Topology Optimization via Frequency Tuning of Neural Design Representations

Nikan Doosti
Max Planck Institute for Informatics
Saarbrücken, Germany
mdoosti@mpi-inf.mpg.de

Julian Panetta
University of California, Davis
Davis, United States
jpanetta@ucdavis.edu

Vahid Babaei
Max Planck Institute for Informatics
Saarbrücken, Germany
vbabaei@mpi-inf.mpg.de

ABSTRACT

Structural topology optimization seeks to distribute material throughout a design domain in a way that maximizes a certain performance goal. In this work, we solve the topology optimization problem by parameterizing the designs via recently introduced coordinate-based neural networks. Specifically, we show that networks with Fourier feature mapping can achieve state-of-the-art performance. Our method enables the realization of a range of designs using a single mesh via tuning the frequency content of the solutions independently of the finite element discretization grid. This frequency control offers attractive properties, such as mesh-independent results and sub-pixel filtering that leads to appropriate designs for upsampling. We demonstrate our method on the compliance minimization problem, optimizing for the stiffest possible structure within a weight budget for a prescribed set of loads.

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**; **Neural networks**; • **Applied computing** → *Computer-aided design.*

KEYWORDS

Topology optimization, neural networks, generative design

ACM Reference Format:
Nikan Doosti, Julian Panetta, and Vahid Babaei. 2021. Topology Optimization via Frequency Tuning of Neural Design Representations. In *Symposium on Computational Fabrication (SCF '21), October 28–29, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485114.3485124

1 INTRODUCTION

The build volume and resolution of advanced manufacturing systems, such as multi-material 3D printers, are continuously growing [Regehly et al. 2020]. The rapid improvement in hardware technologies has important consequences for computational design algorithms. Advances in these algorithms have not kept pace with the exploding design space, resulting in an increasing gap between *what we can compute and what we can fabricate.* An important and promising direction to address this swiftly widening gap is to explore novel design representations.

In this work, we demonstrate the power of recent coordinate-based neural networks [Mildenhall et al. 2020; Tancik et al. 2020] as a design representation for topology optimization (TO). Topology optimization for structural design is a method for optimally distributing material throughout a design in order to maximize a certain performance metric. Our design algorithm optimizes over the space defined by this naturally differentiable representation using exact gradients of the relevant structural performance metrics evaluated using a high-performance multigrid elasticity solver and analytical sensitivity analysis. We demonstrate that the coordinate-based network representation's inherent support for controlling the frequency spectrum of the output design provides important benefits for TO. On the one hand, by fixing the design frequency, we are able to achieve *mesh-independent* results that are insensitive to refinement of the finite-element discretization grid used for simulation without resorting to standard TO smoothing filters. On the other hand, by tuning this frequency, we can achieve a range of designs using a *single* mesh including those unattainable by classic counterparts.

Our method leads to appealing properties for design *upsampling*. Aage et al. [2017] demonstrate how executing TO at high resolutions leads to designs with a variety of intricate, multi-scale structures. In order to optimize designs with roughly one billion voxels, the immense computational effort is distributed over 8,000 CPUs on a supercomputer. Liu et al. [2018] show that designs of similar resolutions and complexities can be optimized on a single workstation by employing advanced data structures and solvers. However, today's commercially available, lab-friendly, multi-material 3D printers can already fit *tera*-voxel designs within their build volumes [Stratasys 2020]. It is clear that neither vast computing resources nor complex software engineering can feasibly empower current algorithms to scale to this resolution, let alone to the ever increasing manufacturing capabilities on the horizon. Therefore, design upsampling remains a simple yet practical way of creating the necessary content for high-resolution manufacturing devices. As we will show, selecting the proper frequency for a design at low resolution has a significant effect on the compliance of the upsampled design.

2 RELATED WORK

Design representation. In computational design problems, discovering the proper *design representation* is often the crucial step that enables a practical algorithm. Such representations can be nonobvious, and the computational design and fabrication community has exploited a surprisingly creative and diverse range of design representations spanning from offset surfaces [Musialski et al. 2015], to orthogonal geodesic nets [Rabinovich et al. 2018], to Chebyshev nets [Garg et al. 2014]. These representations, however, tend to be highly problem-specific, failing to generalize to other tasks. Accordingly, for many applications, voxel grids remain the most popular representation despite the notoriously poor scaling of their memory requirements. Applying this discrete representation to design heterogeneous objects at high resolutions, as needed to fully leverage digital fabrication hardware, leads to inverse problems with a large, typically prohibitive number of variables. A particularly versatile, resolution-independent classical design parameterization is through *explicit* functional representations [Kou and Tan 2007]. In this type of representation, a design attribute v at position (x, y, z) is queried from an explicit analytical function $v = f(x, y, z)$ [Shin and Dutta 2001]. This method has been successfully applied for continuous modification of heterogeneous designs, such as objects with functionally graded material (FGM) [Elishakoff et al. 2005]. Unfortunately, despite their flexibility, these representations have generally been limited to hand-tuned analytical functions with limited ability to capture certain important material distribution features such as sharp transitions.

Coordinate-based neural networks. Recently a few related explicit functional representations called coordinate-based neural networks have emerged from computer vision and graphics. These representations, capable of parameterizing different signals, such as as shapes [Chen and Zhang 2019; Park et al. 2019] and scenes [Jiang et al. 2020; Mildenhall et al. 2020; Sitzmann et al. 2020], have achieved remarkable success in numerous application domains. There are two particularly successful coordinate-based networks both integrating a Fourier transform. Sitzmann et al. [2020] use sinusoidal activation functions in SIREN, whereas Mildenhall et al. [2020] pass the input through a Fourier feature layer. Both methods vastly improve the representation capacity for high-frequency signals.

