NTopo: Mesh-free Topology Optimization using Implicit Neural Representations

Jonas Zehnder [1] **Yue Li** [2] **Stelian Coros** [2] **Bernhard Thomaszewski** [1 2]

Abstract

Recent advances in implicit neural representations show great promise when it comes to generating numerical solutions to partial differential equations (PDEs). Compared to conventional alternatives, such representations employ parameterized neural networks to define, in a mesh-free manner, signals that are highly-detailed, continuous, and fully differentiable. Most prior works aim to exploit these benefits in order to solve PDE-governed *forward* problems, or associated *inverse* problems that are defined by a small number of parameters. In this work, we present a novel machine learning approach to tackle topology optimization (TO) problems. Topology optimization refers to an important class of inverse problems that typically feature very high-dimensional parameter spaces and objective landscapes which are highly non-linear. To effectively leverage neural representations in the context of TO problems, we use multilayer perceptrons (MLPs) to parameterize both density and displacement fields. Using sensitivity analysis with a moving mean squared error, we show that our formulation can be used to efficiently minimize traditional structural compliance objectives. As we show through our experiments, a major benefit of our approach is that it enables self-supervised learning of continuous solution spaces to topology optimization problems.

1. Introduction

Deep neural networks are starting to show their potential for solving partial differential equations (PDEs) in a variety of problem domains, including turbulent flow, heat transfer, elastodynamics, and many more (Hennigh et al., 2020; Raissi et al., 2019; Rao et al., 2020; Sitzmann et al., 2020; Lu et al., 2019). Thanks to their smooth and analytically-differentiable nature, implicit neural representations with periodic activation functions are emerging as a particularly attractive and powerful option in this context (Sitzmann et al., 2020). In this work, we explore the potential of implicit neural representations for structural topology optimization—a challenging inverse elasticity problem with widespread application in many fields of engineering (Bendsoe & Sigmund, 2013).

Topology optimization (TO) methods seek to find designs for physical structures that are as stiff as possible (i.e. least compliant) while adhering to a given material budget and being subjected to known boundary conditions and loading forces. While TO with mesh-based finite element analysis is a well-studied problem (Bendsøe & Sigmund, 1995), we argue that mesh-free methods provide unique opportunities for machine learning.

We propose the first self-supervised, fully mesh-free method based on implicit neural representations for topology optimization. The core of our approach is formed by two neural networks: a displacement network representing force-equilibrium configurations that solve the forward problem, and a density network that learns optimal material distributions in the domain of interests. To leverage the power of these representations, we cast TO as a stochastic optimization problem using Monte Carlo sampling. Compared to conventional mesh-based TO, this setting introduces new challenges that we must address.

To account for the nonlinear nature of implicit neural representations, we introduce a convex density-space objective that guides the neural network towards desirable solutions. We furthermore introduce several concepts from FEM-based topology optimization methods into our learning-based Monte Carlo setting to stabilize the training process and to avoid poor local minima.

We evaluate our method on a set of standard TO problems in two and three dimensions. Our results indicate that neural topology optimization with implicit representations is able to match the performance of state-of-the-art mesh-based solvers. To further explore the potential advantages of this approach over conventional methods, we show how our

[1]Department of Computer Science and Operations Research, Université de Montréal, Canada [2]Department of Computer Science, ETH Zürich, Switzerland. Correspondence to: Jonas Zehnder <jonas.zehnder@umontreal.ca>.

formulation enables self-supervised learning of continuous solution spaces for this challenging class of problems.

2. Related work

Neural Networks for Solving PDEs Deep neural networks have been widely used in different fields to provide solutions for partial differential equations for both forward simulations and inverse designs (Hennigh et al., 2020; Sitzmann et al., 2020; Raissi et al., 2020). These PDEs can be solved in their strong forms (Dissanayake & Phan-Thien, 1994; Lagaris et al., 1998; Sirignano & Spiliopoulos, 2018) and variational forms (He et al., 2018; Weinan & Yu, 2018). We refer to DeepXDE (Lu et al., 2019) for a detailed review. Explorations of using deep learning in conjunction with numerical solvers in simulation have also been done to accelerate the optimization process (Xue et al., 2020) and for learning the governing physics (Holl et al., 2020; Battaglia et al., 2016; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020; Grzeszczuk et al., 1998; Holden et al., 2019). With their continuous and analytically-differentiable solution fields, neural implicit representations with periodic activation functions (Sitzmann et al., 2020) offer a promising alternative to mesh-based finite element analysis. We leverage this new representation to solve high-dimensional inverse elasticity problems in a fully mesh-free manner.

Differentiable Simulation for Machine Learning There is also a line of work in developing differentiable simulation engines to adopt physics-based supervision in machine learning frameworks (Hu et al., 2019b; Liang et al., 2019; Hu et al., 2019a; Geilinger et al., 2020; Um et al., 2020; Holl et al., 2020). Liang *et al.* (2019) proposed a differentiable cloth simulation method that enables optimization of material properties, external forces, and other control parameters. Hu *et al.* (2019b) developed a real-time differentiable simulator for soft robotics for applications in reinforcement learning. Geilinger *et al.* (2020) proposed an analytically differentiable simulation framework that handles frictional contacts for both rigid and deformable bodies. To reduce numerical errors in a traditional solver Um *et al.* (2020) leverage a differentiable fluid simulator inside the training loop. Inspired but different from their work, we compute the analytical derivatives for our forward problem by leveraging auto-differentiation of neural networks.

Neural Representations Using implicit neural representation for complex signals has been an on-going topic of research in computer vision (Saito et al., 2020; Park et al., 2019), computer graphics (Mildenhall et al., 2020; Tancik et al., 2020), and engineering (Sitzmann et al., 2020; Hennigh et al., 2020). PIFu (2020) learns in the implicit representation of human bodies for monocular shape estimation. Sitzmann *et al.* (2020) use MLPs with sinusoidal activation functions to represent signed distance function of complex audio signal, surface geometries and wave propagation. Takikawa *et al.* (2021) propose an efficient neural representation that enable real-time rendering of high quality surface reconstructions. Mildanhall *et al.* (2020) proposed a neural radiance field representation for novel view synthesis. Witnessing the promising evidence from these work, we leverage the same advantages of implicit neural representations to learn the high dimensional solutions of structural topology optimization problems.

Topology Optimization with Deep Learning Topology optimization methods aim to find an optimal material distribution of a design domain given a material budget, boundary conditions, and applied forces. Building on a large body of finite-element based methods (Bendsoe & Kikuchi, 1988; Bendsoe & Sigmund, 2013; Wang et al., 2003; Sigmund, 2007; Bendsøe & Sigmund, 1999; Bendsøe, 1989; Andreassen et al., 2011; Sigmund, 2001; Allaire et al., 2004; 2005; Guo et al., 2014; Zhang et al., 2016; 2018; Luo et al., 2008; van Dijk et al., 2013), recent efforts have explored the use of deep learning techniques in this context. One line of work leverages mesh-based simulation data and convolutional neural networks (CNNs) to accelerate the optimization process (Zhang et al., 2019; Yu et al., 2019; Banga et al., 2018; Ulu et al., 2016; Nie et al., 2020; Lei et al., 2019; Lin et al., 2018). Perhaps closest to our work is the method by Hoyer *et al.* (2019), who reparameterize design variables with CNNs, but use mesh-based finite element analysis for simulation. While Hoyer *et al.* (2019) map latent vectors to discrete grid densities, Chandrasekhar *et al.* (2020) use fully connected layers to learn a continuous mapping from spacial locations to density values. Both methods use an explicit mesh for forward simulation and sensitivity analysis, whereas our method fully relies on neural implicit representations.

