# Equivariant Deep Learning for 3D Topology Optimization

**David Erzmann**

A thesis submitted for the degree of
*Doktor der Ingenieurwissenschaften*

Center for Industrial Mathematics
University of Bremen
Germany

April 2024

Supervisors:

1. Prof. Dr. Dr. h.c. Peter Maaß
2. Prof. Dr. Andreas Rademacher
3. Prof. Dr. Heribert Blum

Dissertation Defense Date: June 20, 2024

# Abstract

The 21st century witnessed the ascent of deep learning, paving the way for data-driven topology optimization approaches via neural networks. This paradigm shift holds the promise of significant speed-ups compared to classical methods; however, deep learning still faces challenges related to generalization and the requirement for training data. Moreover, the field suffers from a dramatic lack of research infrastructure, which hampers both progress and comparability of results. Notably, prior to our research, neither a reliable yet flexible code base nor public three-dimensional datasets existed. We tackled these challenges by contributing a new public dataset and developing a Python library for three-dimensional topology optimization utilizing deep learning. Additionally, we demonstrate that incorporating physical information into the training process significantly reduces the reliance on extensive training data and enhances overall generalization capabilities.

This work consists of four publications. The initial publication introduces the SELTO dataset, comprising nearly $10,000$ three-dimensional samples, each providing topology optimization problems and corresponding solutions. The second publication presents DL4TO, a pioneering PyTorch-based deep learning library for topology optimization, accompanied by comprehensive documentation and online tutorials. The third publication showcases the efficacy of equivariant neural networks and a physics-inspired data preprocessing strategy, substantially reducing the need for extensive training datasets and enhancing generalization capabilities. Finally, the fourth paper employs neural operators to replace the PDE solver in the widely used SIMP method, which is its primary bottleneck. This publication underscores the efficacy of equivariance in learned methods and emphasizes the crucial role of a gradient-consistent loss function when applied in a gradient-based optimization scheme like SIMP.

# Zusammenfassung

Die Gewährleistung von Qualität und Stabilität mechanischer Strukturen unter Berücksichtigung von Kosten- und Gewichtsvorgaben stellt seit jeher eine große Herausforderung im industriellen Bauwesen dar. Mit dem Aufkommen von Computersystemen entstanden in den 1990er Jahren völlig neue Möglichkeiten zur automatisierten Strukturoptimierung. Insbesondere bieten Methoden der Topologieoptimierung Ingenieuren ein leistungsstarkes Werkzeug zum Entwurf potenziell optimaler Designs, basierend auf mathematischen Optimierungsalgorithmen. Trotz erheblicher Fortschritte auf dem Gebiet bestehen weiterhin Herausforderungen, insbesondere aufgrund langer Rechenzeiten für hochauflösende dreidimensionale Strukturen.

Große Fortschritte im Bereich des Deep Learnings in den 2010er Jahren ebneten den Weg für neue datengetriebene Ansätze zur Topologieoptimierung mittels neuronaler Netzwerke. Dieser Paradigmenwechsel verspricht eine erhebliche Verkürzung der Rechenzeiten im Vergleich zu klassischen Methoden. Trotz des enormen Potenzials stehen Deep Learning Methoden nach wie vor großen Herausforderungen gegenüber, insbesondere hinsichtlich der Generalisierbarkeit von Ergebnissen und dem hohen Bedarf an umfassenden Trainingsdaten. Zusätzlich weist das Gebiet der datengetriebenen Topologieoptimierung einen drastischen Mangel an Forschungsinfrastruktur auf, der sowohl den Fortschritt als auch die Vergleichbarkeit von Ergebnissen behindert. Trotz des großen Forschungsinteresses existieren weder ein zuverlässiges und flexibles Code-Framework noch dreidimensionale Trainingsdatensätze.

Bei diesen Herausforderungen setzt die vorliegende Dissertation an. Hintergund war die Erarbeitung und Veröffentlichung eines Datensatzes und einer darauf aufbauenden Python-Bibliothek. Die Daten und der Code wurden öffentlich bereitgestellt mit dem Ziel, den Zugang in den Bereich für künftige Forschende zu erleichtern und einen Beitrag zur besseren Vergleichbarkeit und Transparenz von Ergebnissen zu leisten. Außerdem wird gezeigt, dass die Integration physikalischer Informationen in den Trainingsprozess die Abhängigkeit von umfangreichen Trainingsdaten erheblich reduziert und die Generalisierbarkeit der Modelle verbessert.

Die Dissertation basiert auf vier Publikationen an der Schnittstelle von Deep Learning und Topologieoptimierung. Die erste Veröffentlichung ist der SELTO-Datensatz, der fast 10.000 dreidimensionale Problem-Ground-Truth Datenpaare umfasst. Die zweite

Veröffentlichung präsentiert DL4TO, eine vom Verfasser entwickelte Deep Learning-Bibliothek für Topologieoptimierung, basierend auf PyTorch. Neben dem Code umfasst die Bibliothek eine umfangreiche Dokumentation sowie Online-Tutorials. In der dritten Veröffentlichung wird gezeigt, dass durch äquivariante neuronale Netzwerke und eine physikinspirierte Datenvorverarbeitungsstrategie der Bedarf an großen Trainingsdatensätzen erheblich reduziert werden kann. Außerdem wird die Verbesserung der allgemeinen Qualität und Generalisierbarkeit der Modelle demonstriert. In der vierten Publikation wird der PDE-Löser, welcher das größte Bottleneck in der weit verbreiteten SIMP-Methode darstellt, durch einen neuronalen Operator ersetzt. Hier demonstriert der Verfasser erneut die Effektivität des äquivarianten Lernens und hebt die entscheidende Rolle einer gradientenkonsistenten Lossfunktion in gradientenbasierten Optimierungsschemata hervor.

# Acknowledgements

I want to express my sincere appreciation to Dr. Sören Dittmer for his outstanding support and guidance throughout my doctoral journey. Additionally, I am grateful to Prof. Dr. Dr. h.c. Peter Maaß for his trust and patience and for granting me the opportunity to pursue my PhD in his working group. I was fortunate to be a part of the AG Industriemathematik, and I want to convey my appreciation to other members of the working group, particularly Judith Nickel, Janek Gödeke, Hannes Albers, and Clemens Arndt, for engaging in many exciting and encouraging discussions. Special thanks go to the administrators, Dörte Mindermann and Judith Barthel, for their assistance with organizational matters.

Embarking on my PhD during the challenging COVID-19 pandemic, I am grateful to Dr. Jens Behrmann for encouraging me to pursue a doctoral degree and for his guidance during my Master's thesis.

I extend profound thanks to my friends and family for their unwavering support over the past few years. Throughout our Bachelor's, Master's, and doctoral studies, Marek Wiesner, Alexander Denker, Lea Höyns, Rabea Sondershaus, and I mutually uplifted and assisted each other. Additionally, the constant encouragement from my parents and my brother Julian is deeply valued. Above all, I want to give my heartfelt appreciation to Lennart Hinz for being a constant companion and supporter throughout the entire journey.

# Contents

# Chapter 1

# Introduction

Developing and constructing three-dimensional products, particularly in industrial settings, prompts the challenge of ensuring quality and stability within specified cost and weight constraints. Since the 1990s, a novel domain known as *topology optimization* (TO) [1] in computer-aided engineering has emerged. TO focuses on determining the optimal arrangement of materials under predefined constraints within a given design space. The prevailing methods for TO, such as density-based techniques like SIMP, involve discretizing the design domain into a voxel grid, where each voxel is assigned a material density value. SIMP and similar methods use an iterative update scheme to progressively adjust the density field based on predefined optimization objectives and constraints.

Despite significant progress in the field, challenges persist, especially in dealing with long computation times for high-resolution and three-dimensional components. These extended computation times primarily arise from the resource-intensive solving of the underlying partial differential equation (PDE) for linear elasticity, which is required at each iteration.

Motivated by the recent rise of deep learning (DL), researchers have explored integrating neural networks into TO [2, 3]. DL-based TO holds the potential for substantial speed-ups compared to classical approaches. Consequently, various use cases and strategies for employing neural networks in TO have emerged. Initial efforts aimed at reducing the number of SIMP iterations or even eliminating classical iterations altogether. Alternatively, DL methods can accelerate classical TO iterations by replacing the PDE solver with a neural network, addressing SIMP's primary bottleneck. While extensive research has been conducted on DL-based approaches for PDEs in recent years, applying these methods to TO remains relatively new and has yet to be explored. Finally, an alternative approach involves using neural networks to reparameterize TO's density field through an implicit representation. Since these models are mesh-independent, they can represent the density function at arbitrary resolutions.

This thesis provides a comprehensive review of the latest DL methodologies applied to TO. It includes a critical analysis of the drawbacks and limitations inherent in various approaches, supplemented by insights and recommendations derived from my research to advance and enhance the current state of the field. Furthermore, this thesis integrates four of my published works addressing critical aspects of learned TO. These publications have substantially improved the research infrastructure and understanding of DL's capabilities and limitations in the field.

## 1.1 Structure of the Thesis

This thesis is split into two parts. In the first part, we define the classical TO problem and discuss common approaches for classical and learned TO. In Chapter 2, we establish foundational knowledge essential for subsequent discussions. Sections 2.1 and 2.2 provide an introduction to PDEs and DL, respectively. In Chapter 3, we derive the linear elasticity problem formulation and discuss SIMP, ESO, and level-set methods, which are the three most common classical approaches for TO. Chapter 4 provides a comprehensive overview of recent advancements and hindrances in applying DL techniques to TO. Finally, the second part of the thesis encompasses our original publications on learned topology optimization methodologies.

## 1.2 Contributions and Papers

This cumulative thesis is based on the following publications:

**SELTO Dataset: A Benchmark Dataset for Deep Learning for 3D Topology Optimization**
Sören Dittmer, David Erzmann, Henrik Harms, Rielson Falck, Marco Gosch
*Zenodo*, (2023).
`DOI: 10.5281/zenodo.7781392`
Rielson Falck and Marco Gosch (Ariane Group) generated the dataset using the SIMP implementation in the commercial Synera software. I undertook the task of translating the dataset into Python, ensuring compatibility with the DL4TO software library, and conducted thorough testing and bug-fixing.

**DL4TO: A Deep Learning Library for Sample-Efficient Topology Optimization**
David Erzmann*, Sören Dittmer, Henrik Harms, Peter Maaß
*International Conference on Geometric Science of Information, Springer*, (2023).

DOI: 10.1007/978-3-031-38271-0_54
The concept and structure of the DL4TO library was formed in discussions with Sören Dittmer. I took on the primary role in implementing the library, as well as managing its maintenance, documentation, and online tutorials. Additionally, I conceived and wrote the paper and conducted the numerical experiments.

**SELTO: Sample-Efficient Learned Topology Optimization**
Sören Dittmer, David Erzmann*, Henrik Harms, Peter Maaß
*Currently under review at Heliyon, arXiv preprint arXiv:2209.05098*, (2023).
DOI: 10.48550/arXiv.2209.05098
The idea was formed in discussions with Sören Dittmer. I am responsible for the implementation and did the majority of the writing of the paper.

**Equivariant Neural Operators for Gradient-Consistent Topology Optimization**
David Erzmann*, Sören Dittmer
*Journal of Computational Design and Engineering*, (2024).
DOI: 10.1093/jcde/qwae039
The idea was formed in discussions with Sören Dittmer. I am responsible for the implementation and did the majority of the writing of the paper.

## 1.3 List of Abbreviations

| | |
|---|---|
| **3D** | Three-dimensional |
| **TO** | Topology optimization |
| **SIMP** | Solid isotropic material with penalization |
| **ESO** | Evolutionary structural optimization |
| **BESO** | Bi-directional evolutionary structural optimization |
| **PDE** | Partial differential equation |
| **FDM** | Finite difference method |
| **FEM** | Finite element method |

*Main author

**AI** Artificial intelligence

**ML** Machine learning

**DL** Deep learning

**NN** Neural network

**MLP** Multilayer perceptron

**CNN** Convolutional neural network

**PINN** Physics-informed neural network

**NO** Neural operator

**DeepONet** Deep operator network

**FNO** Fourier neural operator

**INR** Implicit neural representation

**RL** Reinforcement learning

# Part I

# Background

# Chapter 2

# Preliminaries and Foundations

In this chapter, we provide a concise overview of the foundational concepts essential for understanding the context of this thesis. We start with an overview of relevant PDEs in Section 2.1. Subsequently, we briefly introduce the basics of deep learning in Section 2.2.

## 2.1 Partial Differential Equations

*Partial differential equations* (PDEs) are an essential tool for modeling natural phenomena. PDEs naturally arise in a wide range of applications, ranging from scattering problems [4], electrodynamics [5], quantum physics [6] to finance [7] and ecology [8]. This section presents an introductory overview of PDEs and discusses numerical methods for their solution. For a more comprehensive exploration of the topic, see [9, 10, 11].