Deep learning and computational design. Deep learning has recently shown promise for advancing computational design algorithms on two fronts. In the first line of work, it has been used to develop differentiable surrogate models capable of accurately predicting the performance of candidate designs at a fraction of the cost of a full simulation [Baque et al. 2018; Umetani and Bickel 2018; Wang and Shan 2006]. Machine learning techniques have demonstrated success in generating approximate solutions to problems such as fluid simulation [Tompson et al. 2017], elasticity [Zhang et al. 2016], and even general partial differential equations [Li et al. 2020]. The second line of work, closest to our own, employs neural networks as a design representation to, oftentimes, replace the standard regular grid of variables used in density-based TO [Chandrasekhar and Suresh 2021; Hoyer et al. 2019]. Concurrent work [Zehnder et al. 2021] applies both approaches to minimum compliance TO, representing both the optimal design and its static equilibrium under the applied loads with coordinate-based networks—thereby achieving a fully mesh-free algorithm at the cost of an expensive inner optimization implementing the equilibrium solve. While we rely on a computational mesh to efficiently evaluate compliance, we show that our method can still achieve mesh-independent solutions. Furthermore, compared to Zehnder et al. [2021], we investigate the effect of frequency tuning of the coordinate-based networks in topology optimization and demonstrate the importance of performing the optimization steps in the underlying design space rather than in density space to reap its full benefits.

3 BACKGROUND

3.1 Compliance minimization

We showcase our optimal design framework on the popular minimum compliance TO problem for linear elasticity: determining the spatial distribution of a fixed budget of material that achieves the stiffest possible structure for a prescribed loading scenario. We employ the standard Solid Isotropic Material with Penalization (SIMP) approach [Bendsoe and Sigmund 2013], which casts the TO problem as the optimization of a *density* scalar field $\rho : \mathbb{R}^n \to [0, 1]$ that in turn determines both the mass and stiffness of the material at each point. The TO algorithm is designed to ultimately converge to an essentially binary design, with ρ assuming a value of either nearly 0 (free space) or 1 (full fabrication material) at each point. However, intermediate density values are generated during the course of optimization to enable the use of gradient-based optimization.

Under the SIMP method, the total volume and elasticity tensor field (material stiffness) of a design specified by density field ρ are *defined* as:

$$V(\rho) := \int_\Omega \rho \, \mathrm{d}\mathbf{x}, \quad C(\rho) := (\epsilon + (1-\epsilon)\rho^p) C^{\text{base}},$$

where Ω is the design domain, C^{base} is the elasticity tensor of the full printing material, p is the SIMP penalty parameter that we fix at $p = 3$ in our experiments, and $\epsilon = 10^{-4}$ is a small constant used to prevent the linear elasticity operator from being singular. This interpolated elasticity tensor field can then be used to simulate the structure's deformation under the prescribed loads and evaluate the *compliance* (work done by the loads), the performance metric of interest. We discretize the linear elasticity equation using finite elements, employing tensor-product polynomial basis functions on a regular grid. The results shown in this paper all use bilinear (2D) and trilinear basis functions (3D) $\overrightarrow{\phi}_i$, but the code supports arbitrary polynomial degrees. The FEM equilibrium displacement vector $\mathbf{u}(\rho)$ for the design is found by solving the linear system:

$$K(\rho)\mathbf{u}(\rho) = \mathbf{f}, \quad K(\rho) := \sum_e (\epsilon + (1-\epsilon)\rho(\mathbf{x}_e)^p) S_e^T K_0 S_e,$$
$$[K_0]_{ab} := \int_{\mathcal{E}} \varepsilon(\overrightarrow{\phi}_a) : C^{\text{base}} : \varepsilon(\overrightarrow{\phi}_b) \, \mathrm{d}\mathbf{x},$$

where $K(\rho)$ is the *stiffness matrix*, and $\mathbf{f}$ is the load vector determined from the applied boundary conditions. K is assembled from the full-density per-element stiffness matrix K_0 (identical for all elements to the one for a reference element $\mathcal{E}$ since the simulation mesh is a regular grid), scaled according to the SIMP interpolation rule for the density evaluated at element center $\mathbf{x}_e$, and finally distributed to the appropriate entries of the sparse global stiffness matrix by the rectangular *selection matrix* S_e. This is the sparse binary matrix that, when applied to $\mathbf{u}$, extracts the displacements of the element's nodes as a small vector $\mathbf{u}_e := S_e\mathbf{u}$. We solve the linear system using a high-performance multigrid-preconditioned conjugate-gradient solver based on [Wu et al. 2016] in order to evaluate the compliance $J(\rho) := \mathbf{f} \cdot \mathbf{u}(\rho)$. We can finally formulate the minimum compliance TO problem as:

$$\begin{aligned} &\min_{\rho} J(\rho) \\ &\text{s.t.} \quad V(\rho) \le V_0 \\ &\qquad 0 \le \rho \le 1, \end{aligned} \tag{1}$$

where V_0 is the volume (material) budget.

The analytical gradients of volume and compliance with respect to the element center densities $\rho_e = \rho(\mathbf{x}_e)$ are:

$$\frac{\partial V}{\partial \rho_e} = |\mathcal{E}|, \quad \frac{\partial J}{\partial \rho_e} = -p(1-\epsilon)\rho^{p-1}\mathbf{u}_e^T K_0 \mathbf{u}_e. \tag{2}$$

3.2 Traditional density-based TO and filtering

The traditional approach for solving (1) is to represent the design directly using the vector of element-center densities ρ_e (defining a piecewise constant density field) and then apply a finite dimensional optimization algorithm to the resulting discrete minimization problem. A well-known artifact that immediately arises is *checkerboard* patterns (arrangements of voxels alternating between fully solid and fully void) due to the stiffness of these patterns being overestimated by the finite element simulation. The standard mitigation for this issue employed in the literature is to insert a *smoothing filter* [Bruns and Tortorelli 2001] into the representation that maps the grid of optimization variables (sometimes called *blueprint densities*) to a smoothed piecewise constant density field. This smoothing filter takes the form of a weighted average of blueprint densities over a neighborhood surrounding the output voxel.