3. Problem Statement and Overview

Given applied forces, boundary conditions, and a target material volume $\hat{V}$, the goal of topology optimization is to the material distribution that leads to the stiffest structure. This task can be formulated as a constrained optimization problem,

$$\begin{aligned} \min_{\rho,u} L_{\text{comp}} &= \int e(\rho, u, \omega)\, d\omega \\ \text{s. t. } c_{\text{sim}}(\rho, u) &= 0 \\ \rho(\omega) &\in \{0, 1\} \\ \int_{\Omega} \rho\, d\omega &= \hat{V}\ , \end{aligned} \tag{1}$$

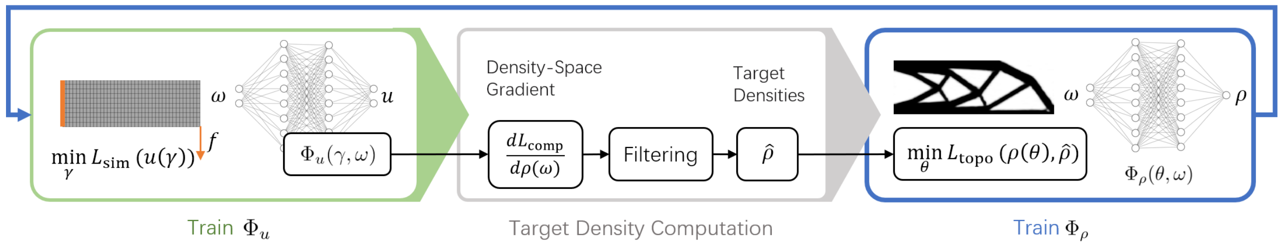
$\min_{\gamma} L_{\text{sim}}(u(\gamma))$
f
ω
u
$\Phi_u(\gamma,\omega)$
Density-Space Gradient
$\frac{dL_{\text{comp}}}{d\rho(\omega)}$
Filtering
Target Densities
$\hat{\rho}$
$\min_{\theta} L_{\text{topo}}(\rho(\theta),\hat{\rho})$
ω
ρ
$\Phi_\rho(\theta,\omega)$
Train Φ_u
Target Density Computation
Train Φ_ρ

Figure 1. Neural topology optimization pipeline. We compute optimal material distributions by alternately training two neural networks: the displacement network Φ_u and the density network Φ_ρ, mapping spatial coordinates ω to equilibrium displacements u and optimal densities ρ, respectively. In each iteration, we first update Φ_u by minimizing the total potential energy of the system. We then perform sensitivity analysis to compute density-space gradients which, after applying our sensitivity filtering, give rise to target density fields $\hat{\rho}$. Finally, we update Φ_ρ by minimizing the convex objective L_{topo} based on mean squared error between current and target densities.

where ρ is the material density field, ω runs over the domain Ω, c_{sim} are constraints requiring that the displacement u must be in static equilibrium, and e is the pointwise compliance, which is equivalent to the internal energy $1/2\,\varepsilon : \sigma$; see Section 4.2 for details.

Although manufacturing typically demands a binary material distribution, densities are often allowed to assume continuous values while encouraging convergence to binary solutions. We follow the same strategy and parameterize densities ρ and displacements u using implicit neural representations, $\Phi_\rho(\theta,\omega)$ and $\Phi_u(\gamma,\omega)$, respectively. By sampling a batch of locations ω^b and $\rho^b = \rho(\omega^b)$, we compute an estimate of L_{comp} using Monte Carlo integration

$$L_{\text{comp}} \approx \frac{|\Omega|}{n} \sum_{i}^{n} e(\omega_i^b) \tag{2}$$

where $|\Omega|$ is the volume of Ω. If u is a displacement in force equilibrium, we can compute the total gradient of the compliance loss with respect to the densities of the batch as

$$s = \frac{dL_{\text{comp}}}{d\rho} = -\frac{\partial L_{\text{comp}}}{\partial \rho}\,. \tag{3}$$

We will refer to this expression as the *density-space gradient*; see the supplemental document for a detailed derivation. The density-space gradient indicates how compliance changes w.r.t. changes in the density value, assuming that the force equilibrium constraints remain satisfied. On this basis, we compute the total gradient of the compliance loss with respect to the neural network parameters as

$$\frac{dL_{\text{comp}}}{d\theta} = \frac{\partial \rho}{\partial \theta}^T \frac{dL_{\text{comp}}}{d\rho}. \tag{4}$$

Using this gradient together with a penalty on the volume constraint would be one potential option for solving Equation (1). In practice, however, we observed that this approach does not lead to satisfying behavior. We elaborate on this problem below.

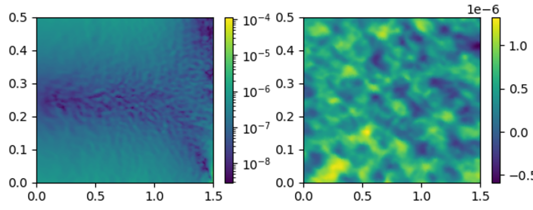

Figure 2. Failure of naive gradient descent. *Left*: negative density-space gradient on the domain for the beam example in the first iteration. Log-scale coloring is used to emphasize structure. *Right*: after one step of gradient descent (learning rate 10^{-5}), the neural network output has lost all structure from the density-space gradient.

Topology optimization is a non-convex optimization problem, whose solutions depend on the optimization method (Hoyer et al., 2019) and the path density values take. Much research has been done on developing optimization strategies that converge to *good* local minima for FEM-based solvers. Although it may not be the most effective choice in terms of reduction in compliance loss, the density-space gradient is generally assumed to be a good update direction. However, when using standard optimizers to update the parameters of the density network, the resulting change in densities is not at all aligned with the density-space gradient; see Figure 3. We tested both ADAM and SGD with little difference in the results. We note that the step taken by the optimizer reduces the objective, but the noise with respect to the density-space gradient persists over iterations, thus leading densities further and further away from desirable solutions. Eventually, this process converges to local minima that are meaningless from a structural point of view; see Section 5.4.

To analyze this unexpected behavior, we consider the change in densities induced by a step in the negative direction of the density-space gradient,

$$\Delta\rho = -\alpha \frac{dL_{\text{comp}}}{d\rho} \tag{5}$$

where α is the learning rate. The corresponding first-order change in network parameters is

$$\Delta\theta = -\alpha \frac{\partial\rho}{\partial\theta}^T \frac{dL_{\text{comp}}}{d\rho} \tag{6}$$

However, using this parameter change in the Taylor series expansion of the network output, we obtain

$$\begin{aligned}\Delta\rho' &= \rho(\theta+\delta\theta) - \rho(\theta) = \\ &= -\alpha \frac{\partial\rho}{\partial\theta}\frac{\partial\rho}{\partial\theta}^T \frac{dL_{\text{comp}}}{d\rho} + O(\alpha^2)\,.\end{aligned} \tag{7}$$

It is evident that, unless the Jacobian $\partial\rho/\partial\theta$ of the neural network is a unitary matrix, the direction of density change is different from Equation (5) even as $\alpha \to 0$.