### 2.1.1 Categorization of PDEs

PDEs are mathematical equations that involve partial derivatives of an unknown function of several independent variables. They can be categorized into different types based on their properties and characteristics. In the context of this thesis, we focus on *second-order* PDEs, i.e., PDEs with a highest derivative of order two. Second-order PDEs can be classified into three categories: hyperbolic, parabolic, and elliptic. From a physical perspective, *hyperbolic* equations model the transport of physical quantities, such as fluids or waves. *Parabolic* PDEs describe evolutionary phenomena, including heat conduction and particle diffusion. *Elliptic* equations are associated with steady-state systems, generally corresponding to the minimum of energy [12].

A PDE can be *linear* or *nonlinear*. It is termed linear if it exhibits linearity in the unknown function and all its derivatives. Linear PDEs possess well-understood properties, making them more tractable than their nonlinear counterparts.

### 2.1.2 Numerical PDE Solvers

The majority of PDEs arising from real-world problems lack analytic solutions. Consequently, for many practical problems, numerical solvers become the sole viable option. Here, we provide a brief overview of the most prevalent numerical methods for solving PDEs, along with their key characteristics.

In addressing elliptic PDEs, the *finite difference method* (FDM) and the *finite element method* (FEM) emerge as the foremost techniques for discretization and solution. FDM involves discretizing the computational domain into a grid and approximating spatial derivatives of the PDE solution through linear combinations of values at grid points, utilizing finite difference operators [11]. This approach is based on a local Taylor series expansion and proves relatively straightforward for implementation on rectangular geometries. However, it typically necessitates a uniform grid approximation of the domain, potentially posing challenges for complex geometries. On the other hand, the FEM [13] focuses on approximating the PDE solution within a finite-dimensional vector space spanned by basis functions with local support. The basis functions are often chosen as piece-wise polynomials supported on a set of elements, which are adjacent cells in the mesh. While more flexible than FDM, FEM is usually more challenging to implement and requires the construction of a finite element mesh, which can be computationally expensive. Both finite difference and finite element methods yield large, sparse, and highly structured linear algebra systems.

Beyond FDM and FEM, alternative approaches exist. *Spectral methods* [11, 14, 15] form one such category, relying on approximating the solution $u$ within a finite-dimensional vector space spanned by basis functions with global support, typically involving Chebyshev polynomials or trigonometric functions. Spectral methods have an exponential convergence rate and usually require only a small number of basis functions to achieve a given accuracy, leading to smaller but dense linear algebra systems. The *finite volume method* (FVM) solves the PDE by discretizing the domain into control volumes and integrating the governing equations over these volumes. The FVM is especially effective for discretizing hyperbolic equations and convection-dominated problems. Alternatively, approaches like *particle methods* or *discrete element methods* [16] focus on individual elements and their interactions. These methods are particularly suited for simulating complex applications such as sand simulation, which can be challenging to model using other methods.

For time-dependent PDEs, the typical procedure involves time discretization using a time-stepping scheme, such as backward differentiation schemes (e.g., backward

Euler) and Runge-Kutta methods [11, 13]. Subsequently, a spatial discretization method, such as previously discussed, is employed to solve the resulting stationary PDE at each time step. Alternatively, joint space-time methods can be employed to discretize the space and time domain simultaneously, which is especially useful in the case of moving domains [17, 18].

### 2.1.3 Tensor Field Operators

We require operators tailored for tensor fields to formulate the strong and variational formulation of the PDE for linear elasticity in Chapter 3. Let $m, n, k, d \in \mathbb{N}$, and $\Omega \subset \mathbb{R}^d$ be a spatial domain for the subsequent definitions. We commence with the definition of the double dot product, a generalization of inner products to tensor fields.

**Definition 2.1.1.** Let $A : \Omega \to \mathbb{R}^{m\times n}$ and $B : \Omega \to \mathbb{R}^{m\times n}$. Then the *double dot product* is defined as

$$A : B = \sum_{j=1}^{n} \sum_{i=1}^{m} A_{ij} B_{ij}.$$

For higher-order tensors fields, the definition can be adjusted accordingly, e.g., for $C : \Omega \to \mathbb{R}^{m\times n\times k}$ we have

$$(A : C)_\ell = \sum_{j=1}^{n} \sum_{i=1}^{m} A_{ij} C_{ij\ell}, \qquad \ell \in \{1, \dots, k\}.$$

Taking the double dot product between two tensors of orders $p$ and $q$ will result in a tensor of order $p + q - 4$.

We continue with the definition of the trace of a second-order tensor field:

**Definition 2.1.2.** Let $A : \Omega \to \mathbb{R}^{n\times n}$. Then the *trace* of $A$ is defined as

$$\operatorname{tr} A = \sum_{i=1}^{n} A_{ii}.$$

Concluding this section, we introduce the definitions of differential operators for tensor and vector fields. Denoting $\mathcal{C}^1$ as the space of continuously differentiable functions, we define the divergence of a second-order tensor field, followed by the Jacobian of a vector field.

**Definition 2.1.3.** Let $A \in \mathcal{C}^1(\Omega, \mathbb{R}^{n\times d})$. Then the *divergence* of $A$ is defined as

$$(\operatorname{div} A)_i = \sum_{j=1}^{d} \partial_j A_{ij}, \qquad i \in \{1, \dots, n\}.$$

**Definition 2.1.4.** Let $A \in \mathcal{C}^1(\Omega, \mathbb{R}^{n})$. Then the *Jacobian* of $A$ is defined as

$$(\nabla A)_{ij} = \partial_j A_i, \qquad i \in \{1, \dots, n\}, \;\; j \in \{1, \dots, d\}.$$

## 2.2 Deep Learning

In contemporary society, *artificial intelligence* (AI) technology significantly impacts various aspects of our daily lives. Beyond its influence on consumer recommendation systems [19], web searches [20], weather forecasts [21], and autonomous driving [22, 23], AI algorithms are integral to everyday products like smartphones and cameras [24]. AI is a dynamic field within computer science and mathematics that focuses on creating intelligent systems capable of emulating human-like cognitive functions. This involves algorithms and models that enable machines to perform tasks traditionally associated with human intellect, such as learning from data, reasoning through complex problems, and understanding natural language.

In everyday discourse, the terms AI, *machine learning* (ML), and *deep learning* (DL) are frequently employed interchangeably. Despite their close association, it is essential to note that these concepts possess distinct characteristics. Figure 2.1 illustrates the relationship between AI, ML, and DL.

ML is a subset of AI aiming at learning patterns from data by using statistical methods for self-optimization. Instead of manually hard-coding programming software routines to execute a specific task, the machine undergoes a *training* process involving data. This training equips the machine with the capability to learn and autonomously perform the designated task.

DL, a subset of ML that gained significant attention in recent years, is inspired by the neuron structure of the human brain. It utilizes *neural networks* (NNs) with multiple layers to analyze and interpret data hierarchically, allowing for more complex and abstract pattern recognition. NNs are flexible parametric function approximators that excel in supervised learning. In the following, we will give a brief introduction to two different classes of NNs in Sections 2.2.1 and 2.2.2 and supervised training in Section 2.2.3. For a more comprehensive introduction and overview regarding DL methodologies, see [25, 26]. In the following section, we begin with an introduction to multilayer perceptrons.

### 2.2.1 Multilayer Perceptrons

*Multilayer perceptrons* (MLPs) constitute acyclic artificial NNs wherein information exclusively progresses forward from the input layer through the hidden layers to the output layer [25]. MLPs are usually employed for learning from data that is neither sequential nor time-dependent. An MLP is *dense*, i.e., each neuron establishes connections with all neurons in the subsequent layer. The building blocks of MLPs are *dense layers*, which we introduce in the following definition.

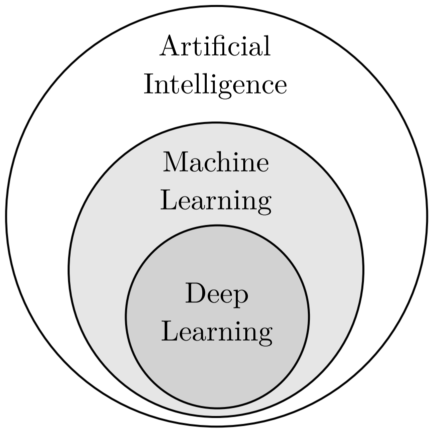


Figure 2.1: DL is a subset of ML, and ML is a subset of AI.

**Definition 2.2.1.** Let $n_0, \dots, n_L \in \mathbb{N}$. An MLP $f_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ with $L \in \mathbb{N}$ layers is defined by

$$f_\theta = f^L_{\theta_L} \circ \cdots \circ f^1_{\theta_1},$$

where $f^i_{\theta_i} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ are *dense* or *fully connected layers* given by

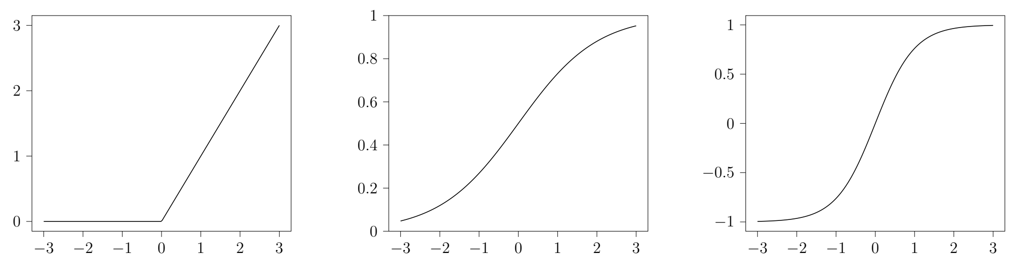
$$f^i_{\theta_i}(x) = \sigma_i(W_i x + b_i). \tag{2.1}$$

Here, $\sigma_i : \mathbb{R} \to \mathbb{R}$ is called *activation function* and is applied element-wise. In each layer, $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ is called *weight matrix* and $b_i \in \mathbb{R}^{n_i}$ is the *bias*. The vector $\theta_i$ denotes the *parameters* of the layer and contains all the entries in $W_i$ and $b_i$. For simplicity, we will write $\theta_i = [W_i, b_i]$. The network parameters are given by $\theta = [\theta_1, \dots, \theta_L]$. The first layer is called the *input layer*, and the last is the *output layer*. Intermediate layers are called *hidden layers*.

The activation function $\sigma_i$ in Equation (2.1) is usually fixed for each layer. One of the most widely used activation functions is the *rectified linear unit* (ReLU) [27] defined by $\sigma_i(x) = \max(0, x)$. Common activation functions are depicted in Figure 2.2. If no activation function is used (i.e., $\sigma_i$ is the element-wise identity function), we call the resulting layer a *linear layer*. The number of layers $L$ is called a *hyperparameter*. Hyperparameters are not learned directly from the training procedure (see Section 2.2.3) but are instead freely chosen a priori.

### 2.2.2 Convolutional Neural Networks

*Convolutional neural networks* (CNNs) represent a specialized category within feed-forward networks, characterized by their non-fully connected nature and utilization of weight sharing to achieve translation equivariance. CNNs can be considered a



(a) ReLU (b) Sigmoid (c) Tanh

Figure 2.2: Plots of the three most widely used activation functions.

restrained version of MLPs that find particular utility in processing structured grids like 2D or 3D images [28]. CNNs are capable of capturing high spatial correlation of neighboring pixels, which is not taken into account by dense layers. Demonstrating remarkable effectiveness, CNNs excel in diverse image recognition tasks like object detection [29], facial recognition [30], medical image analysis [31, 32], and video analysis [33, 34].

A fundamental component of CNNs is the integration of NNs as filters for a series of discrete convolutions followed by nonlinear mappings. The principle of weight sharing enables identical filters to be deployed at various locations across the input image, thereby reusing the same weights to extract features. Typically, multiple layers of such filters are stacked, incorporating additional operations like pooling, downsampling, or upsampling between layers. The collective structure of convolutional layers constitutes the CNN architecture.

**Definition 2.2.2.** Let $c_1, c_2 \in \mathbb{N}$ be the number of input and output channels, respectively. Further, let $\sigma : \mathbb{R} \to \mathbb{R}$ be an element-wise activation function and let $W \in \mathbb{R}^{c_1 \times c_2 \times m \times n}$ and $b \in \mathbb{R}^{c_2}$. In two dimensions and in its simplest form, i.e., without padding, stride, and dilation, a *convolutional layer* is defined as a mapping

$$f_\theta : \mathbb{R}^{c_1 \times h \times w} \to \mathbb{R}^{c_2 \times (h-m+1) \times (w-n+1)},$$

$$f_\theta(x)_{\tilde{c},i,j} = \sigma\left(\sum_{c=1}^{c_1} \sum_{r=1}^{m} \sum_{s=1}^{n} W_{c,\tilde{c},r,s} \cdot x_{c,(i+r-1),(j+s-1)} + b_{\tilde{c}}\right).$$

Here, $W$ represents the convolutional kernels, $b$ is the bias, and $\theta = [W, b]$ denotes the parameters of the layer. The input $x$ is a tensor with dimensions height $h \geq m$, width $w \geq n$, and number of channels $c_1$.

