In: Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI. Springer-Verlag, Berlin, Heidelberg, pp 74–91, https://doi.org/10.1007/978-3-031-19809-0_5

Svanberg K (1987) The method of moving asymptotes–a new method for structural optimization. Int J Numer Meth Eng 24(2):359–373. https://doi.org/10.1002/nme.1620240207

Svanberg K (2002) A class of globally convergent optimization methods based on conservative convex separable approximations. SIAM J Optim 12(2):555–573. https://doi.org/10.1137/s1052623499362822

Ulyanov D, Vedaldi A, Lempitsky VS (2017) Deep image prior. International Journal of Computer Vision 128:1867 – 1888. https://api.semanticscholar.org/CorpusID:4531078

Verbart A, Langelaar M, Fv K (2016) A unified aggregation and relaxation approach for stress-constrained topology optimization. Struct Multidiscip Optim 55(2):663–679. https://doi.org/10.1007/s00158-016-1524-0

Wang F, Lazarov BS, Sigmund O (2010) On projection methods, convergence and robust formulations in topology optimization. Struct Multidiscip Optim 43(6):767–784. https://doi.org/10.1007/s00158-010-0602-y

Wang S, Wang MY (2005) Radial basis functions and level set method for structural topology optimization. Int J Numer Meth Eng 65(12):2060–2090. https://doi.org/10.1002/nme.1536

White DA, Stowell ML, Tortorelli DA (2018) Toplogical optimization of structures using fourier representations. Struct Multidiscip Optim 58(3):1205–1220. https://doi.org/10.1007/s00158-018-1962-y

Woldseth RV, Aage N, Bærentzen JA, et al (2022) On the use of artificial neural networks in topology optimisation. Structural and Multidisciplinary Optimization 65(10). https://doi.org/10.1007/s00158-022-03347-1

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82. https://doi.org/10.1109/4235.585893

Xia L, Breitkopf P (2014) Concurrent topology optimization design of material and structure within fe^2 nonlinear multiscale analysis framework. Comput Methods Appl Mech Eng 278:524–542. https://doi.org/10.1016/j.cma.2014.05.022

Xia L, Da D, Yvonnet J (2018) Topology optimization for maximizing the fracture resistance of quasi-brittle composites. Comput

Methods Appl Mech Eng 332:234–254. https://doi.org/10.1016/j.cma.2017.12.021

Xu S, Cai Y, Cheng G (2009) Volume preserving nonlinear density filter based on heaviside functions. Struct Multidiscip Optim 41(4):495–505. https://doi.org/10.1007/s00158-009-0452-7

Xue L, Liu J, Wen G, et al (2021) Efficient, high-resolution topology optimization method based on convolutional neural networks. Frontiers of Mechanical Engineering 2021 16:1 16:80–96. https://doi.org/10.1007/S11465-020-0614-2

Yoon GH (2022) Transient sensitivity analysis and topology optimization of particle suspended in transient laminar fluid. Comput Methods Appl Mech Eng 393:114696. https://doi.org/10.1016/j.cma.2022.114696

Yoon GH, Kim YY (2005) Element connectivity parameterization for topology optimization of geometrically nonlinear structures. Int J Solids Struct 42(7):1983–2009. https://doi.org/10.1016/j.ijsolstr.2004.09.005

Yoon GH, Jensen JS, Sigmund O (2006) Topology optimization of acoustic-structure interaction problems using a mixed finite element formulation. Int J Numer Meth Eng 70(9):1049–1075. https://doi.org/10.1002/nme.1900

Yoon GH, Choi H, Hur S (2018) Multiphysics topology optimization for piezoelectric acoustic focuser. Comput Methods Appl Mech Eng 332:600–623. https://doi.org/10.1016/j.cma.2017.12.002

Yoshimura M, Shimoyama K, Misaka T et al (2016) Topology optimization of fluid problems using genetic algorithm assisted by the kriging model. Int J Numer Meth Eng 109(4):514–532. https://doi.org/10.1002/nme.5295

Yu M, Ruan S, Wang X et al (2019) Topology optimization of thermal-fluid problem using the mmc-based approach. Struct Multidiscip Optim 60(1):151–165. https://doi.org/10.1007/s00158-019-02206-w

Zehnder J, Li Y, Coros S, et al (2021) Ntopo: Mesh-free topology optimization using implicit neural representations. In: Ranzato M, Beygelzimer A, Dauphin Y, et al (eds) Advances in Neural Information Processing Systems, vol 34. Curran Associates, Inc., pp 10368–10381, https://proceedings.neurips.cc/paper_files/paper/2021/file/55d99a37b2e1badba7c8df4ccd506a88-Paper.pdf

Zhang S, Norato JA, Gain AL et al (2016) A geometry projection method for the topology optimization of plate structures. Struct Multidiscip Optim 54(5):1173–1190. https://doi.org/10.1007/s00158-016-1466-6

Zhang Z, Li Y, Zhou W et al (2021) Tonr: An exploration for a novel way combining neural network with topology optimization. Comput Methods Appl Mech Eng 386:114083. https://doi.org/10.1016/j.cma.2021.114083