To avoid converging to bad local minima, we seek to update the network parameters such that the network output changes along the density-space gradient. To this end, we define point-wise density targets $\hat{\rho}^b$ indicating how the network output should change. We then minimize the convex loss function

$$L_{\text{topo}}(\theta) = \frac{1}{n_b n}\sum_{b}^{n_b}\sum_{j}^{n}\left\|\rho(\theta,\omega_j^b) - \hat{\rho}(\omega_j^b)\right\|^2 \,. \tag{8}$$

While our density-space optimization strategy greatly improves results, we have still observed converge to minima with undesirable artefacts. Drawing inspiration from mesh-based topology optimization (Bourdin, 2001), we solve this problem with a sensitivity filtering approach (termed $\mathcal{FT}$ in Algorithm 1). To further accelerate convergence, we also adapt the optimality criterion method (Bendsøe & Sigmund, 1995) from mesh-based TO to our setting ($\mathcal{OC}$ in Algorithm 1).

Our resulting neural topology optimization algorithm, which we dubbed *NTopo*, is summarized in Algorithm 1 and further explained in the following Section.

Algorithm 1 NTopo: neural topology optimization.

```
 1: Initialize $\gamma^{(-1)}$, $\theta^{(0)}$ for $\Phi_\rho$, $\Phi_u$
 2: Initialize two optimizers: $\text{opt}_\theta, \text{opt}_\gamma$
 3: Run initial simulation: $\gamma^{(0)} \leftarrow \arg\min_\gamma L_{\text{sim}}(\gamma, \theta^{(0)})$
 4: for $n_{\text{opt}}$ iterations do
 5:     for $n_{\text{sim}}$ iterations do
 6:         $\gamma^{(l+1)} \leftarrow \text{opt}_\gamma.\text{step}(\gamma^{(l)}, \partial L_{\text{sim}}/\partial\gamma)$
 7:     end for
 8:     for $b = 1,..,n_b$ batches do
 9:         compute $\rho^b$, $s^b$
10:     end for
11:     $\hat{s} \leftarrow \mathcal{FT}(\rho^{1,\dots,n_b}, s^{1,\dots,n_b})$
12:     $\hat{\rho} \leftarrow \mathcal{OC}(\rho^{1,\dots,n_b}, \hat{s}^{1,\dots,n_b}, \hat{V})$
13:     for $b = 1,..,n_b$ batches do
14:         $\theta^{(l+1)} \leftarrow \text{opt}_\theta.\text{step}(\theta^{(l)}, \hat{\rho}^b, \partial L_{\text{topo}}/\partial\theta)$
15:     end for
16: end for
17: return $\Phi_\rho$
```

4. Neural Topology optimization

4.1. Neural Representation

Let $\Phi(x)$ be a neural network representation of a mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. We use SIREN (Sitzmann et al., 2020) for our MLP networks, which is a dense feed-forward neural network with sinusoidal activation functions. A SIREN-MLP with l layers, $l-1$ hidden layers, and h neurons in each layer hidden, is defined as

$$\phi_0 = \sin(\omega_0 W_0 x + b_0), \tag{9}$$

$$\phi_i = \sin(W_i\phi_{i-1} + b_i),\ 1 \le i \le l-1 \tag{10}$$

$$y = W_l\phi_{l-1} + b_l \in \mathbb{R}^m\ , \tag{11}$$

where x is the input vector, y is the output vector, W_i are weight matrices, b_i are biases, and ω_0 is a frequency dependent factor. We use a standard five-layer MLP with residual links and no regularization. The weights of all layers are initialized as suggested by Sitzmann *et al.* (2020).

4.2. Computing Static Equilibrium Solutions

We parametrize the displacement field u using a neural network $\Phi_u(\gamma, \omega)$. To find the displacement field u in static equilibrium, we minimize the variational form of linear elasticity which is given by

$$\begin{aligned}\min_u\ \ & L_{\text{sim}}(u(\gamma)) = \int_\Omega \frac{1}{2}\varepsilon(u) : \sigma(u)\, d\omega - \int_\Omega u^T f\, d\omega \\ \text{s. t.}\ \ & u|_{\partial\Omega_d} = u^*\end{aligned} \tag{12}$$

where $\partial\Omega_d$ is the part of the boundary of the domain with prescribed displacement u^*. In two dimensions we compute the stress tensor under plane stress assumption

$$\sigma = \frac{E}{1-\nu^2}\left((1-\nu)\varepsilon + \nu\,\text{trace}(\varepsilon)\mathbf{I}\right) \tag{13}$$

where ν is Poisson's ratio, E is Young's modulus, $\mathbf{I} \in \mathbb{R}^{2\times 2}$ is the identity matrix and $\varepsilon = \frac{1}{2}\left(\nabla u + \nabla u^T\right)$ is the linear Cauchy strain. In 3D we use Hooke's law

$$\sigma = \lambda\,\text{trace}(\varepsilon)\mathbf{I} + 2\mu\epsilon \tag{14}$$

where $\lambda = E\nu/((1+\nu)(1-2\nu))$ and $\mu = E/(2(1+\nu))$.

Following SIMP (Sigmund, 2001), we parameterize the Young's modulus E using the density field as

$$E(\rho) = \rho^p E_0 \; . \tag{15}$$

Larger values for the exponent p together with the volume constraint encourage binary solutions.

Sampling To evaluate the integral in Equation (12) we resort to a Quasi-Monte Carlo sampling approach. We generate stratified samples on a grid with $n_x \times n_y$ cells in 2D (and $n_x \times n_y \times n_z$ in 3D). We adjust n_x, n_y, n_z to match the aspect ratio of the domain, such that cell width, height and depth are similar.

Enforcing Dirichlet Boundary Conditions By constructing a function d that is zero on the fixed boundary and an interpolator of the function u^*, we enforce that the displacement field

$$u(\omega) = d(\omega)\Phi_u(\omega) + (I \circ u^*)(\omega) \tag{16}$$

always satisfy the essential boundary conditions, thus turning the constrained problem into an unconstrained one; see also (McFall & Mahan, 2009). We use simple boundaries in our examples for which analytic functions d are readily available; see the supplemental document for more details.

4.3. Density Field Optimization

We reparameterize the density field using a neural network Φ_ρ which maps spatial locations to their corresponding density values. The bound constraints on the densities are satisfied by applying a scaled logistic sigmoid function to the network output, specifically

$$\rho(\omega) = \text{sigmoid}(5\,\Phi_\rho(\omega)) \tag{17}$$

The total volume constraint is satisfied by the optimality criterion method described below.

Moving Mean Squared Error. Equation (8) can be interpreted as a moving mean squared error

$$\frac{1}{|\Omega|}\int_\Omega ||\rho(\theta,\omega) - \hat{\rho}(\omega)||^2 d\omega \; . \tag{18}$$

We minimize this loss using a mini-batch gradient descent strategy, where we use every batch only once. We collect multiple batches of data from $\hat{\rho}$ before we update ρ and $\hat{\rho}$, specifically $\hat{\rho}$ changes once every outer iteration. For this reason, we refer to this updating scheme as moving mean squared error (MMSE) in the following.