3D convolutions can be derived analogously to Definition 2.2.2. Note that there are various extensions to this standard multi-channel convolution, for example, using padding and using strided, grouped or separable convolutions [35, 36].

In addition to employing convolutional layers, most CNNs integrate downsampling layers to diminish spatial dimensions. This reduces computational costs and makes the NN invariant to small translations [25]. One of the most widely used architectures for diverse image-to-image tasks, such as image reconstruction and segmentation, is the U-Net architecture [32]. Derived from fully convolutional networks [37], the U-Net comprises two main components: a contracting and an expansive part. Contemporary iterations of the U-Net often incorporate attention mechanisms [38] within both the contracting and expansive sections.

### 2.2.3 Supervised Training

We will now discuss using NNs in a supervised training setting. In supervised learning, the objective is to deduce general rules from a given dataset $\mathcal{D} = \{(a_i, u_i)\}_{i=1}^N$ of size $N \in \mathbb{N}$. Here, $a_i$ and $u_i$ represent independent and identically distributed (i.i.d.) samples from a probability distribution over a joint sample space $\mathcal{A} \times \mathcal{U}$, i.e. $(a_i, u_i) \sim (\mathcal{A}, \mathcal{U})$. We formally define the learning task as identifying a function $f : \mathcal{A} \to \mathcal{U}$ that maps inputs to desired outputs.

We need to restrict ourselves to a specific hypothesis class of functions $\mathcal{H}$ to facilitate the optimization process. In the case of NNs, we obtain the parametric hypothesis class given by

$$\mathcal{H} = \{f_\theta : \mathcal{A} \to \mathcal{U}\,;\, \theta \in \Theta\}\,,$$

where $\Theta$ encompasses all possible network parameterizations. The hypothesis class can be restricted to include prior knowledge, referred to as *inductive bias*. Incorporating prior knowledge into the hypothesis class can drastically improve the generalization capabilities of the DL models, for example, through the use of group equivariant convolutions [39] (see Section 4.5).

In order to fit $f$ to the data, we introduce a loss function $\ell : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, which we intend to minimize over the training data pairs. This leads to the formulation of the *empirical risk*

$$\frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(a_i),\, u_i), \tag{2.2}$$

which is a Monte-Carlo approximation [40] of the *expected risk* $\mathbb{E}_{(a,u)\sim(\mathcal{A},\mathcal{U})}\, \ell(f_\theta(a), u)$. Once the empirical risk (2.2) is defined, the next step involves solving the optimization problem to minimize it with respect to the network's parameters $\theta$. This process is referred to as *training* the network. Since the empirical risk serves only as a surrogate for the actual risk, it is not necessary to minimize the empirical risk exactly. Often, early stopping rules are applied based on the empirical risk on a held-out validation dataset.

Since the parameter space $\Theta$ for NNs is typically high dimensional, finding a closed-form solution for the empirical risk minimization scheme is impractical. Further, computing the gradient of the empirical risk (2.2) requires evaluating the network for all examples in the dataset. This is prohibitive for large datasets with thousands or even millions of data points. Therefore, iterative numerical procedures are employed, particularly stochastic gradient descent methods [25, 41]. Variants of stochastic gradient descent, such as the widely used Adam algorithm [42], are commonly used in practical applications. The stochastic nature arises from minimizing the empirical risk (2.2) with respect to randomly chosen partial sums, referred to as *batches* [43]. The term *epoch* indicates the number of passes completed over the entire training dataset during the optimization process. For an in-depth exploration of optimization methods, refer to [25].

Lastly, using gradient-based optimization schemes requires the computation of the gradient of the loss with respect to the parameter vector $\theta$. This is achieved through the *backpropagation algorithm* [44, 45], which is based on an efficient implementation of the chain rule. See [46] for a comprehensive explanation of the backpropagation algorithm.

In addition to supervised learning, there exist many alternative learning tasks [25]. A notable example is *unsupervised learning*, where only inputs $\{a_i\}_{i=1}^N$ are provided without the associated labels. Typical unsupervised learning tasks include generative modeling, dimensionality reduction, or clustering.

# Chapter 3

# Classical Topology Optimization

A mechanical *structure*, as defined by J. E. Gordon [47], refers to "any assemblage of materials which is intended to sustain loads". Therefore, in the context of mechanics, *structural optimization* entails configuring a material assembly in an optimal way to support external loads efficiently. Defining what we mean by *optimal* is imperative to give significance to this pursuit. For instance, this may refer to *weight minimization*, prompting the search for the lightest possible structure. Conversely, one might interpret optimality as maximizing mechanical stiffness or resistance to buckling or instability.

The study of structural optimization can be traced back to Michell [48], who developed the classical theory of optimal structural layout design at the beginning of the 20th century. Since the 1960s, with the invention of the *finite element method* (FEM) and the advent of modern digital computers, structural optimization has been introduced naturally into engineering design practices to shorten the development cycle of products. Recently, driven by the increasing demands to design competitive products efficiently and reliably, structural optimization has become a growing field of research. Although initiated for mechanical design problems in civil and aerospace engineering, the field has spread to a wide range of other physical disciplines, including acoustics [49], fluids [50], optics [51] and electromagnetics [52].

Generally, we can divide structural optimization into three categories [53]:

- *Size optimization*: Size optimization involves determining the optimal size of individual components within a structure. It focuses on finding the most efficient size and thickness for each predefined structure element.

- *Shape optimization*: Shape optimization deals with optimizing the geometric configuration or contours of individual components within a structure. It aims to enhance a structure's performance by adjusting its elements' shapes.

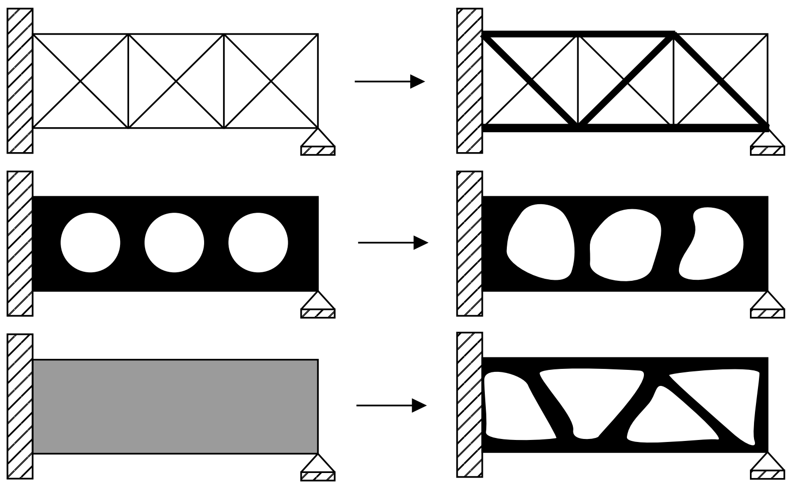


Figure 3.1: Categories of structural design optimization. Top: size optimization, middle: shape optimization, bottom: topology optimization.

- *Topology optimization* (TO): This is the most general form of structural optimization. TO involves determining the optimal distribution of material within a given design space. It explores the layout of structural elements, deciding where material should be placed and where voids or less material should exist to achieve the best structural performance.

For a visual comparison of the three categories of structural optimization, see Figure 3.1. Among the three categories, TO is the most powerful and, at the same time, most challenging one from both theoretical and numerical solution perspectives. Therefore, the study of TO has become the central research theme over the last three decades in the structural optimization community. Note that the key advantage of TO over shape or size optimization is that no specified initial structural topology needs to be presumed a priori.

In mathematical terms, the goal of TO is to find an optimal material distribution that minimizes a given objective function $\mathcal{J}$, subject to constraints $\mathcal{C}_k$ and governed by a so-called *state equation*, which is typically determined by an underlying PDE. Let the design domain $\Omega$ denote a fixed smooth open domain of $\mathbb{R}^d$ for $d \in \{2, 3\}$. The material distribution in $\Omega$ is described by a binary point-wise *density function* $\rho : \Omega \to \{0, 1\}$ that can take either the value 0 (void) or 1 (solid material). To keep the definition of the TO problem as general as possible, we introduce a *hypothesis space* $\mathcal{H}$ that defines the set of admissible density functions. The classical mathematical formulation of the corresponding *TO problem* is then given by

$$\min_{\rho \in \mathcal{H}} \mathcal{J}(u(\rho), \rho) \tag{3.1}$$

subject to:

$$\mathcal{C}_0(\rho) = \frac{1}{|\Omega|} \int_\Omega \rho \, \mathrm{d}\Omega \leq V_{\max}, \tag{3.2}$$

$$\mathcal{C}_k(u(\rho), \rho) \leq 0, \quad k \in \{1, \ldots, N\}, \tag{3.3}$$

$$\text{state equation}, \tag{3.4}$$

where the constraint functional $\mathcal{C}_0$ represents a *volume constraint* based on a maximum *volume fraction* $V_{\max} > 0$. The *state field* $u$ satisfies the state equation (3.4), usually given by linear elasticity (see Chapter 3.1). Additional constraint equations (3.3) can optionally be incorporated to impose further conditions on the design variables.

## 3.1 The Linear Elasticity Problem

Finding an optimized material distribution for a specific design problem involves consideration of the underlying mechanical forces, boundary conditions, and material properties. In the following, we discuss incorporating physical information into the TO problem (3.1)-(3.4). This is realized via the state equation, wherein, for structural TO, the governing PDE corresponds to *linear elasticity.* This PDE can be considered a simplification of the general nonlinear theory for elasticity.

According to Oliver and de Saracibar [54], the simplifying hypotheses of the linear elasticity theory are fulfilled if:

1. The displacements and their gradients are small, resulting in small strains.
2. There exists a neutral state in which the strains and stresses are zero.
3. The deformation process is considered to be isothermal and adiabatic, i.e., temperature effects and heat generation can be neglected.

In the following, we assume these hypotheses to be fulfilled, resulting in a linear relationship between stresses and strains (see Figure 3.2).

We will develop various formulations of the linear elasticity problem in the upcoming sections. Beginning with Section 3.1.1, we initiate the discussion by introducing the strong formulation of the PDE. Subsequently, in Sections 3.1.2 and 3.1.3, we will establish the variational and discretized formulations, which will serve as the foundation for numerical implementations.

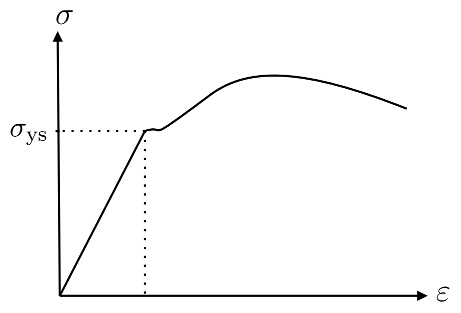


Figure 3.2: A typical stress-strain curve of a low-carbon steel in one dimension. The curve demonstrates linear elasticity for small strains, with the slope given by Young's modulus $E$. In higher dimensions, this linear relationship between stresses and strains is given via the elastic constitutive tensor field $\mathbb{C}$, which depends linearly on $E$. The transition from elastic to plastic deformation marks the initiation point, with the stress at this juncture defined as the yield stress $\sigma_{\text{ys}}$.

### 3.1.1 Strong Formulation

In the linear elasticity setting, the state equation (3.4), expressed through a PDE, yields the *displacement field* $u(\rho) : \Omega \to \mathbb{R}^d$ as its solution, where $\rho \in \mathcal{H}$ denotes a specific density configuration. This displacement field must satisfy the boundary conditions applied to the design domain $\Omega$. Specifically, the *isotropic linear elasticity problem* [55],

Find the displacement field $u(\rho) : \Omega \to \mathbb{R}^d$ such that:

$$\begin{cases} -\text{div}\, \sigma(u(\rho), \rho) = f & \text{in } \Omega \\ u(\rho) = 0 & \text{on } \partial\Omega_D \\ \sigma(u(\rho), \rho) \cdot n = g & \text{on } \partial\Omega_N \end{cases} \tag{3.5}$$

is considered the state equation for all structural TO problems addressed in this thesis. Here, $\sigma(u(\rho), \rho) : \Omega \to \mathbb{R}^{d\times d}$ and $f : \Omega \to \mathbb{R}^d$ represent the second-order symmetric *Cauchy stress tensor field* and the vector field of *volumetric forces*, respectively. The outer boundary $\partial\Omega$ comprises two mutually disjoint subsets: the *Dirichlet boundary* $\partial\Omega_D$ and the *Neumann boundary* $\partial\Omega_N$. Adopting the prevalent approach in linear elasticity, we exclusively focus on homogeneous Dirichlet conditions. Along the Neumann boundary, we specify surface tractions $g$ in alignment with the unit outward normal $n$ (see Figure 3.3 for an illustration).