Smoothing filters mitigate the most egregious mesh-dependence issues of the voxel-based representation, but the filters themselves are still mesh-dependent, and the frequency of the generated designs is limited by the resolution of the finite element grid. As we demonstrate in Section 5.2, simultaneously eliminating checkerboard artifacts while still generating designs with features nearing the resolution of the simulation grid is problematic.

4 METHOD

4.1 Neural design parameterization

Instead of parametrizing the density field ρ by a vector of element-center density variables, we employ a feedforward neural network F_θ with *Fourier feature* mapping [Tancik et al. 2020] to define a continuous function of the Cartesian coordinates $\mathbf{x} \in \mathbb{R}^n$ ($n \in 2, 3$) in the design domain:

$$\rho(\mathbf{x}) = F_\theta\left(\mathcal{F}_\sigma(\mathbf{x})\right) = s \circ f^L \circ g^{L-1} \circ f^{L-1} \circ \ldots \circ g^1 \circ f^1\left(\mathcal{F}_\sigma(\mathbf{x})\right) \tag{3a}$$

$$\mathcal{F}_\sigma(\mathbf{x}) = [\cos(2\pi\mathbf{B}\mathbf{x}), \sin(2\pi\mathbf{B}\mathbf{x})] \tag{3b}$$

$$f^l(\mathbf{x}^{l-1}) = \mathbf{W}^l\mathbf{x}^{l-1} + \mathbf{b}^l, \quad \forall\ 1 \le l \le L, \tag{3c}$$

The activation function g^l is the well-known rectified linear unit (ReLU). The last activation function s is a sigmoid function that helps push the densities toward binary values (the standard "Heaviside projection filter" used in TO). The weights $\mathbf{W}^l$ and biases $\mathbf{b}^l$ at all layers (1 to L) make up the network's parameters θ, which are the unknowns we seek. The peculiarity of this network comes from the Fourier feature mapping operator $\mathcal{F}(\cdot)$. The matrix $\mathbf{B} \in \mathbb{R}^{d\times n}$ is sampled from a normal distribution with mean 0 and variance σ^2, with d being the size of the Fourier feature layer. Standard deviation σ is referred to as the Fourier feature scale, or *scale* for short, and is the most important hyperparameter for our method.

Tancik et al. [2020], relying on the Neural Tangent Kernel regression [Jacot et al. 2018], showed that the Fourier feature mapping, a special case of Fourier features in kernel regression [Rahimi et al. 2007], enables tuning the range of frequencies that can be learned by the network. So far, the frequency tuning in explicit neural representations has received little attention, only being tuned to avoid overfitting or underfitting. Only in a concurrent work, Dupuis and Jacot [2021] lay the theoretical ground of the relationship between classic filtering in SIMP and the filtering induced by the neural representation.

4.2 Neural topology optimization

Now we describe our fully differentiable design pipeline for the minimum compliance TO problem (Figure 1). We initialize weights θ of the explicit neural design representation (ENDR) so that it generates a uniform *gray* design that exactly satisfies the volume constraint, a common initialization used in TO. We use orthogonal random initialization (Saxe et al. [2013]) for weights at all layers except the last, similar to Hoyer et al. [2019]. In the last layer, we set the weight close to zero (randomly sampling from a normal distribution with mean 0 and standard deviation 10^{-4}) and set the biases to V_0 so that they generate the desired uniform gray level (as in Zehnder et al. [2021]). This initialization strikes a balance between a being a favorable starting point for network training and satisfying the problem-specific volume constraint.

At each iteration, a candidate design is evaluated by querying the density from the ENDR at the simulation grid element centers $\mathbf{x}_e$ and running our multigrid FEM solver to compute compliance. The analytical gradients of compliance and volume are then used to update the ENDR weights θ.

Note that our self-supervised pipeline does *not* depend on labeled data; the network weights are optimized to directly minimize compliance and satisfy the volume constraint. In contrast to some other efforts employing neural parameterization for TO that rely entirely on automatic differentiation to differentiate the performance metric with respect to the network parameters [Hoyer et al. 2019], we follow Chandrasekhar and Suresh [2021] and use efficient analytical formulas (2) for the derivatives of volume and compliance with respect to the sampled densities. We then backpropagate these these derivatives through the network to obtain the necessary gradients with respect to θ using automatic differentiation $\left(\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial \rho}\frac{\partial \rho}{\partial \theta}\right)$.

4.3 Enforcing constraints

Our ENDR enforces the pointwise density bounds by construction due to the sigmoid activation function s it applies to its output. The upper bound on volume is a nonlinear inequality constraint in our representation that requires more careful consideration.

We obtained our best results by optimizing our model using the popular algorithm Adam [Kingma and Ba 2014], which cannot enforce such constraints directly. Hoyer et al. [2019] proposed globally biasing the output densities to manually enforce the target volume at every step as an equality constraint. We found this strategy degraded the optimizer's convergence and tended to introduce

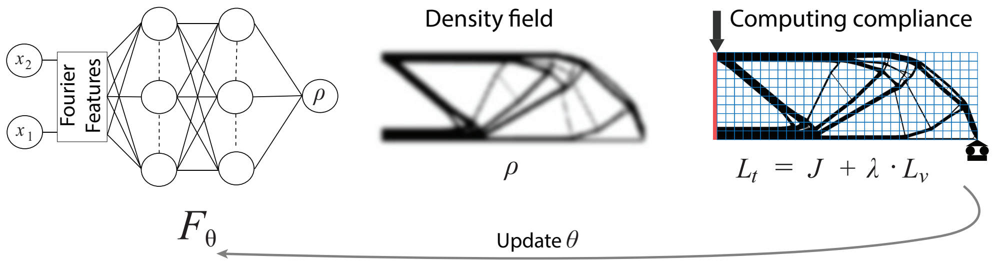

Figure 1: A single iteration of our end-to-end differentiable topology optimization via explicit design representation networks.

artifacts into the design. The Optimality Criteria (OC) algorithm is a longstanding heuristic alternative to gradient-descent-style optimizers for TO that also globally adjusts the design at each step to enforce the volume constraint [Bendsoe and Sigmund 2013]. While it is usually quite effective when operating on traditional density-based representations, we found it not to perform well for the highly nonlinear ENDR (Section 6).