Sensitivity filtering $\mathcal{FT}$. In conventional topology optimization codes, filtering is an essential component for discovering desirable minima that avoid artefacts such as checkerboarding (Sigmund, 2007). While our neural representation does not suffer from the same discretization limitations that give rise to checkerboard patterns, we have nevertheless observed convergence to undesirable minima. Indeed, the neural representation alone does not remove the inherent reason for such artefacts: topology optimization is an underconstrained optimization problem with a high-dimensional null-space. Isolating good solutions from this null-space requires additional priors, filters, or other regularizing information.

In order to address this problem, we propose a *continuous* sensitivity filtering approach that, instead of using discrete approximations (Bourdin, 2001), is based on continuous convolutions. Following this strategy, we obtain filtered sensitivities as

$$\hat{s}(\omega) = \frac{\int H(\Delta\omega)\rho(\omega+\Delta\omega)s(\omega+\Delta\omega)\,d\Delta\omega}{\max(\gamma,\rho(\omega))\int H(\Delta\omega)\,d\Delta\omega} \tag{19}$$

where γ is set to 10^{-3} and H is the kernel

$$H(\Delta\omega) = \max\left(0, r - \|\Delta\omega\|\right) \tag{20}$$

with radius r. We know that the samples ω^i are distributed inside a grid, we use this fact to compute our approximation to the continuous filter

$$\hat{s}^i_j = \frac{\sum_{k\in N^j} H(\omega^i_j - \omega^i_k)\rho^i_k s^i_k}{\max(\gamma,\rho^i_j)\sum_{k\in N^j} H(\omega^i_j - \omega^i_k)} \tag{21}$$

where the neighborhood N is appropriately defined by cell sizes and radius r, such that points inside the footprint of the kernel H are in the neighborhood N. This is not an unbiased estimator of Equation (19) which could change the solution in undesired ways, but is has worked fine in our experiments.

Multi Batch-based Optimality Criterion Method $\mathcal{OC}$ The optimality criterion method we use is stochastic but otherwise very similar to the one in the discrete case (Andreassen et al., 2011). First a set of multiplicative updates B^i are computed, which then are applied to compute the target densities

$$\hat{\rho}^i = \text{clamp}(\rho^i(B^i)^\eta, \max(0,\rho^i - m), \min(1,\rho^i + m)) \tag{22}$$

where m limits the maximum movement of a specific target density and η is a damping coefficient. We used $m = 0.2$ and $\eta = 0.5$. The update B^i are computed using

$$B^i = \frac{-\hat{s}^i}{\lambda\frac{\partial V}{\partial \rho^i}} \tag{23}$$

where λ is found using a binary search such that the estimated volume of the updated densities using Monte Carlo integration matches the desired volume across all batches

$$\frac{|\Omega|}{n_b n}\sum_{b}^{n_b}\sum_{j}^{n}\hat{\rho}_j^b = \hat{V} \tag{24}$$

4.4. Continuous Solution Space

Apart from solving individual topology optimization problems for fixed boundary conditions and material budgets, our method can be readily extended to learn entire spaces of optimal solutions for, e.g., a continuous range of material volume constraints $\{\hat{V}^i\}$. To this end, we seek to find a density function $\rho(\omega, \hat{V})$ which yields the optimal density at any point ω in the domain for any material volume $\hat{V} \in [\underline{V}, \overline{V}]$ in the target range. In a supervised setting, a common approach is to first compute k solutions corresponding to different material volume constraints $\{\hat{V}^k\}$ and then fit the neural network using a mean squared error. By contrast, our formulation invites a fully self-supervised approach based on a modified moving mean squared error,

$$\int_{\underline{V}}^{\overline{V}}\int_{\Omega}\left\|\rho(\omega,\hat{V}) - \hat{\rho}(\omega,\hat{V})\right\|^2 d\omega d\hat{V}\ . \tag{25}$$

We minimize this loss by sampling one $\hat{V}^k$ at random and then update the density network using the same method as described for the single target volume case.

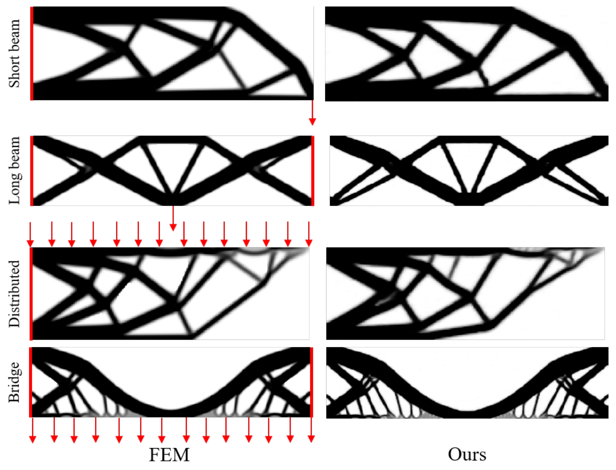

Figure 3. 2D comparison on four example problems. We provide our solutions as well as the corresponding FEM-based solutions to demonstrate the effectiveness of our approach qualitatively. Force and Dirichlet boundary conditions are visualized in the left images.

5. Results

To analyze our method and evaluate its results, we start by comparing material distributions obtained with our approach on a set of 2D example problems. To demonstrate the effectiveness of our method, we provide comparisons to a conventional FEM-based solver (Andreassen et al., 2011); Section 5.1. We then investigate the ability of our formulation to learn a continuous space of solutions in a fully self-supervised way. Comparisons to a data-driven, supervised approach indicate better efficiency for our method, suggesting that our approach opens new opportunities for design exploration in engineering applications (Section 5.2). We then turn to topology optimization problems with non-trivial boundary conditions and demonstrate generalization to 3D examples (Section 5.3). An ablation study of our method justifies the choice of using sensitivity filtering and casting the nonlinear topology objective into an MMSE form (Section 5.4). We provide detailed descriptions of our learning settings in the end (Section 5.5).

Table 1. Statistics of 2D comparisons. Comparing to a standard FEM-based solver, our results achieve quantitatively better compliance value for all examples.

EXP.	MTHD.	COMP.	VOL.	DOFS
BEAM	SIMP	1.173×10^{-3}	50.00%	30,000
	OURS	1.166×10^{-3}	49.96%	28,300
LONGBEAM	SIMP	2.734×10^{-4}	40.00%	31,250
	OURS	2.723×10^{-4}	39.97%	28,300
DISTRIBUTED	SIMP	1.981×10^{1}	40.00%	30,000
	OURS	1.975×10^{1}	40.00%	28,300
BRIDGE	SIMP	3.947×10^{0}	40.00%	31,250
	OURS	3.877×10^{0}	40.00%	28,300

5.1. Comparisons with FEM Solutions

We demonstrate that our results are competitive to a standard FEM-based solver (Andreassen et al., 2011). As can be seen in Figure 3, results are qualitatively similar, but our method often finds more complex supporting structures that lead to lower compliance values (see Table 1). For fair comparisons, the compliance values of these structures are evaluated using the FEM-based solver and we consistently use fewer degrees of freedom (DoFs). The DoFs in these two methods are the number of network weights and the number of finite element cells, respectively. We refer to the supplemental material for detailed descriptions of these examples.