For isotropic materials, the relationship between $\sigma$ and $u$ is governed by *Hooke's law* [56], i.e.,

$$\begin{aligned} \sigma(u(\rho), \rho) &= \mathbb{C}(\rho) : \varepsilon(u(\rho)) := \lambda(\rho)\, \text{tr}\, [\varepsilon(u(\rho))]\, \mathbb{I}_d + 2\mu(\rho)\, \varepsilon(u(\rho)), \\ \varepsilon(u(\rho)) &= \nabla^S u(\rho) := \frac{1}{2}\left(\nabla u(\rho) + (\nabla u(\rho))^T\right), \end{aligned} \tag{3.6}$$

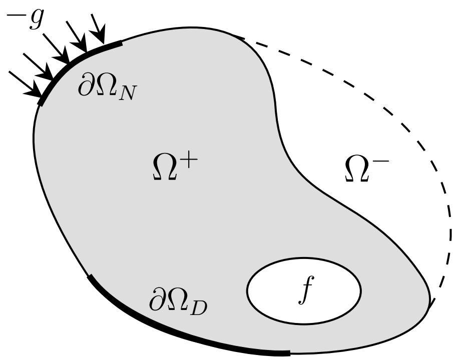


Figure 3.3: Sketch of an elasticity problem: We consider a design space $\Omega = \Omega^+ \cup \Omega^-$ with homogeneous Dirichlet boundary conditions on $\partial\Omega_D$ and Neumann boundary conditions on $\partial\Omega_N$, represented by normal tractions $g$. Volumetric forces are indicated by a source term $f$.

where $\lambda(\rho)$, $\mu(\rho) : \Omega \to \mathbb{R}_{>0}$ are the *Lamé elasticity parameters* of the material and $\mathbb{I}_d$ is the $\mathbb{R}^{d\times d}$-identity operator. $\varepsilon$ is the symmetric *strain tensor field* and $\mathbb{C}(\rho)$ is called the fourth-order *elastic constitutive tensor field*. For the definition of tensor operators like the double dot product, trace and divergence of a tensor field, and the Jacobian of a vector field, see Section 2.1.3.

Instead of using the Lamé parameters, Hooke's law (3.6) can equivalently be expressed in terms of *Young's modulus* $E(\rho) \in \mathbb{R}_{>0}$ and *Poisson's ratio* $\nu \in [0, {}^1\!/_2)$ of the material, with

$$\lambda(\rho) = \frac{E(\rho)\,\nu}{(1-2\nu)\,(1+\nu)} \quad \text{and} \quad \mu(\rho) = \frac{E(\rho)}{2\,(1+\nu)}. \tag{3.7}$$

Young's modulus quantifies a material's stiffness by representing the stress-strain ratio during elastic deformation (see Figure 3.3). Poisson's ratio characterizes a material's compressibility. A value of ${}^1\!/_2$ signifies a perfectly incompressible scenario, which we prohibit in this thesis. Note that from combining Equations (3.6) and (3.7) it follows that $\mathbb{C}(\rho)$ is linear with respect to Young's modulus $E(\rho)$. Additionally, we assume $E$ to depend linearly on $\rho$. This leads to

$$\mathbb{C}(\rho) = \rho\,\mathbb{C}^+, \tag{3.8}$$

where $\mathbb{C}^+$ represents the elastic constitutive tensor field for the stiff material, remaining independent of $\rho$. The relationships between design variables like $\rho$ and structural responses determined by the laws of mechanics, like $u$, are commonly referred to as *sensitivities* [57].

The predominant optimization objective in traditional TO involves minimizing the *compliance* of designs. The compliance objective seeks an optimal arrangement of topology that enhances overall *stiffness* or minimizes the work applied to the structure. Mathematically, we can express the compliance objective as

$$\mathcal{J}(u(\rho), \rho) = \langle u(\rho), f \rangle, \tag{3.9}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product within the considered Hilbert space, such as $L_2(\Omega, \mathbb{R}^d)$ or $\mathbb{R}^d$. Some studies explore alternative optimization objectives, e.g., volume minimization accompanied by a stress constraint is notably discussed in the literature [58, 59, 60, 61]. However, compliance minimization is dominant, presumably due to its simple formulation and good practical results. Our research aligns with this observation, revealing that the pursuit of compliance minimization frequently yields structures that exhibit enhanced mechanical efficiency, demonstrating superior stiffness and load-carrying capacity.

## 3.1.2 Variational Formulation

In order to obtain the variational formulation of the isotropic linear elasticity problem, we multiply both sides of the PDE with a test function $v \in \mathcal{U} := \left\{ v \in H^1(\Omega, \mathbb{R}^d)\,;\, v|_{\partial\Omega_D} = 0 \right\}$, integrate over the domain $\Omega$ and apply Green's theorem. We define vector-valued Sobolev spaces analogously to the scalar-valued case:

**Definition 3.1.1.** Let $d, n \in \mathbb{N}$ and $\Omega \subset \mathbb{R}^d$. Then the *vector-valued Sobolev space* $H^1(\Omega, \mathbb{R}^n)$ is given by

$$H^1(\Omega, \mathbb{R}^n) := \left\{ u \in L^2(\Omega, \mathbb{R}^n)\,;\, \partial_j u \in L^2(\Omega, \mathbb{R}^n) \ \ (j \in \{1, \ldots, d\}) \right\}.$$

The variational formulation of the linear elasticity problem (3.5) then reads as follows:

$$\text{Find the displacement field } u(\rho) \in \mathcal{U} \text{ such that}$$
$$\begin{cases} a(u(\rho), v) = L(v) \qquad \forall v \in \mathcal{U}, \\ \quad \text{where} \\ a(u(\rho), v) = \displaystyle\int_\Omega \nabla^S u(\rho) : \mathbb{C}(\rho) : \nabla^S v \ \mathrm{d}\Omega, \\ \quad L(v) = \displaystyle\int_\Omega f v \ \mathrm{d}\Omega + \int_{\partial\Omega_N} g v \ \mathrm{d}S \end{cases} \tag{3.10}$$

| property of the PDE | solution operator's kernel | matrix structure |
|---|---|---|
| periodic BCs & const. coeffs | convolutional kernel | circulant |
| localized behavior | off-diagonal decay | banded |
| elliptic / parabolic | off-diagonal low rank | hierarchical |

Table 3.1: Solution operators associated with linear PDEs can often be represented as integral operators. Depending on the PDE, the solution operator's integral kernel has unique properties, which impose a certain matrix structure in the discretized case [62].

### 3.1.3 Discretized Formulation

By using a discretization scheme like the FDM or FEM, the variational formulation (3.10) can be reformulated in the following discretized problem formulation,

Find the nodal displacement vector $\hat{u}(\rho)$ such that

$$\begin{cases} \mathbb{K}(\rho)\hat{u}(\rho) = F \\ \quad \text{where} \\ \quad \mathbb{K}(\rho) = \int_\Omega B^T \mathbb{C}(\rho) B \ \mathrm{d}\Omega, \\ \quad F = \int_\Omega N^T f \ \mathrm{d}\Omega + \int_{\partial\Omega_N} N^T g \ \mathrm{d}S \end{cases} \tag{3.11}$$

where $\mathbb{K}(\rho)$ is called the *stiffness matrix* of the system. For $x \in \Omega$, the full displacement field $u(\rho)$ and its symmetric gradients $\nabla^S u(\rho)$ are approximated using the *nodal displacements* $\hat{u}(\rho)$ as follows

$$\begin{aligned} u(\rho)(x) &= N(x)\,\hat{u}(\rho) \\ \nabla^S u(\rho)(x) &= B(x)\,\hat{u}(\rho) \end{aligned}$$

where $N$ and $B$ are called the *displacement shape function matrix* and the *strain-displacement matrix*, respectively.

Note that Equation (3.8) allows the following reformulation of the stiffness matrix:

$$\mathbb{K}(\rho) = \int_\Omega B^T \rho \mathbb{C}^+ B \ \mathrm{d}\Omega. \tag{3.12}$$

As $B$, $\mathbb{C}^+$, and $F$ remain unaffected by variations in $\rho$, they can be assembled once for any given TO problem and subsequently be re-utilized for future density changes. Further, subsequent calculations of the tensor products and solutions of the linear system can be significantly accelerated by exploiting sparsity. The structure of $\mathbb{K}(\rho)$ is inherited from the properties of the PDE (see Table 3.1). Since the PDE for linear elasticity is elliptic and has localized behavior, the matrix has a banded and hierarchical structure (see Figure 3.4 for a visualization of the typical matrix structure arising in TO).

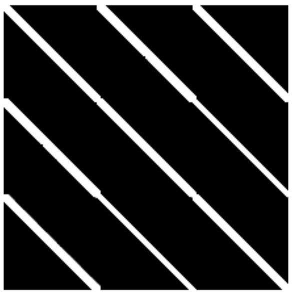


Figure 3.4: Visualization of the typical sparse banded structure of the stiffness matrix $\mathbb{K}(\rho)$. In the image, zero values are displayed in black, and non-zero values in white.

### 3.1.4 Solving the Linear System

In density-based TO, the computational burden primarily lies in solving the linear system derived from the discretized PDE formulation (3.11). This computational task often constitutes the most resource-intensive aspect of the numerical algorithms, consuming as much as 97% of the total computational time [63]. In this section, we will provide a brief overview of potential linear solvers employed to address this PDE.

Linear solvers can be categorized into *iterative* and *direct* methods. Iterative methods start with an initial guess and iteratively apply update steps until a specified convergence criterion is met within a certain tolerance. Examples include Jacobi iteration, Krylov subspace methods like GMRES [64] and the preconditioned conjugate gradient method [65], as well as multigrid methods [66]. Direct solvers decompose a matrix into a form that allows for a direct inversion. Standard direct methods include UMFPACK [67], MUMPS [68], and traditional matrix decompositions like the LU, Cholesky, and QR decomposition. The convergence rate of linear solvers usually depends on the matrix's condition number. To enhance convergence performance, *preconditioning* is frequently employed on the matrix, aiming to reduce its condition number substantially.

To ensure the effectiveness of any method for solving sparse linear systems, exploiting the sparsity inherent in the system matrix is essential. Iterative techniques, like Krylov subspace methods, excel in this aspect, relying on matrix-vector multiplication to reach a solution. In contrast, incorporating sparsity into a direct method is typically more intricate, necessitating a profound understanding of the inherent sparsity structure within the specific problem [69].

As the number of degrees of freedom increases, the computational cost of direct solvers also increases significantly, often leading to memory and processing time limitations. In contrast, iterative solvers can be more efficient for large systems, especially if the coefficient matrix exhibits specific properties like sparsity. In the work carried out for this thesis, we have found that for less than approximately $10^5$ degrees of freedom, a sparse UMFPACK solver is efficient at solving sparse linear systems. Beyond this range, iterative solvers are often more practical and scalable.

## 3.2 Methods

The field of TO has significantly developed since the seminal paper by Bendsøe and Kikuchi [1] in 1988. Since then, TO has evolved in both theoretical exploration and practical applications [70, 71, 72, 73]. Various literature surveys [74, 75, 76] have comprehensively documented recent advancements and real-world applications of TO.

Initially, TO has been regarded as a binary optimization problem, which is known to be ill-conditioned in the context of structural compliance-related designs [77, 78]. The primary challenge in this context lies in solving a large-scale integer programming problem, where the high computing cost typically precludes the use of gradient-free algorithms [79, 80]. In 1988, Bendsøe and Kikuchi [1] revolutionized the field by introducing the *homogenization* approach. This approach transformed the identification of optimal topology into a size optimization problem, utilizing geometry parameters as design variables to describe the microstructure of materials. While initially promising, the homogenization method demanded extensive computational resources and has gradually given way to alternative methods.

The TO problem described by Equations (3.1)–(3.4) has traditionally been tackled using one of two methodologies: either through a density-based approach, exemplified by SIMP or evolutionary methods, or as a boundary optimization problem, as seen in level-set methods. These methodologies are commonly known as *Eulerian* (fixed mesh) or *Lagrangian* (mesh follows the boundary) approaches [76].

In the following sections, we explore the SIMP method in Section 3.2.1, which stands out as the predominant technique for TO. Additionally, in this section, we present a proof for a novel insight, contending that the smoothed Heaviside function step within the SIMP algorithm can be equivalently interpreted as a proximal mapping addressing an optimization problem featuring a concave penalty function (see Theorem 3.2.1). Subsequently, we give an overview of evolutionary approaches in Section 3.2.2 and the level-set method in Section 3.2.3. Finally, in Section 3.3, we provide an evaluative comparison of the proposed methods.

### 3.2.1 SIMP Method

Arguably, the most common class of classical approaches for TO are *density-based* methods. Density-based methods aim to minimize the optimization objective by

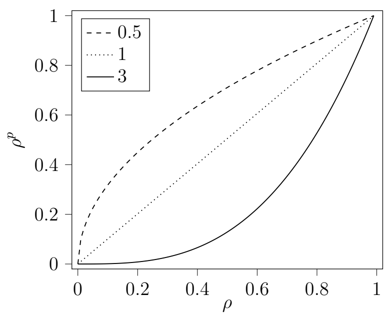


Figure 3.5: The SIMP power law $\rho^p$ for different values of $p$. Compared to the linear case $p = 1$, higher values of $p$ lead to lower entries in the elastic constitutive tensor $\mathbb{C}(\rho)$ for intermediate density values. This leads to a penalization of non-binary densities for the overall stiffness.

directly adjusting the material's density distribution $\rho : \Omega \to \{\rho_{\text{min}}, 1\}$ over a discretized voxel grid on $\Omega$. Here, values of 1 correspond to solid elements, and $\rho_{\text{min}} > 0$ is a minimum density value representing void elements, with a typical choice being $\rho_{\text{min}} = 10^{-3}$. A non-zero minimum density is required to prevent singularities in the stiffness matrix [74]. Since the exploration of all possible density configurations for even small numbers of elements (e.g., 1000) involves a prohibitively large number of analyses ($2^{1000} \approx 10^{301}$), density methods always resort to iterative approaches.