We instead enforce the volume bound as a soft constraint, adding a one-sided loss term L_v to the compliance objective to form a total loss function L_t :

$$L_v = \max(-\log(1 + V_0 - V(\rho)), 0) \tag{4a}$$

$$L_t = J + \lambda \cdot L_v, \tag{4b}$$

where weight λ controls how precisely the volume constraint is enforced. We update λ at each iteration to balance the objective terms, but the gradient of L_t is computed treating λ as a constant. We recommend using $\lambda = \min(J/L_v, J_1/3L_{v_1})$, where J_1 and L_{v_1} are the compliance and volume loss after the first iteration. In our experiments, this scheme generally keeps the volume constraint violation below 10^{-4} even in upsampled designs.

5 EVALUATION

5.1 Experimental settings

We use $L = 4$ layers in all experiments. Unless explicitly stated otherwise, we use 256 and 512 neurons per layer for 2D and 3D examples, respectively, and $d = 1024$ Fourier feature samples. We use Adam [Kingma and Ba 2014] to train the network, and we set the learning rate to 10^{-5} in 2D and 3×10^{-4} in 3D. Models are trained for 5000 iterations unless otherwise specified. The minimum compliance problem is solved for an isotropic base material C^{base} with Poisson's ratio $\nu = 0.3$ and Young's modulus $E = 1$. We do not use any regularizations, learning rate schedulers, or other adaptive schemes to accelerate convergence.

5.2 Frequency priors for generative design

The central observation of our work is that the scale hyperparameter σ has a profound effect in the context of neural topology optimization. Different values of σ bake different frequency priors into the density field representation and lead to different (locally) optimal designs. As we will see, this capacity to control the design's spatial frequency continuously and independently of the grid resolution proves extremely useful in creating designs beyond the reach of traditional topology optimization.

In Figure 2, we demonstrate the effect of tuning σ for a fixed boundary condition and simulation grid. Increasing σ admits higher frequencies in the design representation, prompting the optimizer to generate more intricate results with finer branching features. We note that the stiffest design is achieved by the highest scale parameter, though all solutions provide good stiffness apart from the overly coarsened $\sigma = 2$ design. Additionally, all solutions, even those obtained for large σ do not suffer from *overfitting*. In simple image fitting experiments, Tancik et al. [2020] have shown that increasing the scale leads to over-fitting the input signal, which manifests as noise when sampling the ENDR at a perturbation of the coordinates used for training. In our experiments, however, the TO results do not suffer this issue. We upsampled our solution for $\sigma = 13$ at $20\times$ resolution along each dimension and found that the design did not change significantly in compliance, topology, or appearance.

Subvoxel filtering via frequency control. As discussed in Section 3.2, the complexity of designs produced by traditional grid-based TO algorithms is tied to the resolution of the simulation grid and the radius of the smoothing parameter (Figure 3). As we show in Figure 3, our method's scale parameter σ has an effect very similar to the radius parameter for a discrete filter: larger scales correspond to smaller filter radii and thus more detailed designs. But, via σ, we can *continuously* control the frequency of the output design in a completely mesh-indepedent way while using a fixed simulation grid. Most importantly, for sufficiently high σ, we obtain highly detailed designs that correspond to a *subvoxel* smoothing filter radius and yet do not suffer from checkerboarding. As we will see, these subvoxel solutions can be excellent candidates for design upsampling.

Mesh-independent solutions. In classic TO methods, the well-known issue of mesh-dependent solutions has been mitigated with several different techniques like perimeter control [Petersson 1999], constraining the density gradients [Borrvall 2001], or, most popularly, density filters. Using ENDR, a fixed σ achieves very similar designs regardless of the underlying resolution; fixing scale σ limits the complexity introduced by the increase in mesh resolution. Figure 4 visualizes the solutions to the same TO problem solved at three different mesh resolutions with the same scale, and we observe very similar structures in each.

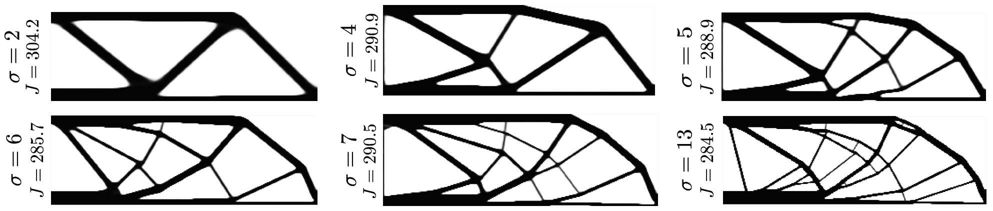

Figure 2: The effect of the scale hyperparameter σ on the quality and compliance (topology optimization of an MBB beam at 300×100 resolution). The σ range shown here, at this resolution, produces a diverse set of designs beyond which we did not obtain novel solutions.