5.2. Learning a Continuous Solution Space

Optimal designs for different volume constraints Here we demonstrate the capability of our method to learn continuous spaces of optimal density distributions for a continuous range of material volume constraints. We minimize the objective defined in Equation (25), volume constraint samples are drawn randomly in the range of $[30\%, 70\%]$.

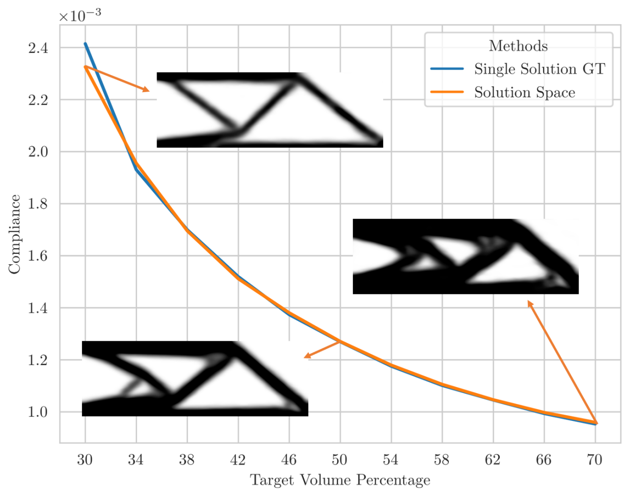

Figure 4. Results and compliance comparisons between evaluating our solution space network at discrete volume locations and the reference solution produced by our method in the single volume constraint case. As can be seen, the learned solution space does not compromise the quality individual solutions.

Our method is able to learn the solution space within $1,000$ iterations. To evaluate the accuracy of our learned solutions, we apply our single-volume algorithm for 11 discrete material volume constraints sampled uniformly across the target range. As can be seen in Figure 4, our solution space network does not compromise the quality of individual designs. The mean and maximum error in compliance are 0.75% and 3.83% respectively, and the mean and maximum volume violations are 0.5% and 2.83%, respectively. We therefore conclude that our model successfully learned the continuous solution space. Furthermore, we argue that the level of accuracy is acceptable for design exploration in many engineering applications.

Comparison with supervised setting We further compare our self-supervised training techniques with a supervised setting under the same computational budget. The cost of performing $1,000$ optimization steps with our larger solution space network is similar but less than finding 11 solutions under single volume constraints. We select 11 volume constraint values uniformly and train the network from these solutions in a supervised fashion. As can be seen in Figure 5, the network performs poorly for unseen data leading to infeasible designs and significant volume violations. On the contrary, our self-supervised approach leads to more plausible material distributions.

5.3. Irregular BCs and 3D Results

Our formulation extends to more complex boundaries, which we illustrate on a set of additional examples. Figure 6 shows multiple examples where densities are constrained

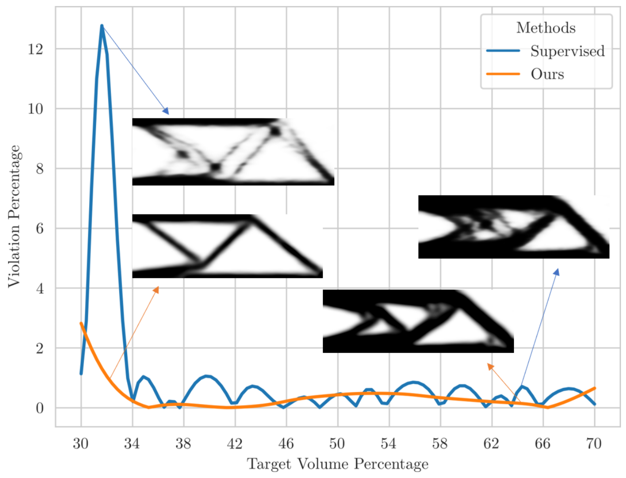

Figure 5. Comparisons with supervised setting. In this figure we demonstrate our self-supervised learning methods outperforms its supervised counterparts with less computational cost. The target volume constraint is shown on the x-axis and the constraint violation (in percentage) of the resulting structures is given on the y-axis.

on circular sub-domains and the Dirichlet boundary $\partial\Omega_d$ is also circular in one of the two examples.

Our method also generalizes to 3D, as demonstrated on the two examples shown in Figure 7. Due to the symmetry of the configuration, we apply symmetric boundary conditions to reduce the domain of interests to half and quarter for the cantilever beam and bridge example respectively to save computational cost and memory usage. Our method finds smooth solutions with various supporting features for the two tasks.

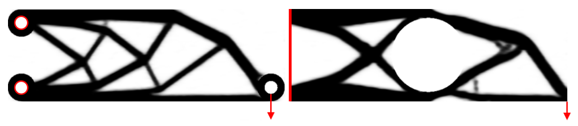

Figure 6. Curved boundaries: In the left example 3 holes have been put in the design using constraints on the density field. On the right, the density field is constrained to have a hole in the middle.

5.4. Ablation Studies

Here we provide evidence for the necessity of using a sensitivity filter, and our moving mean squared error loss during the optimization process. In the first test, we do not resort to the optimality criterion to compute the target density field nor do we use the mean squared error. Rather, we add a soft penalty term to satisfy the volume constraint and update the neural network only once per iteration. In the second test, we adopt the proposed method without filtering. Comparing with our reference solutions at different iterations, the alter-

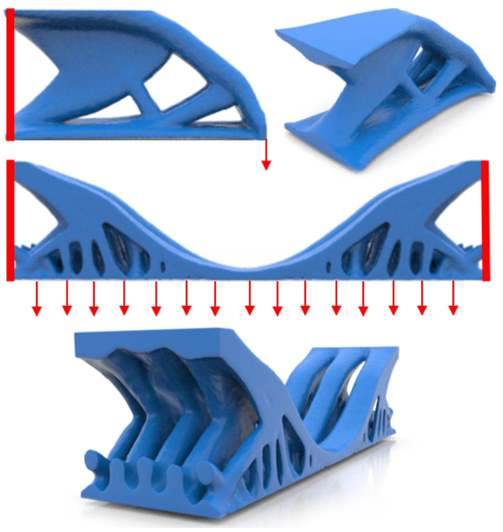

Figure 7. 3D Results. Here we show our solutions for two 3D examples to demonstrate the generality of our method. Top: A 3D cantilever beam example. Middle and bottom: A 3D bridge example. The mesh has been generated using a marching-cubes algorithm (Lewiner et al., 2003).

natives either converge significantly slower or arrive at poor local minima with many artefacts, such as disconnected struts or rough boundaries.