An essential consideration in density-based methods involves choosing a suitable density interpolation scheme and penalization technique to represent $\rho$ as a continuous design variable, $\rho(\cdot) \in [\rho_{\text{min}}, 1]$. This offers significant advantages over a discrete formulation, especially when dealing with a large number of variables, which is a common scenario in TO [81]. With the choice of continuous density parameterization comes the need to steer $\rho$ towards a solid-void solution. This is typically accomplished by using implicit penalization techniques, the most common of which is the *solid isotropic material with penalization* (SIMP) method [82, 83, 84]. In the SIMP method, intermediate density values are penalized via a *power law* when computing the optimization objective,

$$\mathbb{C}(\rho) = \rho^p\, \mathbb{C}^+, \qquad \rho(\cdot) \in [\rho_{\text{min}}, 1]. \tag{3.13}$$

where $\mathbb{C}^+$ represents the elastic constitutive tensor for the stiff material (see Equation (3.8)). The exponent $p$ controls the penalization of non-binary densities. For $p = 1$, the compliance minimization problem is convex and, therefore, has a unique solution [71, 85]. For the same objective, a choice of $p > 1$ penalizes intermediate densities and favors binary solutions. The established choice of the SIMP exponent is $p = 3$, which was first proposed by Bendsøe and Sigmund [71]. A higher value of $p$ yields more binary solutions, albeit at the expense of making the optimization problem more challenging to solve (see Figure 3.5). Additionally, it exacerbates the

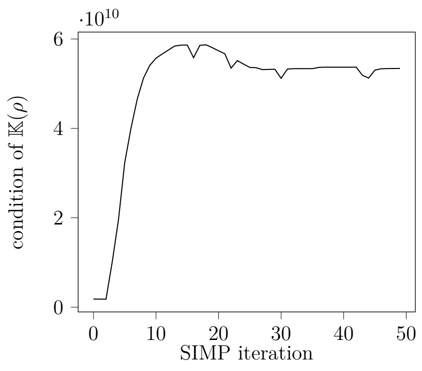


Figure 3.6: Condition numbers of the stiffness matrix $\mathbb{K}(\rho)$ over the course of SIMP iterations, on a problem from the disc complex dataset [86]. We can observe that the linear system arising from the PDE for linear elasticity is highly ill-conditioned.

ill-conditioning of the stiffness matrix. Throughout SIMP iterations on an arbitrary TO problem from the *disc complex* dataset [86], the condition number frequently exceeds $10^{10}$, as depicted in Figure 3.6. From Equations (3.12) and (3.13) it follows that the global stiffness matrix $\mathbb{K}(\rho)$ can be obtained via

$$\mathbb{K}(\rho) = \int_\Omega B^T \rho^p \, \mathbb{C}^+ B \; \mathrm{d}\Omega.$$

Instead of employing the implicit SIMP penalization scheme, an alternative is to opt for explicit penalization. This involves incorporating the term

$$\int_\Omega \rho \, (1 - \rho) \; \mathrm{d}\Omega$$

into the objective function [87, 88]. However, determining a suitable weighting factor to ensure smooth convergence to binary designs proves challenging, leading to limited popularity for this approach.

Algorithm 1 presents a pseudo-code implementation of the SIMP method. It is essential to highlight that computing the gradient steps involves determining partial derivatives $\partial u/\partial \rho$ through the PDE solver. However, the efficiency of this process can vary depending on the specific implementation of the solver, and it may be considerably slow or even impossible. A commonly employed solution is the application of the *adjoint method* [89]. This method entails solving a single linear system to compute the partial derivatives in each iteration, with the system matrix being the adjoint of the forward pass. Consequently, the adjoint method allows for the efficient calculation of sensitivities. For a deeper understanding of the adjoint method and its application in differentiable physics, see [90, 91, 92].

An interesting alternative perspective on SIMP involves interpreting $\rho$ as a probability density, where $\rho(x)$ denotes the likelihood of $x \in \Omega$ being occupied by material. Thus,

**Algorithm 1** Solid isotropic material with penalization (SIMP) method

```
Require: $F$, $p$, $N$                          ▷ Forces, SIMP exponent and #iterations
Initialize: $\rho_0$                           ▷ Start with an initial density distribution
for $i = 0, \ldots, N-1$ do
    $\rho_i \leftarrow$ project($\rho_i$)                  ▷ Project density values into the unit interval
    $\tilde{\rho}_i \leftarrow$ filter($\rho_i$)                   ▷ Smooth densities to avoid checkerboard patterns
    $\tilde{\rho}_i \leftarrow$ binarize($\tilde{\rho}_i$)                 ▷ Binarize using a smoothed Heaviside function
    $u$, $\sigma_{vM}$ = pde_solver($F$, $\tilde{\rho}_i^p$)         ▷ Solve PDE for $\tilde{\rho}_i$ and exponent $p$
    loss = criterion($F$, $u$, $\tilde{\rho}_i$)         ▷ Evaluate optimization objective
    $\rho_{i+1} \leftarrow$ gradient_step($\rho_i$, loss)   ▷ Update densities via gradient descent
end for
$\rho_N \leftarrow$ project($\rho_N$)
$\rho_N \leftarrow$ filter($\rho_N$)
$\rho_N \leftarrow$ binarize($\rho_N$)
return $\rho_N$                                ▷ Return final density distribution
```

by sampling from the Bernoulli distribution $\mathcal{B}$, each sample $s \sim \mathcal{B}(\rho)$ represents a binary material distribution. Given that $\mathbb{E}_{s\sim\mathcal{B}(\rho)}(s) = \rho$, the conventional non-probabilistic interpretation of SIMP emerges as the scenario where averaging is performed over an infinite number of samples $s$. However, akin to the principles of stochastic gradient descent, employing a finite number of samples enables a Monte Carlo approximation of the expected value, thereby aiding in circumventing local optimization minima. Notably, when considering only one sample, we obtain binary material densities despite the continuous design variable represented by the probability density $\rho$. To our knowledge, this probabilistic perspective on SIMP has not been explored in the existing literature and represents an entirely novel approach.

*Regularization* in TO refers to controlling the density values or sensitivities to prevent rapid oscillations of the density distribution. Most notably, a common issue in density-based TO is *mesh dependency*, which concerns the phenomenon that different discretization sizes result in different topologies. This may lead to *checkerboarding*, which refers to the formation of adjacent solid-void elements arranged in a checkerboard pattern (see Figure 3.7). The two primary methods of regularization are filtering and constraint techniques.

Filtering methods are applied via direct modification of density variables or sensitivities, while constraint methods utilize localized or global-level constraints added to the optimization problem. Standard constraint methods include adding a total variation penalty term to the objective function [93, 94] and regularized penalty methods [94]. Filtering methods are widely regarded as the prevalent regularization technique for TO, mainly due to their ease of application. Notable examples encompass the *sensitivity filter* [95] and the *density filter* [96, 97], which modify either the sensitivity

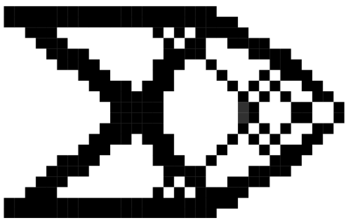


Figure 3.7: Checkerboard pattern in a solution obtained from the SIMP method. We can avoid checkerboarding via regularization. Image from [98].

or the density value of an element based on the sensitivity or density of elements in a localized neighborhood.

A basic consequence of filtering is the formation of gray transition material between solid and void regions. Consequently, several binarization schemes have been developed to promote near-binary solutions via a *smoothed Heaviside function* $\mathcal{H}_\lambda$, with a steepening factor $\lambda > 0$ [99, 100, 101]. A common definition for $\mathcal{H}_\lambda$ is given by

$$\begin{aligned} \mathcal{H}_\lambda : [0,1] &\longrightarrow [0,1] \\ x &\longmapsto \frac{\tanh\left(\frac{\lambda}{2}\right) + \tanh\left(\lambda x - \frac{\lambda}{2}\right)}{2\tanh\left(\frac{\lambda}{2}\right)}, \end{aligned} \tag{3.14}$$

where higher values of $\lambda$ correspond to steeper slopes. It is common to start with a small value for $\lambda$ and to increase $\lambda$ throughout iterations to obtain crisp binary solutions [93]. The advantage of the specific parameterization in (3.14) is that for $x \in [0,1]$ the Heaviside approximation $\mathcal{H}_\lambda$ closely approximates the identity mapping for small $\lambda > 0$, which is a well-suited initialization. In the following theorem, we show that filtering with a smoothed Heaviside function is not only a heuristic method but also has mathematical justification. To our knowledge, this constitutes a novel finding that has not been investigated prior to this thesis.

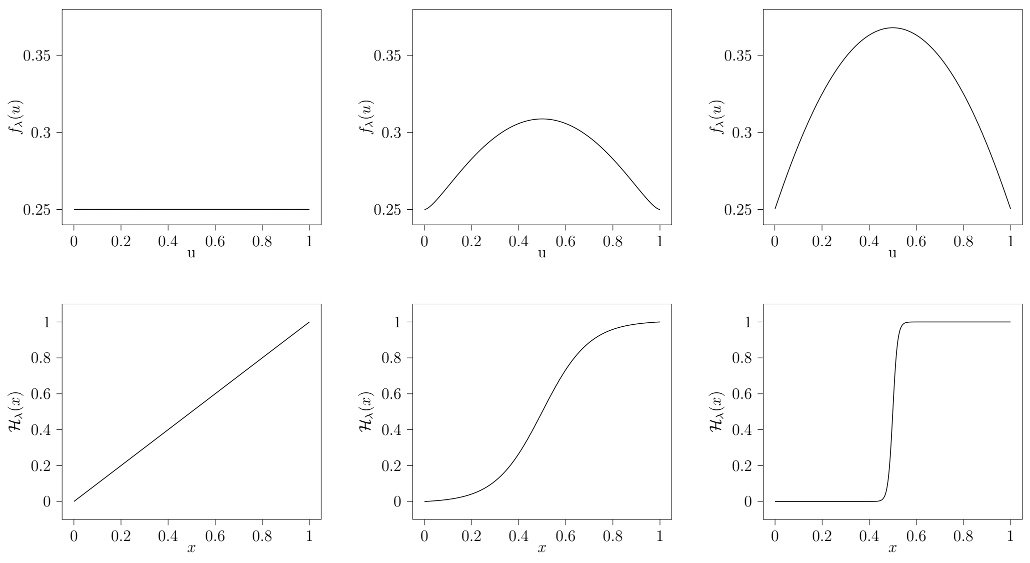
**Theorem 3.2.1.** *Let $\lambda > 0$ and let $\mathcal{H}_\lambda$ be the smoothed Heaviside function given by* (3.14)*. Then there exists a differentiable concave function $f_\lambda$ such that*

$$\mathcal{H}_\lambda(x) = \operatorname{prox}_{f_\lambda}(x) := \operatorname*{arg\,min}_{u\in[0,1]} \left( f_\lambda(u) + \frac{1}{2}\left\|u - x\right\|_2^2 \right)$$

*for all $x \in [0,1]$.*

*Proof.* Let $\lambda > 0$ and $x \in [0,1]$. The smoothed Heaviside function $\mathcal{H}_\lambda$ is bijective and continuous and, therefore, continuously invertible, with the inverse given by

$$\mathcal{H}_\lambda^{-1}(u) = \frac{1}{2\lambda}\left(\lambda + 2\tanh^{-1}\left((-1+2u)\tanh\left(\frac{\lambda}{2}\right)\right)\right)$$



(a) $\lambda = 0.1$ (b) $\lambda = 5$ (c) $\lambda = 50$

Figure 3.8: Top: penalty functions $f_\lambda$ in the optimization objective $\frac{1}{2}\|u - x\|_2^2 + f_\lambda$ for different values of $\lambda$. Bottom: associated smoothed Heaviside functions $\mathcal{H}_\lambda$ that are given by the respective proximal mapping $\mathcal{H}_\lambda = \operatorname{prox}_{f_\lambda}$.

for $u \in [0, 1]$. Note that $\mathcal{H}_\lambda^{-1}$ is differentiable and has the following derivative:

$$\mathcal{H}_\lambda^{-1'}(u) = \frac{2}{\lambda} \cdot \frac{\tanh\left(\frac{\lambda}{2}\right)}{1 - (2u-1)^2 \tanh^2\left(\frac{\lambda}{2}\right)} \in (0, 1].$$

Since $\mathcal{H}_\lambda^{-1} - \mathrm{id}$ is continuous there exists a differentiable function $f_\lambda$ with the derivative $f_\lambda'(u) = \mathcal{H}_\lambda^{-1}(u) - u$. Let $g_\lambda(u) := f_\lambda(u) + \frac{1}{2}\left\|u - x\right\|_2^2$ and let $u^\star := \mathcal{H}_\lambda(x)$. Then

$$\begin{aligned} 0 &= f_\lambda'(u^\star) + u^\star - \mathcal{H}_\lambda^{-1}(u^\star) \\ &= f_\lambda'(u^\star) + u^\star - x \\ &= g_\lambda'(u^\star). \end{aligned}$$

The function $f_\lambda$ is concave but $g_\lambda$ is strictly convex, since

$$\begin{aligned} f_\lambda''(u) &= \mathcal{H}_\lambda^{-1'}(u) - 1 \le 0 \quad \text{and} \\ g_\lambda''(u) &= f_\lambda''(u) + 1 \\ &= \mathcal{H}_\lambda^{-1'}(u) > 0 \end{aligned}$$

for all $u \in [0, 1]$. From the convexity of $g$, it follows that $u^\star = \mathcal{H}_\lambda(x)$ is the unique minimizer of $g_\lambda$. □

Theorem 3.2.1 shows that we can interpret the application of a smoothed Heaviside function $\mathcal{H}_\lambda$ as a proximal mapping that solves an optimization problem with a concave penalty function $f_\lambda$. The higher the value of $\lambda$, the more $f_\lambda$ punishes intermediate densities. See Figure 3.8 for a visualization of $f_\lambda$ for different smoothed Heaviside functions. Note that we employ non-convex penalization while maintaining the convexity of the total cost function, similar to the convex-non-convex strategies proposed by Lanza et al. [102].