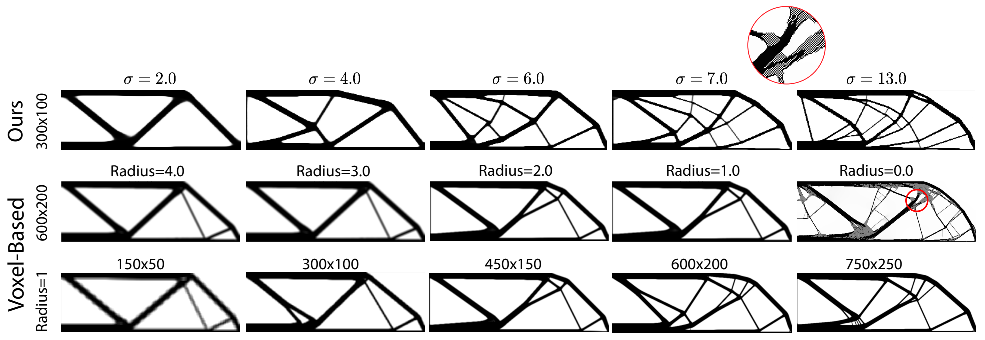

Figure 3: Subvoxel filtering via frequency control. We compare our ENDR-based optimizer with a traditional voxel-based design representation employing a smoothing filter whose radius is expressed in units of the grid's voxel width. Without the filter, a highly detailed design is generated, but it suffers from checkerboard patterns (red circle). Using a filter radius of only one voxel eliminates the checkerboard but severely compromises the design's complexity. Our frequency control scheme not only avoids explicit voxel-based smoothing, but it can achieve designs corresponding to subvoxel filtering.

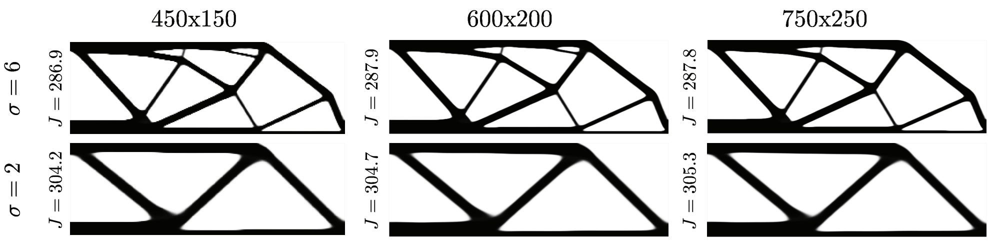

Figure 4: We demonstrate the mesh-independence of our method by visualizing density fields at different resolutions for a set of fixed σ values.

5.3 Design upsampling

Given the extreme resolution demands of today's hardware, upsampling is a simple and attractive way to generate designs for manufacturing. In practical scenarios like optimizing a structure to be 3D printed, we are limited by the time and/or memory budget for FEM computations. Ideally, we would like to fix a maximum *source* resolution based on a computation budget, compute an optimal design at this resolution, and upsample the design to the desired *target* resolution.

Synthesizing high-resolution designs from low-resolution ones is fundamentally difficult in topology optimization. The optimal design that can be resolved at a coarse resolution is likely to have an entirely different topology from the optimal design for a fine grid, and so a simple upsampling scheme is bound to be sub-optimal. Given that better designs at higher resolutions have more detailed structures, we propose to leverage the subvoxel filtering capability of our method and generate more detailed designs as candidates for upsampling. As we saw in Section 5.2, ENDR can generate highly detailed but checkerboard-free designs without increasing mesh resolution. This is particularly interesting since one can obtain a solution at a coarse resolution that qualitatively and quantitatively matches the result of a fine-resolution voxel-based design with a single-voxel smoothing radius.

Figure 5 demonstrates the potential of this approach. Our experiment follows these steps: (1) Train a model with scale σ at a chosen coarse *source resolution* grid$_s$ to obtain density field ρ_S. (2)

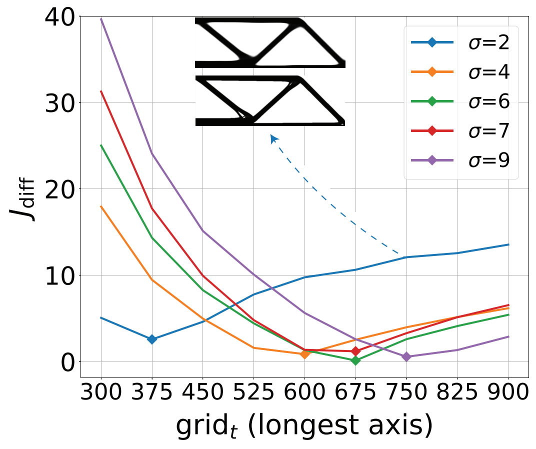

Figure 5: We compute at a source resolution, here 300 × 100, a design, upsample it to different target resolutions (375, ..., 900), and use them as initializations for a standard voxel-based method. The smallest change in the compliance value indicates the best upsampling ratio for a source design.

Interpolate ρ_s to a chosen fine *target resolution* grid_t by querying the ENDR [1], obtaining ρ_t. (3) Using ρ_t sampled at the element centers as an initialization, run the traditional voxel-based TO on grid_t to obtain the piecewise constant density field ρ_{voxel}. (4) Compute the compliance reduction J_{diff} achieved by step (3). The following stopping criterion is used for step (3):

$$\text{error} = \frac{||\bar{\rho}^i - \bar{\rho}^{i-1}||^2}{n_e} \leq \text{tol}, \tag{5}$$

where n_e is the number of grid voxels, $\text{tol} = 10^{-7}$ is the convergence tolerance, and $\bar{\rho}^i$ is the vector of per-element density variables at the i^{th} optimization iteration. In the ideal case, we would find $J_{\text{diff}} \approx 0$, meaning the upsampled design generated on grid_s is already nearly optimal for grid_t. As we see in Figure 5, depending on the scale, different neural designs computed at coarse resolutions are the best candidates for upsampling to higher resolutions. We note that at 300x100 source resolution, higher σs (corresponding to sub-pixel smoothing radii) tend to create more detailed solutions that cannot be properly represented on the coarse grid using the traditional TO's voxel-based smoothing filter but that perform well when discretized at higher resolutions.