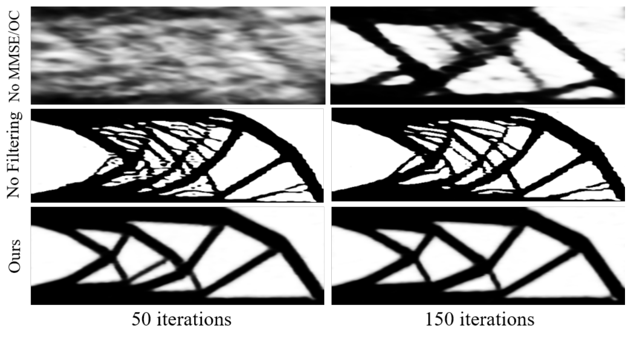

Figure 8. Ablation studies on the cantilever short beam test case. Top row: without moving mean squared error and optimality criterion, instead the density-space gradient is directly applied to the network and we only update of the density network once per optimization iteration. Middle row: proposed method without filtering. Bottom row: proposed method. We verify that both our moving mean squared error and filtering are quintessential.

5.5. Training Details.

We use Adam (Kingma & Ba, 2014) as our optimizer for both displacement and density networks and the learning rate of both is set to be $3 \cdot 10^{-4}$ for all experiments. We use $\omega_0 = 60$ for the first layer and 60 neurons in each hidden layer in 2D, and 180 hidden neurons in 3D. For the solution space learning setup, we use 256 neurons in each hidden layer in the density network to represent the larger solution space. For all experiments, we initialize the output of the density network close to a uniform density distribution of the target volume constraint by initializing the weights of the last layer close to zero and adjusting the bias accordingly. The simulation network is trained for $1,000$ iterations per each outer density optimization loop. We used $E_0 = 1, \nu = 0.3, p = 3, n_b = 50$ and $[n_x, n_y] = [150, 50]$ in 2D and $[n_x, n_y, n_z] = [80, 40, 20]$ in 3D.

6. Conclusion

We propose a novel, fully mesh-free topology optimization method using implicit neural representations. At the heart of our method lies two neural networks that represent force-equilibrium configurations and optimal material distribution, respectively. To overcome the challenges originating from the nonlinear nature of neural networks, we cast the topology optimization objective into a convex form and introduce several concepts known from conventional topology optimization to our setting. The effectiveness of our proposed method is demonstrated on different standard topology optimization examples with comparisons to their FEM-based solutions. We observe qualitatively similar results that are quantitatively on par or better than their reference solutions. We further demonstrate the capability of using our self-supervised learning set up to represent a solution space and prove better efficiency under less computational budget comparing to supervised learning. Our method can handle irregular boundary conditions and can generalize well to 3D. We believe our method can help with other non-convex inverse design problems. As such we consider it a stepping stone towards solving many varieties of inverse design problems using neural networks.

Limitations and Future Works We use the sigmoid function to enforce box constraints, however, the gradient becomes small when approaching 0 or 1. It could potentially hinder convergence but we do not find this to be a problem in practice. Better ways of enforcing box constraints in density space are an interesting avenue for future work. Similar to other Monte Carlo-based methods, advanced sampling strategies, for instance, importance sampling can be explored to speed up the training process, both of the simulation and optimization. We would like to further explore learning the solution spaces under different loading forces and boundary conditions to enable more applications in the future.

Acknowledgements

This research was supported by the Discovery Accelerator Awards program of the Natural Sciences and Engineering Research Council of Canada (NSERC) and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant No. 866480).

References

Allaire, G., Jouve, F., and Toader, A.-M. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194(1):363–393, 2004.

Allaire, G., De Gournay, F., Jouve, F., and Toader, A.-M. Structural optimization using topological and shape sensitivity via a level set method. *Control and cybernetics*, 34(1):59, 2005.

Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B. S., and Sigmund, O. Efficient topology optimization in matlab using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16, 2011.

Banga, S., Gehani, H., Bhilare, S., Patel, S., and Kara, L. 3d topology optimization using convolutional neural networks. *arXiv preprint arXiv:1808.07440*, 2018.

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D., and Kavukcuoglu, K. Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*, 2016.

Bendsøe, M. P. Optimal shape design as a material distribution problem. *Structural optimization*, 1(4):193–202, 1989.

Bendsoe, M. P. and Kikuchi, N. Generating optimal topologies in structural design using a homogenization method. 1988.

Bendsøe, M. P. and Sigmund, O. *Optimization of structural topology, shape, and material*, volume 414. Springer, 1995.

Bendsøe, M. P. and Sigmund, O. Material interpolation schemes in topology optimization. *Archive of applied mechanics*, 69(9-10):635–654, 1999.

Bendsoe, M. P. and Sigmund, O. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

Bourdin, B. Filters in topology optimization. *International journal for numerical methods in engineering*, 50 (9):2143–2158, 2001.

Chandrasekhar, A. and Suresh, K. Tounn: Topology optimization using neural networks. *Structural and Multidisciplinary Optimization*, pp. 1–15, 2020.

Dissanayake, M. and Phan-Thien, N. Neural-network-based approximations for solving partial differential equations. *communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

Geilinger, M., Hahn, D., Zehnder, J., Bächer, M., Thomaszewski, B., and Coros, S. Add: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15, 2020.

Grzeszczuk, R., Terzopoulos, D., and Hinton, G. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 9–20, 1998.

Guo, X., Zhang, W., and Zhong, W. Doing topology optimization explicitly and geometrically—a new moving morphable components based framework. *Journal of Applied Mechanics*, 81(8), 2014.

He, J., Li, L., Xu, J., and Zheng, C. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*, 2018.

Hennigh, O., Narasimhan, S., Nabian, M. A., Subramaniam, A., Tangsali, K., Rietmann, M., Ferrandis, J. d. A., Byeon, W., Fang, Z., and Choudhry, S. Nvidia simnet^{TM}: an ai-accelerated multi-physics simulation framework. *arXiv preprint arXiv:2012.07938*, 2020.

Holden, D., Duong, B. C., Datta, S., and Nowrouzezahrai, D. Subspace neural physics: Fast data-driven interactive simulation. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 1–12, 2019.

Holl, P., Koltun, V., and Thuerey, N. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020.

Hoyer, S., Sohl-Dickstein, J., and Greydanus, S. Neural reparameterization improves structural optimization. *arXiv preprint arXiv:1909.04240*, 2019.

Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J., and Durand, F. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019a.

Hu, Y., Liu, J., Spielberg, A., Tenenbaum, J. B., Freeman, W. T., Wu, J., Rus, D., and Matusik, W. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pp. 6265–6271. IEEE, 2019b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5): 987–1000, 1998.

Lei, X., Liu, C., Du, Z., Zhang, W., and Guo, X. Machine learning-driven real-time topology optimization under moving morphable component-based framework. *Journal of Applied Mechanics*, 86(1), 2019.

Lewiner, T., Lopes, H., Vieira, A. W., and Tavares, G. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools*, 8(2): 1–15, 2003.

Liang, J., Lin, M., and Koltun, V. Differentiable cloth simulation for inverse problems. 2019.

Lin, Q., Hong, J., Liu, Z., Li, B., and Wang, J. Investigation into the topology optimization for conductive heat transfer based on deep learning approach. *International Communications in Heat and Mass Transfer*, 97:103–109, 2018.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. Deepxde: A deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502*, 2019.

Luo, Z., Wang, M. Y., Wang, S., and Wei, P. A level set-based parameterization method for structural shape and topology optimization. *International Journal for Numerical Methods in Engineering*, 76(1):1–26, 2008.

McFall, K. S. and Mahan, J. R. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. *IEEE Transactions on Neural Networks*, 20(8):1221–1233, 2009.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pp. 405–421. Springer, 2020.