The proof of Theorem 3.2.1 presents a concrete guideline for how to construct a suitable penalty function $f_\lambda$ for a given value of $\lambda > 0$. In fact, we can obtain $f_\lambda$ by integrating $\mathcal{H}_\lambda^{-1}(u) - u$ which results in the rather complicated formula

$$\begin{aligned} f_\lambda(u) = \frac{1}{\lambda}\Bigg(\lambda u\left(\frac{1-u}{2}\right) + \frac{\cosh\left(\frac{\lambda}{2}\right)}{\sinh\left(\frac{\lambda}{2}\right)}\Bigg(\frac{1}{4}\ln\left(2u^2 + (2-2u)u\cosh(\lambda) - 2u + 1\right) \\ -\frac{1}{2}\sqrt{\frac{\cosh(\lambda)-1}{\cosh(\lambda)+1}}\operatorname{arctanh}\left((2u-1)\sqrt{\frac{\cosh(\lambda)-1}{\cosh(\lambda)+1}}\right)\Bigg) \\ +u\operatorname{arctanh}\left((2u-1)\tanh\left(\frac{\lambda}{2}\right)\right)\Bigg). \end{aligned}$$

Finally, some theoretical convergence properties of SIMP have been discussed in the literature [103, 104, 105]. For additional references concerning both the origins and theoretical mechanisms behind the SIMP method, see [106, 107, 108].

### 3.2.2 ESO Method

The *evolutionary structural optimization* (ESO) method was first introduced by Xie and Steven [109, 110, 111] based on similar ideas presented in [112, 113]. Over the years, ESO has gained popularity due to its simplicity and ease of implementation. Starting from a fully stiff design, voxel-wise sensitivities are calculated via a user-defined rejection criterion, usually based on compliance (see Equation (3.9)) or von Mises stresses [55, 114]. Elements with a low rejection criterion value are gradually removed, leading to an optimized material configuration. This is summarized in Algorithm 2.

ESO is considered a *hard-kill method*, i.e., removed elements are not included in the subsequent PDE analyses. Consequently, no criterion function is computed for the void elements. Unlike the SIMP method, ESO utilizes a discrete element design variable, $\rho(\cdot) \in \{0, 1\}$, to define binary topology layouts. This alteration in the design variable may lead to convergence issues and a high dependency on the initial configuration, often resulting in suboptimal solutions [115, 116]. Also, ESO is entirely based on engineering heuristics and has no proof of optimality [117]. Despite these challenges, ESO has been widely applied across various problem domains, including structural [118, 119, 120], nonlinear [121, 122], thermal [123, 124, 125], and contact problems [126, 127].

**Algorithm 2** Evolutionary structural optimization (ESO) method

```
  Require: $F$, $N$, $r$                              ▷ Forces, #iterations and rejection ratio
  Initialize: $\rho_0$                                ▷ Start with an initial density distribution
  for $i = 0, \ldots, N-1$ do
      $u = \text{pde\_solver}(F, \rho_i)$             ▷ Solve PDE for current densities
      $s = \text{criterion}(u, F, \rho_i)$            ▷ Calculate sensitivities based on rejection criterion
      $s \leftarrow \text{filter}(s)$                 ▷ Optional: Filter sensitivities
      for each element $j$ do
          if $s_j \leq r \min(s)$ then $(\rho_i)_j = 0$   ▷ Remove elements with low sensitivity
          end if
      end for
      $\rho_{i+1} \leftarrow \rho_i$                  ▷ Update densities
  end for
return $\rho_N$                                       ▷ Return final density distribution
```

Addressing the limitations of the original ESO, the *bi-directional evolutionary structural optimization* (BESO) method [128, 129, 130] was introduced in 1999. BESO extends ESO by allowing both the removal of inefficient elements and the concurrent addition of new elements. New material is typically introduced in void areas near elements with high criterion function values [128, 129, 130].

Efforts to alleviate mesh dependencies in hard-kill methods involve the application of density regularization techniques [131, 132, 133], akin to those used in SIMP. However, these approaches often fail to achieve convergent solutions. As an alternative to hard-kill ESO/BESO methods, *soft-kill* approaches retain void elements as very soft elements, allowing the computation of the criterion function throughout the entire domain.

In modern BESO implementations, using a power law approach like in the SIMP method is common, enhancing numerical stability and method potential [134]. In this context, the design variable $\rho(\cdot) \in \{\rho_{\min}, 1\}$, is modified with a penalization factor $p$ using the original SIMP material interpolation (3.13).

There have been attempts to analyze the convergence of ESO. For example, Tanskanen [114] has shown that ESO updates follow the same optimization path as a form of the simplex algorithm would take. This type of analysis gives a theoretical basis for ESO as an optimization algorithm. However, it does not address the fact that it is using a linear programming method to optimize a nonlinear function. In 2013, Browne introduced a mesh refinement technique to reduce the nonmonotonic convergence behaviour [117].

### 3.2.3 Level-Set Method

The *level-set method* was originally developed as a mathematical tool for computing the motion of interfaces in two or three dimensions [135, 136]. Many formulations of level-set-based approaches have been proposed over the years since Haber and Bendsøe [137] suggested its application to TO, e.g., [136, 138, 139]. In contrast to density-based approaches, the optimal layout is implicitly defined by a scalar *level-set function* (LSF) $\phi : \Omega \to \mathbb{R}$. This is achieved by representing the boundary of the design by the zero-level iso-contour of the LSF [140, 141],

$$\partial\Omega^+ = \{x \in \Omega\,;\, \phi(x) = 0\}\,,$$

where

$$\begin{aligned} &\Omega^+ := \{x \in \Omega\,;\, \phi(x) \geq 0\} \quad \text{and} \\ &\Omega^- := \{x \in \Omega\,;\, \phi(x) < 0\} \end{aligned}$$

denote mutually disjoint subdomains of $\Omega$ that indicate the presence or absence of material, respectively.

A structural optimization process can be described by introducing a *pseudo time $t$* that represents the iterations in the optimization process. We can then express the dynamic model as

$$\partial\Omega^+(t) = \{x(t) \in \Omega\,;\, \phi(x(t), t) = 0\}\,.$$

By differentiating both sides of $\phi(x(t), t) = 0$ with respect to time and applying the chain rule, we obtain the first-order *Hamilton-Jacobi* (HJ) equation

$$\frac{\partial \phi}{\partial t}(x, t) + v\, \nabla\phi(x, t) = 0. \tag{3.15}$$

In level-set-based TO, this PDE describes the motion of the material interface due to a design *velocity field* $v = {}^{dx}/_{dt}$. Note that Equation (3.15) holds for each iso-contour and, therefore, for all $x \in \Omega$. As the outward normal to the zero-level contour relates to the gradient of the LSF by $n = {}^{-\nabla\phi}/_{|\nabla\phi|}$, the HJ equation can be simplified as follows [136],

$$\frac{\partial \phi}{\partial t}(x, t) + v_n(x, t)\, |\nabla\phi(x, t)| = 0, \tag{3.16}$$

where $v_n = \langle v, n \rangle$ denotes the normal velocity field and represents the sensitivity of moving the interface in the normal direction $n$ with respect to some form of merit function. For example, in the case of unconstrained compliance minimization, the normal velocity function can be written as $v_n = -\nabla^S u : \mathbb{C} : \nabla^S u$ on the traction-free boundaries of the structure (see [142, 143] for a derivation). To account for design constraints, penalty formulations are typically used for constructing merit functions [142, 144, 145]. In theory, the level-set approach leads to designs with sharp and smooth edges, thus avoiding semi-dense elements like those observed in the SIMP method. In practical applications, however, the LSF is usually mapped to a discretized density distribution on a voxel grid using a Heaviside function. See Algorithm 3 for a pseudo-code implementation of the level-set method.

Equation (3.16) can be discretized in pseudo time using explicit [142, 146], semi-implicit [147, 148] or fully implicit schemes. Due to their algorithmic simplicity, explicit methods, such as an Euler forward scheme, are frequently used. Using an Euler forward scheme produces an update formula that can be employed to progressively optimize a structure using an iterative approach [143],

$$\phi^{i+1} = \phi^i - v_n \, \Delta t \left|\nabla \phi^i\right| , \tag{3.17}$$

where $i$ denotes the current iteration, $\Delta t$ is a discrete time step, and $v_n$ is the discrete velocity field normal to the boundary.

Although the iterative updating scheme in Equation (3.17) tends to converge to smooth topologies, it may require many iterations. Nevertheless, it has been extensively investigated [138, 142, 146, 149] and applied to a broad range of design problems, including structural problems [139], vibration problems [146, 150] and thermal problems [151], among others.

**Algorithm 3** Level-set method via a Hamilton-Jacobi equation

```
  Require: $F$, $N$                                      ▷ Forces, #iterations
  Initialize: $\phi^0$                                   ▷ Start with an initial LSF
  for $i = 0, \dots, N-1$ do
      $u$ = pde_solver($F$, $\phi^i$)                    ▷ Solve PDE for current LSF
      $v_n = v_n(u, F, \phi^i)$                          ▷ Calculate normal velocity function
      $\phi^i \leftarrow \phi^i - v_n \, \Delta t \left|\nabla \phi^k\right|$   ▷ Update LSF based on HJ equation
  end for
  $\rho \leftarrow$ cutoff($\phi^N$)        ▷ Obtain densities from LSF using a Heaviside function
return $\rho$                                            ▷ Return final density distribution
```

A significant difference between density and level-set approaches lies in how gradients are computed and how designs are updated. To ensure that the evolution of the LSF is efficiently driven by the shape sensitivities, the gradients should be uniform along the interface [152]. As the HJ equation (3.16) does not maintain uniform spatial gradients of the LSF, re-initialization schemes are often used to prevent the spatial gradients $\nabla\phi$ from becoming too steep or too flat along the interface [136, 138]. Furthermore, since the LSF evolves only along interfaces that exist in the initial design, updating the level-set function via the HJ equation does not allow for the creation of new holes in the material domain. To overcome this limitation, a hole nucleation algorithm called the *bubble technique* [153, 154] was introduced in 1994. The basic idea is to predict the influence of introducing an infinitesimal hole at any point in the design domain and use this information to generate new holes. This influence is commonly determined using *topological derivatives* [155]. The topological derivative $T$ is a measure of how an objective functional $\mathcal{J}$ changes when an infinitesimally small spherical hole is introduced into the structure,

$$T(x) := \lim_{\varepsilon \to 0} \frac{\mathcal{J}(\Omega^+ \setminus B(x,\varepsilon)) - \mathcal{J}(\Omega^+)}{|B(x,\varepsilon)|},$$

where $B(x,\varepsilon)$ is the ball of radius $\varepsilon$ centered at $x \in \Omega^+$. In 2001, Garreau et al. [156] gave specific formulations for the topological derivative of planar linear elasticity equations.