5.4 Comparison with voxel-based and CNN parameterization

Our method obtains results that quantitatively match or exceed the results of two state-of-the-art methods: one employing a traditional voxel-based design representation [Andreassen et al. 2011] and another using a CNN design parameterization [Hoyer et al. 2019]. This performance is at the cost of longer computation due to a larger network and a slower learning rate (see Section 6.1). We restricted the optimizer to use 5000 iterations, but noticed that the voxel-based method and CNN fully converge in around 300 iterations. We use a filter radius of 1 for the voxel-based method and 2 for CNN (the default value in the original paper [Hoyer et al. 2019]), leading to solutions with simpler features compared to ours.

5.5 3D results

Our method can produce 3D designs as demonstrated in Figures 7-9 without any major change to the pipeline except the architectural modifications mentioned in Section 5.1. Despite the cubic increase in degrees of freedom of the finite element simulation under grid refinement, our solver still rapidly converges to a solution displacement field in just a few inexpensive CG iterations.

5.6 Performance

For producing 2D results, we use the sparse Cholesky factorization routine for the elasticity problem at each iteration. The solver time averaged across all iterations is 0.161s and 0.164s for the 300x100 MBB beam and 250x125 bridge examples in the top and bottom rows of Figure 6, respectively. For 3D results, we use a high-performance multigrid-preconditioned conjugate gradient solver at each iteration. The solve time is strongly influenced by two parameters, ϵ (the relative Young's modulus assigned to the void, which impacts the stiffness matrix's conditioning number) and the convergence tolerance, which we have defined as the residual force norm relative to the applied force norm. We set both of these parameters to 10^{-4} which strikes a good balance between accuracy and runtime. We used three coarsening levels in our multigrid hierarchy, ran one *full multigrid* cycle as a preconditioner for each CG iteration, and ran two Gauss-Seidel smoothing iterations at each level before restricting the residual and after interpolating the correction. We also re-use the displacement field computed for the previous density field as an initial guess for the solve on the updated design. With these settings, in 3D, the average times per iteration are 5.74s and 2.84s for the 320x160x80 bridge and 256x128x128 cantilever examples, respectively. The remaining time of each iteration—spent evaluating the network and updating its weights via backpropagation—approaches the elasticity solve time when done on our Nvidia Quadro RTX 8000 GPU but does not exceed it. For higher resolution designs, GPU memory limitations can force the network operations to be done on the CPU. In that case, the network evaluations and updates can become a timing bottleneck. The CPU used in all benchmarks is an AMD Ryzen 5950X.

6 CONVERGENCE

The primary limitation of our method is the large number of training iterations required to reach an optimal design. This is particularly striking in comparison to prior work [Zehnder et al. 2021] where, despite their similar design representation, solutions converged in fewer than 200 iterations as opposed to the 5000 iterations taken by our method. However, we argue that the increased convergence rate comes at the cost of compromising the ENDR's benefits. We investigate the fundamental differences between the "density-space" Optimality Criteria (OC) method employed by [Zehnder et al. 2021] and the direct gradient-based optimization used here.

[1]Experimenting with different upsampling method, we found that neural upsampling, i.e., querying the ENDR at higher resolutions, bilinear, and bicubic interpolation converge to highly similar compliances at larger upsampling ratios.

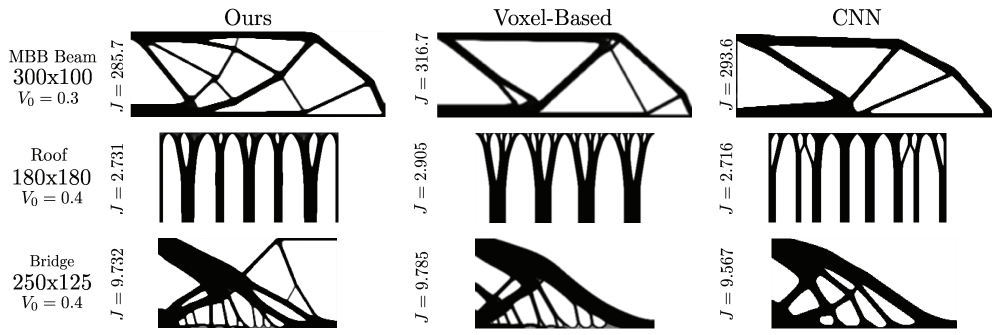

Figure 6: Comparison of the proposed method against the voxel- and CNN-based methods for a set of boundary conditions. Our method produces qualitatively different results whose compliances are either very close to or below those from the other methods. In the middle row, for visualization purposes only, we have mirrored the final result of all models.

6.1 Density-space OC

In order to accelerate convergence and eliminate artifacts, Zehnder et al. [2021] recommend decomposing the design update rule into two substeps (1) using the OC algorithm to update the design in "density space" (updating the vector of densities obtained by querying the network at their quadrature points, analogous to the element centroids in our framework); and (2) training the network to fit the generated density field to the updated design (essentially an image-fitting step). This strategy benefits from the rapid convergence of the OC algorithm for simple density parametrizations, leading to low iteration counts.

We note that if step (2) runs to convergence and is able to perfectly fit the OC update with zero error, the full algorithm will actually follow an identical sequence of steps as if one discarded the representation network and simply ran OC on the discrete vector of density samples. A perfect fit is of course impossible if the chosen σ prevents the network from representing the updated design—meaning the neural-based design will diverge from the discrete optimization—and yet our experiments with this two-step approach still showed severely diminished frequency-control and checkerboard-avoidance benefits when using ENDR in this mode.

Figure 10 reports some characteristic results from these experiments. We follow the approach from Zehnder et al. [2021], alternating between applying OC to update the voxel density field and fitting the network to the updated densities by minimizing a mean squared error (MSE). We omitted the additional smoothing filter recommended in [Zehnder et al. 2021] as our aim is to evaluate the native frequency-control capabilities of the network. We studied the effect of running more or fewer iterations of the MSE minimization between each OC step. For a moderate number of fitting iterations ("#MSE=50"), we observe that ENDR loses much of its ability to smoothly control the frequency content of the design; it produces designs with overly complex, noisy features and high compliance at lower scales and even suffers from checkerboard artifacts at higher scales. Reducing the number of fitting iterations to artificially prevent approximating the discrete OC iterates too closely helps somewhat (Figure 10, right), but still a jagged, unstructured design with poor compliance is obtained.