Nie, Z., Lin, T., Jiang, H., and Kara, L. B. Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain. *arXiv preprint arXiv:2003.04685*, 2020.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 165–174, 2019.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Raissi, M., Yazdani, A., and Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Rao, C., Sun, H., and Liu, Y. Physics informed deep learning for computational elastodynamics without labeled data. *arXiv preprint arXiv:2006.08472*, 2020.

Saito, S., Simon, T., Saragih, J., and Joo, H. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 84–93, 2020.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Sigmund, O. A 99 line topology optimization code written in matlab. *Structural and multidisciplinary optimization*, 21(2):120–127, 2001.

Sigmund, O. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33(4-5):401–424, 2007.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., and Fidler, S. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *arXiv preprint arXiv:2101.10994*, 2021.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

Ulu, E., Zhang, R., and Kara, L. B. A data-driven investigation and estimation of optimal topologies under variable loading configurations. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 4(2):61–72, 2016.

Um, K., Holl, P., Brand, R., Thuerey, N., et al. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *arXiv preprint arXiv:2007.00016*, 2020.

van Dijk, N. P., Maute, K., Langelaar, M., and Van Keulen, F. Level-set methods for structural topology optimization: a review. *Structural and Multidisciplinary Optimization*, 48(3):437–472, 2013.

Wang, M. Y., Wang, X., and Guo, D. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192(1-2):227–246, 2003.

Weinan, E. and Yu, B. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Xue, T., Beatson, A., Adriaenssens, S., and Adams, R. Amortized finite element analysis for fast pde-constrained optimization. In *International Conference on Machine Learning*, pp. 10638–10647. PMLR, 2020.

Yu, Y., Hur, T., Jung, J., and Jang, I. G. Deep learning for determining a near-optimal topological design without any iteration. *Structural and Multidisciplinary Optimization*, 59(3):787–799, 2019.

Zhang, W., Zhang, J., and Guo, X. Lagrangian description based topology optimization—a revival of shape optimization. *Journal of Applied Mechanics*, 83(4), 2016.

Zhang, W., Song, J., Zhou, J., Du, Z., Zhu, Y., Sun, Z., and Guo, X. Topology optimization with multiple materials via moving morphable component (mmc) method. *International Journal for Numerical Methods in Engineering*, 113(11):1653–1675, 2018.

Zhang, Y., Chen, A., Peng, B., Zhou, X., and Wang, D. A deep convolutional neural network for topology optimization with strong generalization ability. *arXiv preprint arXiv:1901.07761*, 2019.

NTopo: Mesh-free Topology Optimization using Implicit Neural Representations — Supplementary Material —

Jonas Zehnder[1] **Yue Li**[2] **Stelian Coros**[2] **Bernhard Thomaszewski**[1,2]

[1]Department of Computer Science and Operations Research, Université de Montréal, Canada
[2]Department of Computer Science, ETH Zürich, Switzerland

1 Sensitivity Analysis.

Here we provide the derivation of the total gradient $dL_{\text{comp}}/d\theta$ for the density network weights θ, taking into account that when we change θ the displacement u changes, such that it stays in static equilibrium. One way of computing the total gradient of the compliance objective is to solve for the discrete adjoint variables. Solving the discretized adjoint problem takes the form of

$$\frac{\partial^2 L_{\text{sim}}}{\partial \gamma^2}\lambda = -\frac{\partial L_{\text{comp}}}{\partial \gamma} \tag{1}$$

however this is a very large dense linear system which would be costly to solve, instead we rely on a point-wise argument to derive the adjoint gradient. We apply the chain rule with the neural network to get the total gradient

$$\frac{dL_{\text{comp}}}{d\theta} = \int_{\Omega} \frac{\partial \rho(\omega)}{\partial \theta} d_{\rho} e(\omega)\, d\omega \tag{2}$$

where $d_{\rho}e(\omega)$ is the point-wise total gradient of the point-wise compliance e. To derive this point-wise gradient we start by introducing the Lagrangian

$$\begin{aligned}\mathcal{L}(u,\rho,\lambda,\mu) &= \int_{\Omega} e + \lambda^T(\nabla\cdot\sigma(u) - f)\, d\omega + \int_{\partial\Omega} \mu^T(\sigma(u)n)\, d\omega && (3)\\ &= \int_{\Omega} e - \nabla\lambda : \sigma(u)\, d\omega - \lambda^T f\, d\omega + \int_{\partial\Omega} -\lambda^T(\sigma(u)n) + \mu^T(\sigma(u)n)\, d\omega && (4)\\ &= \int_{\Omega} e - \epsilon(\lambda) : C : \epsilon(u) - \lambda^T f\, d\omega + \int_{\partial\Omega} (\mu-\lambda)^T(\sigma(u)n)\, d\omega && (5)\end{aligned}$$

where we used integration by parts and leverged the fact that $\nabla\lambda : \sigma = \epsilon(\lambda) : \sigma$ due to the symmetry of the stress tensor σ.

In the case of topology optimization for linear elasticity the adjoint variables are readily available through $\lambda = \mu = u$ [1, 2]. Here we briefly show the derivation:

We denote the directional derivative of F w.r.t. x in the direction of h as

$$D_x F(x)h = \lim_{t\to 0} \frac{F(x+th) - F(x)}{t} \tag{6}$$

with which the adjoint variables are defined by [3]

$$\forall h \in (H^1)^2 : \quad D_u\mathcal{L}(u,\rho,\lambda,\mu)h = 0\,. \tag{7}$$

and by applying the definition of the derivative, this leads to

$$\forall h \in (H^1)^2 : \quad D_u\mathcal{L}(u,\rho,\lambda,\mu)h = \int \epsilon(h) : C : \epsilon(u) - \epsilon(h) : C : \epsilon(\lambda)\, d\omega = 0 \tag{8}$$

which is obviously true for $\lambda = u$.

We can then plugin the adjoint variables into the Lagrangian to get the point-wise gradient of the compliance with respect to the density as

$$\forall \delta\rho \in H^1 : D_\rho\mathcal{L}(u,\rho,\lambda,\mu)\delta\rho = \int_\Omega -\delta\rho\, \frac{1}{2}\epsilon(u) : \frac{\partial C}{\partial \rho} : \epsilon(u)\, d\omega \tag{9}$$

Since this holds for all functions $\delta\rho$ in H^1 we can conclude that the point-wise gradient is

$$d_\rho e(\omega) := -\frac{1}{2}\epsilon : \frac{\partial C}{\partial \rho} : \epsilon = -\frac{\partial e}{\partial \rho} \tag{10}$$

which notably has the opposite sign of the partial gradient $\partial e/\partial\rho$ where the implicit change in the displacement u is not taken into account.