It is also possible to nucleate new holes by adding a source term $\mathcal{R}$ to the HJ equation. This is commonly done in combination with a diffusive term $\mathcal{D}$ that acts as a regularizer, thus preventing rapidly oscillatory level-set fields and interface geometries [76],

$$\frac{\partial \phi}{\partial t}(x,t) + v_n(x,t)\,|\nabla \phi(x,t)| - \mathcal{D}(\phi)(x,t) - \mathcal{R}(\phi)(x,t) = 0.$$

The diffusive term smooths the level-set field using a linear or nonlinear diffusion model. In the linear case, e.g., $\mathcal{D}(\phi) = \operatorname{div}(\nabla \phi)$, one can typically associate a potential with the diffusive term which is minimized in the course of the updating procedure [142, 157, 158].

## 3.3 Conclusion and Discussion

Common for the implementation of the different TO algorithms is that they use an iterative procedure to create a complex mapping from problem-defining characteristics (i.e., supports, loads, and objective function) to an optimized structure. As the iterative process relies on repetitive analysis and design, the PDE for linear elasticity (3.5) must be solved for the intermediate solution at each iteration of these procedures. Especially for larger problems, obtaining this solution becomes computationally expensive, presenting a significant challenge in large-scale topology optimization.

Out of the proposed methods, SIMP is arguably the most common and most widely used approach. This popularity can be attributed to several factors: first, SIMP is a relatively straightforward and intuitive method. It is based on penalizing intermediate densities, making it conceptually simple for engineers and practitioners to understand and implement. This is a significant advantage over level-set methods, which require precise control of the level-set field, particularly the slope near the boundary. Various techniques, such as velocity field smoothing, topological derivatives, and the necessity for re-initialization, underscore the ongoing challenge of effectively managing the level-set field. Furthermore, as most level-set methods employ smoothed Heaviside functions, the crispness is lost when mapping the geometry onto a discretized voxel grid.

Second, SIMP has been widely adopted and extensively studied, leading to the development of efficient implementations in commercial software packages. SIMP has demonstrated success in optimizing a variety of structures across different engineering disciplines. Its performance and reliability in generating manufacturable designs contribute to its continued popularity. Compared to SIMP, particularly the ESO/BESO approaches have been criticized for failing in certain situations pertaining to flaws in only making discrete density updates [108, 115]. Rozvany [108], in a notable critique, highlighted the absence of a solid mathematical foundation for ESO, leading him to advocate for SIMP in TO. A fundamental issue with discrete ESO/BESO methods and their variants is the lack of algorithmic convergence and the challenge of selecting a suitable stopping criterion. Typically, discrete optimization methods face a common problem of being highly sensitive to parameter variations, often resulting in oscillating and non-converging solutions. That makes it particularly difficult to find parameters and update strategies that converge to good designs in stable and efficient manners.

Third, as explored in Chapter 4, SIMP continues to maintain its prominence in the modern era, mainly due to its seamless integration with deep learning methodologies. The gradient-based update steps inherent in SIMP, combined with the voxelized representation of densities, make it a natural fit for neural networks, further accentuating its relevance and adaptability in contemporary engineering optimization practices.

# Chapter 4

# Deep Learning for Topology Optimization

The development of high-performance computing technology not only drives the development of TO but also catalyzes significant progress in various scientific and engineering domains. *Machine learning* (ML), in particular, has harnessed these technological advancements. Up until now, ML has been successfully applied in many mechanics-related fields, including but not limited to predicting mechanical responses [159, 160, 161], solving multi-physics problems [162] and designing meta-materials [163]. Furthermore, the rise of ML – especially *deep learning* (DL) – has played a pivotal role in shaping innovative approaches to TO.

The integration of *neural networks* (NNs) with structural optimization is not a new concept, with its origins dating back to the 1990s [164, 165, 166]. For example, Kodiyalam and Gurumoorthy [166] utilized a two-layer feedforward network to optimize the design of aircraft engine guide and satellite reflector assembly. However, constrained by computing power limitations and the prevalence of traditional ML methods, this combination of NNs and TO did not experience substantial research momentum. Renewed interest in the field emerged in the 2010s, with pivotal works being published by Ulu et al. [167] and Sosnovik and Oseledets [168] in 2014 and 2017, respectively.

Research combining DL and TO can be broadly categorized into three main categories [2]. The first category aims to learn the SIMP solution process by reducing the required number of SIMP iterations or eliminating any classical iterations altogether. The second category accelerates classical TO methods via DL by elevating the PDE solution process, which is SIMP's primary bottleneck. Lastly, an alternative approach is to use NNs to reparameterize TO's density field via an implicit representation, e.g., using a deep image prior. In the upcoming sections, we will analyze all three categories in detail. Subsequently, we will briefly discuss alternative approaches in Section 4.4.

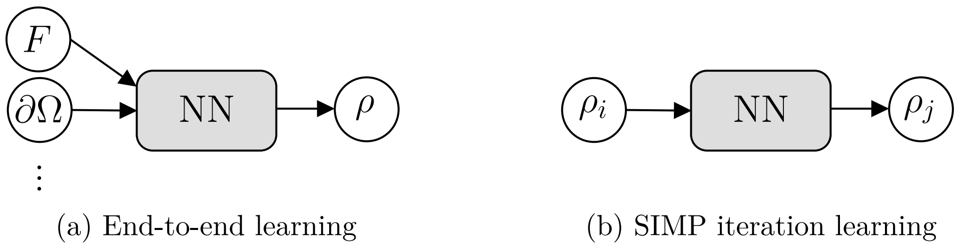


Figure 4.1: Two different approaches for learning the SIMP solution process. (a) The NN predicts an optimized structure for a given problem based on applied forces and boundary conditions. (b) The network learns the density at SIMP iteration $j$ based on the density from a previous iteration $i$, effectively solving a deblurring task.

## 4.1 Learning the SIMP Solution Process

The first serious attempts of TO via DL aimed to reduce the number of SIMP iterations or eliminate any iterative process altogether [167, 168, 169]. In 2014, Ulu et al. [167] introduced a TO method based on ML, incorporating principal component analysis for feature extraction and employing an MLP to map loading configurations to optimized topologies. This work set a standard for ML-based TO. The second pivotal work, conducted by Sosnovik and Oseledets [168] in 2017, pioneered the application of CNNs in TO. They treated the TO problem as an image-to-image regression task, training a U-Net model [32] to map from non-binary intermediate structures at each iteration to the final optimized binary structure.

Both works learn the SIMP solution process via DL but represent two distinct approaches (see Figure 4.1): Sosnovik and Oseledets [168] use DL to reduce the number of SIMP iterations, which we call *SIMP iteration learning*. Ulu et al. [167] go one step further and eliminate all classical iterations. We call this approach *end-to-end learning* or *direct learning*.

Both approaches have different advantages and challenges. SIMP iteration learning relies on the SIMP algorithm and still requires costly PDE solutions. Additionally, the approach operates under the assumption that the density history during the early stages of the optimization procedure adequately determines the nature of the final result. This can be a problematic assumption for problem instances where parts of the structure move during optimization.

On the contrary, end-to-end learning algorithms establish a surrogate model to directly predict an optimal structure based on given design restrictions to achieve iteration-free TO. As pointed out in [2, 3], this leads to faster inference times but requires more training samples. By definition, such models cannot present the full optimization path, which may lead to challenges regarding interpretability and explainability of results. Further, they tend to suffer heavily from generalization to unseen cases. Nonetheless, the direct learning approach is currently one of the most popular applications of DL in TO.

In 2018, Banga et al. [170] transferred the SIMP iteration learning framework to three dimensions. Xue et al. [169] reduced the cost of SIMP iterations by solving the PDE on a coarse grid using a classical solver. They then learn an upscaling to increase the resolution of the solution. This idea has recently been investigated further by [171, 172], who combine the approach with implicit representations (see Section 4.3). Aside from the standard U-Net architecture [32], there have also been experiments with other model types, including *residual neural networks* [173] and *deep belief networks* [174], which employ feature detection to achieve dimensionality reduction in each layer [175].

In 2019, Yu et al. [176] followed up on the end-to-end learning approach to directly predict the final densities without performing any SIMP iterations. They used a *generative adversarial network* (GAN) [177] as an additional post-processing step to increase the resolution of their predictions (see Section 4.4.1 on generative models for TO). Other publications that perform end-to-end learning for TO include [178, 179, 180]. The structural design representation is typically defined by element densities within a regular mesh, similar to the conventional SIMP approach. Some authors base their structural representation on geometrical features inspired by *feature mapping* or *moving morphable components* techniques [181, 182].

While most DL approaches for TO are entirely oblivious to the underlying physics, there is a small number of approaches that have started to include physics-inspired properties into the training process: Banga et al. [170] and Rade et al. [183] augmented the training dataset by including rotations and mirrors of given loads and boundary conditions. In some works, additional inputs related to initial stress or strain [184, 185] and displacements [179, 186] are included to give the network more information on physical properties. To our knowledge, no literature incorporated the underlying physics by modifying the architecture of the DL model itself.

In our publication *SELTO: Sample-Efficient Learned Topology Optimization*, we employ an end-to-end learning approach and demonstrate that we can dramatically improve *sample efficiency*, i.e., the model's performance when trained on only few training samples. Further, we facilitate geometric reasoning by restricting the hypothesis space to group equivariant models (see Section 4.5), drastically improving the prediction quality.

## 4.2 Learning a PDE Solver

The recent advancements of DL in computer vision [187, 188], natural language processing [189, 190] and ecology [191] have caused a surge of interest in applying these techniques to scientific problems. The field of *scientific machine learning*, which combines the approximation power of data-driven ML methodologies with traditional modeling techniques based on PDEs, sets out to use ML tools for accelerating scientific discovery. In the field of TO, the main bottleneck in classical iterative approaches like SIMP is that each iteration requires solving the PDE for linear elasticity. Due to this computational challenge, TO at high resolutions can take hours, even days [192, 193, 194]. This inspired researchers to apply DL-based TO methods to reduce or eliminate the need to solve PDEs via classical methods.

Scientific ML can roughly be categorized into three main areas: (i) learning a PDE solution, (ii) learning a PDE solution operator, and (iii) PDE discovery.

Learning a PDE solution commonly entails incorporating the PDE directly into the training process as constraints [195, 196, 197]. These constraints often stem from prior knowledge and are typically incorporated into the loss function used during training. Notable methods following this paradigm include *physics-informed neural networks* (PINNs) [196, 198, 199, 200], the *deep Ritz method* (DRM) [201], and the *deep Galerkin method* (DGM) [202]. PINNs have recently garnered significant attention for their mesh-independency (see Section 4.3 on neural reparameterization and implicit representations). Consequently, training does not involve discretized grids, as traditional numerical methods do [199]. However, any alterations to physical parameters, coefficients, or boundary conditions necessitate retraining, which is a notable drawback when applied to TO.

On the other hand, operator learning via *neural operators* (NOs) [203, 204] aims to approximate an unknown parameter-to-state operator, which often takes the form of the solution operator associated with a PDE. NOs serve as approximators for infinite dimensional parameter-to-state mappings. Two prominent instances of this approach are the *deep operator network* (DeepONet) [205] and the *Fourier neural operator* (FNO) [206], which we discuss in Sections 4.2.1.1 and 4.2.1.2, respectively. Both methods have brought about significant advancements in the field of scientific machine learning.

Finally, PDE discovery aims to solve the inverse problem of identifying the parameters of a PDE from data. Instead of building models from physical laws, PDE discovery aims to discover unknown physics and the corresponding equations directly from limited observation data [207]. Early prominent examples include *PDE-Find* [208] and *PDE-Net* [209], which involve specifying a collection of candidate *library terms* $f_1, \dots, f_m$, each of which is a function of the PDE solution $u$ and its spatial derivatives. The goal is to find a sparse vector $\xi$ that fulfills

$$u_t = \xi_1 f_1(u) + \dots + \xi_m f_m(u),$$

where $u_t$ denotes the time derivative of $u$. In PDE-Find and PDE-Net, derivatives are numerically approximated using finite differences stencils, requiring the data to lie on a uniform spatio-temporal grid. A significant breakthrough came in 2021 with the introduction of *DeepMoD* [210]. DeepMoD uses automatic differentiation [211] and an implicit representation to calculate the partial derivatives of the NN, from which it evaluates the library terms. Notably, the derivatives can be evaluated anywhere in the problem domain, not just at the points where observational data is available. Today, there exists a large number of PDE discovery algorithms [207, 212, 213, 214, 215, 216, 217], most of which still involve finding a sparse linear combination of the differential terms within the specified library. Therefore, sparse regression is of critical importance to PDE discovery [212]. In this thesis, we assume the structure of the PDE is to be known, thereby excluding any further exploration of PDE discovery at this point.

In the following, we start with a general introduction of NOs in Section 4.2.1, along with a definition of DeepONets and FNOs in Sections 4.2.1.1 and 4.2.1.2, respectively. After that, in Section 4.2.2, we discuss possible applications of learned PDE solvers to TO and reason why we consider NOs a promising approach.

### 4.2.1 Neural Operators

In this section, we review the most common NO [203, 204] architectures used in the literature, namely *deep operator networks* (DeepONets) [218] and *Fourier neural operators* (FNOs) [206]. Operator learning has been successfully applied to many PDEs from different fields, including fluid dynamics [219, 220], continuum mechanics [221], astrophysics [222], quantum mechanics [206] and weather forecasting [223, 224]. For an extensive survey of different NO architectures and their applications, see [225, 226].