For these experiments, we used 128 neurons per layer and $d = 256$ Fourier features. We used 300 outer optimization steps and set Adam's learning rate to 3×10^{-4} for the MSE fitting.

6.2 Native OC

While typically applied to voxel-based design representations, the OC update rule does naturally generalize to other design parametrizations like ENDR—all that is needed is the gradient of compliance and volume with respect to the design variables. One would hope that applying this generalized OC algorithm natively in the ENDR could achieve the rapid convergence of OC seen in density space while retaining the representation's benefits. Unfortunately, the OC method did not converge reliably in our experiments, and we conjecture this is due to the performance of the heuristic OC algorithm degrading due to the high nonlinearity of our design representation.

7 CONCLUSION

We presented a novel approach to solving the long-standing problem of topology optimization. Although neural design parameterization in TO is not new, we revealed some new benefits of this approach. Specifically, the recently proposed neural parameterization with Fourier feature embedding allows for continuous tuning of a frequency prior on the design in a straightforward manner. Our method generates a range of designs independently of the finite element discretization grid, with attractive properties, such as mesh-independency and scale-aware upsampling. Our proposed method is shown to outperform or be on par with standard FEM-based solutions in terms of stiffness. Our current formulation and its accompanying optimization requires a larger number of iterations. We believe reducing the iteration count is feasible through improvements to the volume constraint enforcement strategy and tweaking of the activation functions.

REFERENCES

Niels Aage, Erik Andreassen, Boyan S Lazarov, and Ole Sigmund. 2017. Giga-voxel computational morphogenesis for structural design. *Nature* 550, 7674 (2017), 84–86.

Erik Andreassen, Anders Clausen, Mattias Schevenels, Boyan S Lazarov, and Ole Sigmund. 2011. Efficient topology optimization in MATLAB using 88 lines of code. *Structural and Multidisciplinary Optimization* 43, 1 (2011), 1–16.

Pierre Baque, Edoardo Remelli, Franccois Fleuret, and Pascal Fua. 2018. Geodesic convolutional shape optimization. In *International Conference on Machine Learning*. PMLR, 472–481.

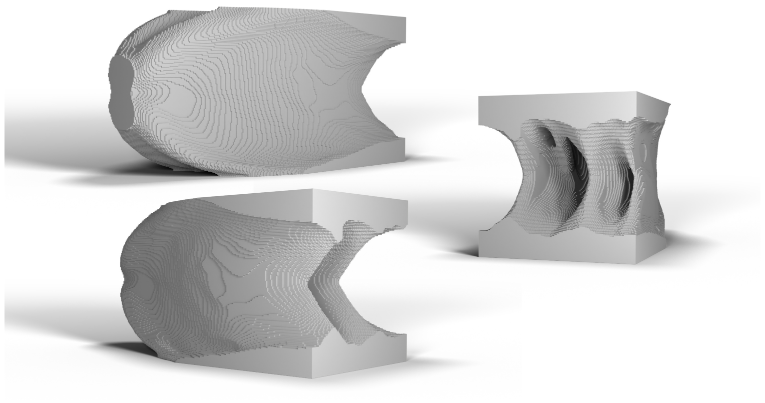

Figure 7: Cantilever Flexion E $\sigma = 4$ in 256x128x128 trained for 2700 iterations.

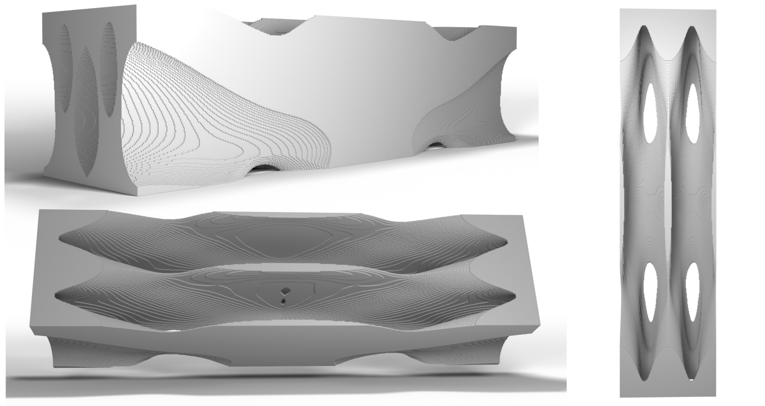

Figure 8: Bridge $\sigma = 1$ in 320x160x80 trained for 3500 iterations. The design is mirrored along X and Z axes for visualization.

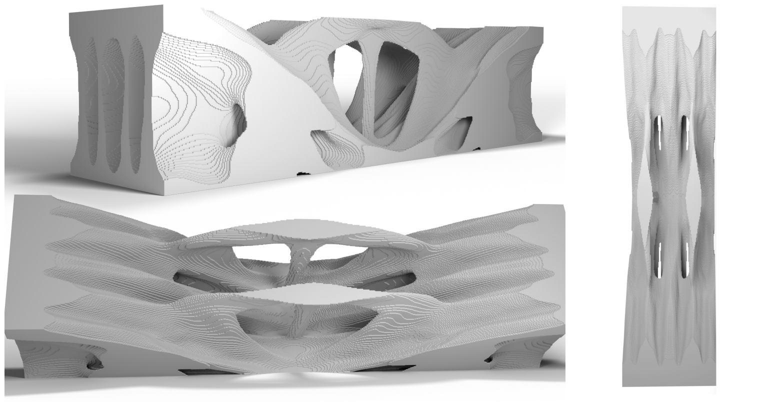

Figure 9: Bridge $\sigma = 2.5$ in 320x160x80 trained for 3500 iterations. The design is mirrored along X and Z axes for visualization.