Batch density-space gradient By sampling a batch of ω^b and $\rho^b = \rho(\omega^b)$ we can approximate the integrals and get a discrete version of the loss

$$L_{\text{comp}} = \int_\Omega e\, d\omega \approx L'_{\text{comp}} = \frac{|\Omega|}{n}\sum_i^n e(\omega_i^b) \tag{11}$$

using the derivations above we arrive at

$$\frac{dL_{\text{comp}}}{d\theta} = \int_\Omega \frac{\partial \rho(\omega)}{\partial \theta} d_\rho e(\omega)\, d\omega = -\int_\Omega \frac{\partial \rho(\omega)}{\partial \theta}\frac{\partial e(\omega)}{\partial \rho(\omega)}\, d\omega \tag{12}$$

$$\approx -\frac{|\Omega|}{n}\sum_i^n \frac{\partial \rho(\omega_i^b)}{\partial \theta}\frac{\partial e(\omega_i^b)}{\partial \rho(\omega_i^b)} = -\frac{\partial L'_{\text{comp}}}{\partial \rho^b}\frac{\partial \rho^b}{\partial \theta} \tag{13}$$

$$= \frac{dL'_{\text{comp}}}{d\rho^b}\frac{\partial \rho^b}{\partial \theta} \tag{14}$$

2 Additional Information for the Results

Here we provide more detailed descriptions of the results in the main paper. The design domains for both the *Short Beam* and the *Distributed* are 1.5×0.5, we apply $d(\omega) = \omega_x$ to the outputs of the displacement network Φ_u to enforce Dirichlet boundary on the left boundary. We apply a concentrated force at the bottom right corner for the *Short Beam* example and distributed forces on the top boundary for the *Distributed* example, all along the negative y-axis. The former example converged in 200 iterations and the latter converged in 400 iterations.

The design domains for both the *Long Beam* and the *Bridge* are 2.0×0.5, however, due to the symmetric configuration, we only perform simulation and optimization on half of the domain to avoid unnecessary computations. The domains are then 1.0×0.5, the analytical function $d_x(\omega)$ for x-direction and $d_y(\omega)$ for y direction are then $d_x(\omega) = \omega_x(\omega_x - 1.0)$ and $d_y(\omega) = \omega_x$. This enforces the left boundary to have no displacement and the normal displacements of the right boundary to vanish. A concentrated force is applied at the bottom right corner of the new design domain for the *Long Beam example* and distributed loading forces are added to the bottom boundary for the *Bridge* example, all long negative y-axis. The iteration count for these two are both 200. For visualization, we mirror the other half.

For the $3D$ *Beam* example, we run our method only on half of the design domain along z-axis, due to symmetry and the actual design domain is $1.5 \times 0.5 \times 0.25$. The analytical function we used for all directions is $d(\omega) = \omega_x$. Distributed forces are applied at $x = 1.5$, $y = 0$ and $z \in [0.0, 0.25]$ For the $3D$ *Bridge* example our method is only performed on a quarter of the design domain ($1.0 \times 0.5 \times 0.25$), where both x and z dimension are reduced to half. The functions to enforce Dirichlet boundary conditions are $d_x(\omega) = \omega_x(\omega_x - 1.0)$, $d_y(\omega) = \omega_x$ and $d_z(\omega) = \omega_x(\omega_z - 0.25)$. Forces are applied to the bottom plane where $y = 0$, $x \in [0.0, 1.0]$ and $z \in [0.0, 0.25]$. All forces applied are along the negative y-axis. The iteration count for these two examples is 100.

2.1 Curved Boundaries

We have used two additional types of constraints in these examples, constraining densities and constraining displacements on curved boundary. Since derivatives of the density field $\rho(\omega)$ do no appear in the objectives directly, constraining densities is rather straight forward. We just overwrite the density when the point ω falls into a constraint region Ω_ρ^c:

$$\tilde{\rho}(\omega) = \begin{cases} \rho(\omega) & \text{if } \omega \notin \Omega_\rho^c \\ \rho^c(\omega) & \text{if } \omega \in \Omega_\rho^c \end{cases} \tag{15}$$

and use the resulting field $\tilde{\rho}$ in place of ρ. For the displacement u, applying constraints is more difficult since its Jacobian appears in the objectives, requiring that the field u is sufficiently smooth inside the domain. We use a length factor d [4], to define the constrained displacement field

$$u(\omega) = d(\omega)\Phi_u(\omega) \tag{16}$$

where d has to be zero on the boundary and have non-zero first-order derivatives on the boundary (otherwise the first-order derivatives are also constrained to zero on the boundary, which we do not want). Here we show a simple analytic method for defining the length factor and is accurate on the whole primitives not only on sampled points: we allow for n primitives that require the ability to compute a C^1 continuous distance function, giving $d_1^2, \ldots d_n^2$ squared distances. We then combine them to construct a smooth length factor

$$d(\omega) = \sqrt{\operatorname{mix}(d_1^2, \ldots, d_n^2)} \tag{17}$$

where mix repeatedly merges two distances by applying a function $m(\cdot, \cdot)$ in a tree like fashion, where we use the following function

$$m(d_1^2, d_2^2) = \frac{d_1^2 d_2^2}{d_1^2 + d_2^2 + \varepsilon} \qquad \varepsilon = 10^{-4} \tag{18}$$

This function is basically the power smooth min [5] with $k = 2$ plus ε, notably this function is zero when either distance d_1 or distance d_2 is zero and seems to have nice first-order derivative properties.

Data of curved boundary examples The domain for both examples is 1.5×0.5.

Three-hole-design: The bottom left of the domain is $[0, 0]$. The holes have inner radius 0.035 and outer radius 0.075, they are located at $\{[0.075, 0.425], [0.075, 0.075], [1.425, 0.075]\}$. The force is applied at $[1.425, 0.04]$. The length factor of one circle was defined using

$$d(\omega) = \begin{cases} 0 & \text{if } \|c - \omega\| < r \\ \|c - \omega\| - r & \text{otherwise} \end{cases} \tag{19}$$

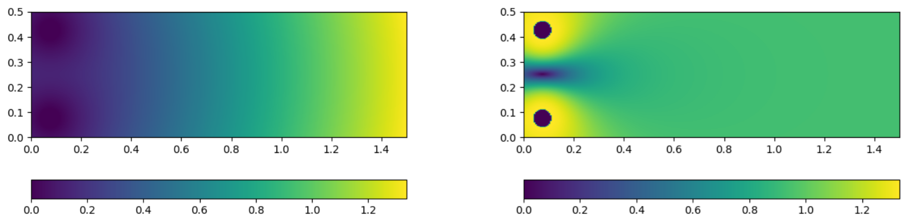

Figure 1: Left: length factor d of the curved boundary example, Right: norm of gradient of length factor $\|\partial d/\partial \omega\|$

where c is the center and r is the radius, and then the length factors of two circles were combined using the procedure explained above.

Hole in the middle-design: The radius of the circle is 0.2 and the center of the circle is at the center of the domain.

References

[1] Grégoire Allaire, François Jouve, and Anca-Maria Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194(1):363–393, 2004.

[2] Zhen Luo, Liping Chen, Jingzhou Yang, Y Zhang, and K Abdel-Malek. Compliant mechanism design using multi-objective topology optimization scheme of continuum structures. *Structural and Multidisciplinary Optimization*, 30(2):142–154, 2005.

[3] Fredi Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc., 2010.

[4] Kevin Stanley McFall and James Robert Mahan. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. *IEEE Transactions on Neural Networks*, 20(8):1221–1233, 2009.

[5] Inigo Quilez. smooth minimum. `https://www.iquilezles.org/www/articles/smin/smin.htm`. Accessed: 2021-15-01.