Martin Philip Bendsoe and Ole Sigmund. 2013. *Topology optimization: theory, methods, and applications.* Springer Science & Business Media.

Thomas Borrvall. 2001. Topology optimization of elastic continua using restriction. *Archives of Computational Methods in Engineering* 8, 4 (2001), 351–385.

Tyler E. Bruns and Daniel A. Tortorelli. 2001. Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering* 190, 26 (2001), 3443–3459. https://doi.org/10.1016/S0045-7825(00)00278-4

Aaditya Chandrasekhar and Krishnan Suresh. 2021. TOuNN: Topology optimization using neural networks. *Structural and Multidisciplinary Optimization* 63, 3 (2021), 1135–1149.

Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 5939–5948.

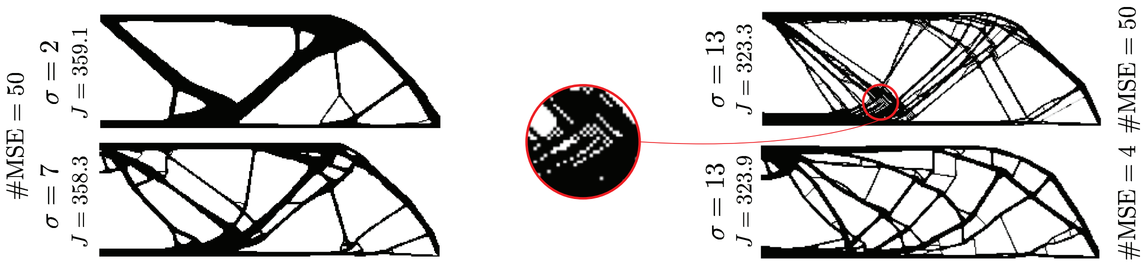

Figure 10: Results from our experiments using density-space OC to train ENDR. While parameter σ retains some control over the design's complexity (left), it is much less effective and significantly worse stiffness is achieved versus directly minimizing compliance with Adam; compare to Figure 2. These issues can be mitigated somewhat by limiting the number of fitting iterations (bottom right), but still most of ENDR's benefits are lost.

Benjamin Dupuis and Arthur Jacot. 2021. DNN-Based Topology Optimisation: Spatial Invariance and Neural Tangent Kernel. *arXiv preprint arXiv:2106.05710* (2021).

I Elishakoff, C Gentilini, and E Viola. 2005. Three-dimensional analysis of an all-round clamped plate made of functionally graded materials. *Acta Mechanica* 180, 1-4 (2005), 21–36.

Akash Garg, Andrew O Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. 2014. Wire mesh design. *ACM Transactions on Graphics* 33, 4 (2014).

Stephan Hoyer, Jascha Sohl-Dickstein, and Sam Greydanus. 2019. Neural reparameterization improves structural optimization. *arXiv preprint arXiv:1909.04240* (2019).

Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572* (2018).

Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. 2020. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6001–6010.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

XY Kou and ST Tan. 2007. Heterogeneous object modeling: A review. *Computer-Aided Design* 39, 4 (2007), 284–301.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895* (2020).

Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-band topology optimization on a sparsely populated grid. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934* (2020).

Przemyslaw Musialski, Thomas Auzinger, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2015. Reduced-order shape optimization using offset surfaces. *ACM Trans. Graph.* 34, 4 (2015), 102–1.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 165–174.

Joakim Petersson. 1999. Some convergence results in perimeter-controlled topology optimization. *Computer Methods in Applied Mechanics and Engineering* 171, 1-2 (1999), 123–140.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018. The Shape Space of Discrete Orthogonal Geodesic Nets. *ACM Trans. Graph.* 37, 6, Article 228 (Dec. 2018), 17 pages. https://doi.org/10.1145/3272127.3275088

Ali Rahimi, Benjamin Recht, et al. 2007. Random Features for Large-Scale Kernel Machines.. In *NIPS*, Vol. 3. Citeseer, 5.

Martin Regehly, Yves Garmshausen, Marcus Reuter, Niklas F König, Eric Israel, Damien P Kelly, Chun-Yu Chou, Klaas Koch, Baraa Asfari, and Stefan Hecht. 2020. Xolography for linear volumetric 3D printing. *Nature* 588, 7839 (2020), 620–624.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* (2013).

Ki-Hoon Shin and Debasish Dutta. 2001. Constructive representation of heterogeneous objects. *J. Comput. Inf. Sci. Eng.* 1, 3 (2001), 205–217.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems* 33 (2020).

Stratasys. 2020. Stratasys J8 family, the ultimate multi-material 3D printer. https://www.stratasys.com/3d-printers/j8-series. [Online; Accessed 15-11-2020].

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems* 33 (2020).

Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. 2017. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*. PMLR, 3424–3433.

Nobuyuki Umetani and Bernd Bickel. 2018. Learning Three-Dimensional Flow for Interactive Aerodynamic Design. 37, 4, Article 89 (July 2018), 10 pages. https://doi.org/10.1145/3197517.3201325

G. Gary Wang and S. Shan. 2006. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design* 129, 4 (05 2006), 370–380. https://doi.org/10.1115/1.2429697 arXiv:https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/129/4/370/5922708/370_1.pdf

Jun Wu, Christian Dick, and Rüdiger Westermann. 2016. A System for High-Resolution Topology Optimization. *IEEE Transactions on Visualization and Computer Graphics* 22, 3 (2016), 1195–1208. https://doi.org/10.1109/TVCG.2015.2502588

Jonas Zehnder, Yue Li, Stelian Coros, and Bernhard Thomaszewski. 2021. NTopo: Mesh-free Topology Optimization using Implicit Neural Representations. *arXiv preprint arXiv:2102.10782* (2021).

Xiaoting Zhang, Xinyi Le, Zihao Wu, Emily Whiting, and Charlie CL Wang. 2016. Data-driven bending elasticity design by shell thickness. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 157–166.