NOs [203] aim to learn an infinite-dimensional mapping from a finite dataset of input-output pairs. Let $d, n, m \in \mathbb{N}$ and $\Omega \subset \mathbb{R}^d$ be bounded and closed. We formulate the learning problem on real-valued Banach spaces $\mathcal{A} := \mathcal{A}(\Omega, \mathbb{R}^n)$ and $\mathcal{U} := \mathcal{U}(\Omega, \mathbb{R}^m)$, with an operator $\mathcal{G}^\dagger : \mathcal{A} \to \mathcal{U}$ mapping between these two spaces. Given an i.i.d. sequence of data points $\{(a_i, u_i)\}_{i=1}^N \sim (\mathcal{A}, \mathcal{U})$ with $u_i = \mathcal{G}^\dagger(a_i)$, the objective is to approximate $\mathcal{G}^\dagger$ with a parameterized mapping $\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}$ with learnable parameters $\theta$.

NOs employ an iterative approach using a sequence of nonlinear operators $\mathcal{G}_i$ that generalize the traditional dense layer given by Equation (2.1). The operators comprise linear integral operations succeeded by point-wise nonlinearities. Specifically, for an input function $v_i$ at the $i$-th layer, the computation of $\mathcal{G}_i : v_i \mapsto v_{i+1}$ is defined by

$$\mathcal{G}_i v_i(x) := \sigma\left(\mathcal{K}_\theta(v_i)(x) + W v_i(x)\right),$$

where $\mathcal{K}_\theta$ is a linear operator parameterized by $\theta$, $W$ represents a linear transformation and $\sigma$ is an element-wise nonlinear activation function. We choose $x \in \Omega_i$, where

$\Omega_i \subset \mathbb{R}^d$ is a bounded domain for all $i$ with the initial domain given by the whole design space, $\Omega_0 = \Omega$.

It is common to choose $\mathcal{K}_\theta$ to be a kernel integral transformation,

$$\mathcal{K}_\theta(v_i)(x) := \int \kappa_\theta(x, y) v_i(y) \, \mathrm{d}y, \tag{4.1}$$

with $\kappa_\theta$ being a NN parameterized by $\theta$. Here, $\kappa_\theta$ serves as a learnable kernel function. In this general form, approximating the kernels or evaluating the integral operators could be computationally expensive. Hence, several NO architectures have been proposed to overcome these challenges, such as DeepONets [218] and FNOs [206].

The main goal of employing NOs is to speed up the PDE solution process. This is particularly relevant for complex systems that are computationally challenging to simulate with traditional numerical PDE solvers. For instance, this situation may arise for high-dimensional PDEs or fluid dynamics applications such as modeling turbulent flows, which require very fine discretizations. Moreover, specific problems in engineering require many evaluations of the solution operator. In these cases, a fast but less accurate solver provided by operator learning may prove helpful in forecasting or optimization.

NOs do not rely on a fixed discretization as they are mesh-free and parameterized by a NN that can be evaluated at any point. This property makes them suitable for solving PDEs on irregular domains or transferring the model to other spatial resolutions [203]. Alternatively, NOs can be employed for parameter identification. Once the solution operator has been approximated, it can be exploited in an inverse problem framework to recover unknown parameters of the PDE, which may be computationally challenging to perform with existing numerical PDE solvers.

#### 4.2.1.1 Deep Operator Networks

*Deep operator networks* (DeepONets) [205, 218] are a promising class of NOs for learning nonlinear operators and capturing the inherent relationships between input and output functions. They extend the capabilities of traditional DL techniques by leveraging the expressive power of the NNs to approximate operators in differential equations, integral equations, or, more broadly, any functional mappings from one function space to another. A key theoretical motivation for DeepONets are universal operator approximation theorems [218, 227]. They can be seen as infinite dimensional generalizations of standard universal approximation theorems for NNs [228, 229], which guarantee that a sufficiently wide NN can approximate any continuous function to any accuracy. Further, since the introduction of DeepONets, several research works focused on deriving error bounds for the approximation of nonlinear operators in various settings, such as learning the solution operator associated with Burger's equation or the advection-diffusion equation [230].

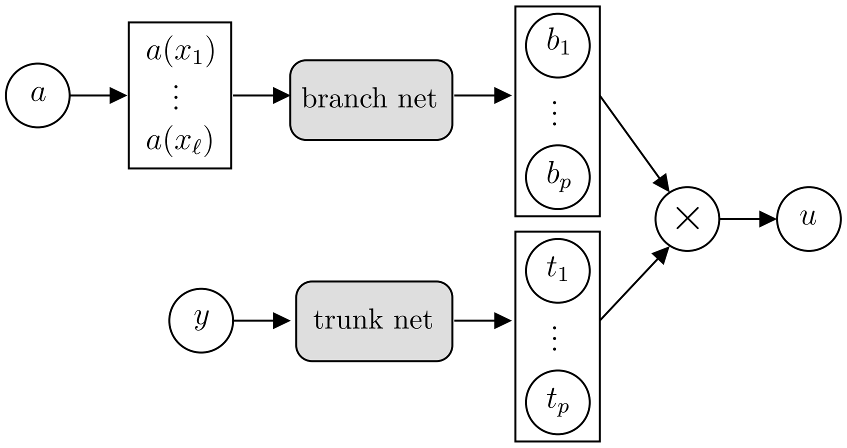


Figure 4.2: Illustration of a standard DeepONet model [205, 218].

A DeepONet is a two-part network consisting of a *branch network* and a *trunk network* (see Figure 4.2). The branch network encodes the operator's input function $a$ into compact, fixed-size latent vectors $b_1(a(x_1),\dots,a(x_\ell)),\dots,b_p(a(x_1),\dots,a(x_\ell))$, where $\{x_i\}_{i=1}^{\ell}$ are the *sensor points* at which the input function $a$ is evaluated. The trunk network decodes these latent vectors to produce an approximation of the true solution $u$ at the location $y \in \Omega$ as

$$u(y) \approx \mathcal{G}_\theta(a)(y) = \sum_{k=1}^{p} b_k(a(x_1),\dots,a(x_\ell))\, t_k(y).$$

The defining feature of DeepONets is their ability to handle functional input and output, thus enabling them to learn a wide array of mathematical operators effectively. It is worth mentioning that the branch and trunk networks can have arbitrary and distinct NN architectures tailored for different purposes. Moreover, while the interplay of the branch and trunk networks is crucial, the output of a DeepONet does not necessarily depend on the specific input points but rather on global properties of the entire input function, which makes it suitable for learning mesh-independent operator maps.

One reason behind the performance of DeepONets might be their connection with the low-rank approximation of operators and the singular value decomposition (SVD). Herein, one can view the trunk network as learning a basis (or, more accurately, a generating system) of functions $\{t_k\}_{k=1}^{p}$ that are used to approximate the operator. The branch network expresses the output function in this basis by learning the coefficients $\{b_k\}_{k=1}^{p}$. Moreover, the branch network can be seen as a model reduction method, which encodes the input function into a compact representation, thus reducing the problem's dimensionality to $p$, where $p$ is the number of branch networks. Additionally, several architectures, namely the POD-DeepONet [231] and SVD-DeepONet [232] have been proposed to strengthen the connections between

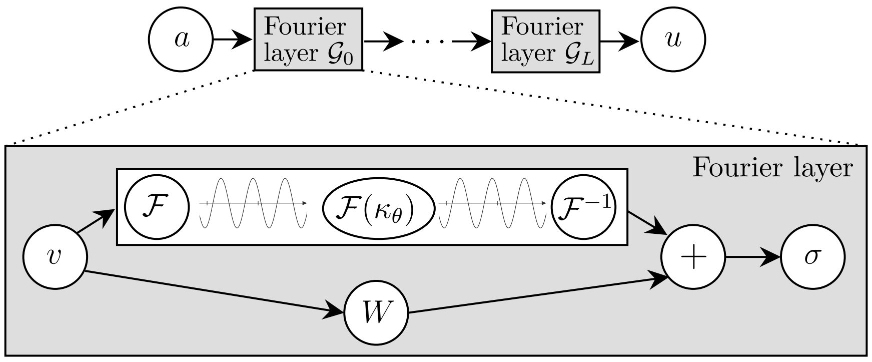

$a$
Fourier layer $\mathcal{G}_0$
Fourier layer $\mathcal{G}_L$
$u$
Fourier layer
$\mathcal{F}$
$\mathcal{F}(\kappa_\theta)$
$\mathcal{F}^{-1}$
$v$
$+$
$\sigma$
$W$

Figure 4.3: Illustration of a standard FNO model [203, 206].

DeepONets and the SVD of the operator.

A desirable property of NNs is discretization invariance, i.e., the model can act on any discretization of the source term and be evaluated at any point of the domain [203]. This property is crucial for the model's generalization to unseen data and the transferability of the model to other spatial resolutions. While DeepONets can be evaluated at any location of the output domain, they are not discretization invariant in their original formulation, as the branch network is evaluated at specific points of the input domain. However, this can for instance be resolved by fixing the set of points on which the input function is evaluated independently of its discretization [203] or by sampling the input functions at local spatial averages [233].

DeepONets have been successfully applied and adapted to a wide range of problems, including predicting cracks in fracture mechanics [234] and predicting linear instabilities in high-speed compressible flows with boundary layers [235].

#### 4.2.1.2 Fourier Neural Operators

*Fourier Neural Operators* (FNOs) [203, 206] are a class of NOs that are motivated by Fourier spectral methods. They have demonstrated significant success in learning and predicting solutions to various PDEs, particularly those with periodic boundary conditions. This capability renders FNOs an invaluable tool in areas where PDEs play a central role, such as fluid dynamics, quantum mechanics, and electromagnetism.

The main idea behind FNOs is to choose the kernels $\kappa_\theta$ in Equation (4.1) as translation-equivariant kernels that satisfy $\kappa_\theta(x, y) = \kappa_\theta(x - y)$. For $x \in \Omega_i$ we then obtain

$$\mathcal{K}_\theta(v_i)(x) = (\kappa_\theta * v_i)(x), \tag{4.2}$$

which in many PDE-related applications is a natural choice from the perspective of fundamental solutions [236]. FNOs efficiently parameterize the kernel function $\kappa_\theta$

by exploiting the Fourier space representation of $v_i$. From Equation (4.2) and the convolution theorem we derive

$$\mathcal{K}_\theta(v_i)(x) = \mathcal{F}^{-1}\left(\mathcal{F}(\kappa_\theta) \cdot \mathcal{F}(v_i)\right)(x),$$

where $\mathcal{F}$ denotes the Fourier transformation, which can be efficiently calculated using the Fast Fourier Transform (FFT). For an illustration of an FNO layer, see Figure 4.3.

Assuming periodicity of $\kappa_\theta$, a Fourier series expansion yields discrete Fourier modes. By truncating the series to a maximum of $k_{\max}$ modes, $\mathcal{F}(\kappa_\theta)$ is directly parameterized as a learnable weight tensor with $k_{\max}$ channels. This truncation acts as a low-pass filter, leading to relatively smooth outputs [237]. If the input domain is discretized uniformly with $\ell$ sensor points, and the vector $\mathcal{F}(\kappa_\theta)$ contains at most $k_{\max} < \ell$ modes, the convolution can be executed with quasi-linear complexity in $\mathcal{O}(\ell \log \ell)$ operations using the FFT [62]. This represents a noteworthy enhancement compared to the $\mathcal{O}(\ell^2)$ operations needed to compute the integral in Equation (4.2) using a quadrature rule. In practical scenarios, by limiting the number of Fourier modes to $k_{\max} \ll \ell$, one can often achieve a satisfactory approximation accuracy, especially when the input and output functions are smooth, leading to rapid decay of their coefficients in the Fourier basis.

While FNO is fast and accurate, it has limitations on the input format and the problem domain. Since FNO is implemented using the FFT, it can only easily be applied on rectangular domains with uniform meshes. In the case of irregular domain shapes, it is possible to define embeddings into larger rectangular domains [231]. However, this is less efficient and wasteful, especially for highly irregular geometries. Similarly, if the input data is in the form of non-uniform meshes such as triangular meshes, several works have been proposed to extend the FNO architecture to more general domains [231, 238, 239, 240]. This can, however, cause significant interpolation errors, especially for nonlinear PDEs.

Similarly to DeepONets, FNOs are universal approximators, in the sense that they are dense in the space of continuous operators [203, 241, 242]. However, despite being universal approximators, NOs could theoretically require a huge number of parameters to approximate a given operator to a prescribed accuracy $\varepsilon > 0$. As an example, [243] showed that the size of NOs must grow exponentially fast as $\varepsilon$ decreases to approximate any operator between functions whose Fourier coefficients decay only at a logarithmic rate. Fortunately, these pessimistic lower bounds are not observed in practice when learning solution operators associated with PDEs [242]. In practical comparisons FNOs often show a better cost-accuracy trade-off than DeepONets regarding the required number of training samples and network size [244].

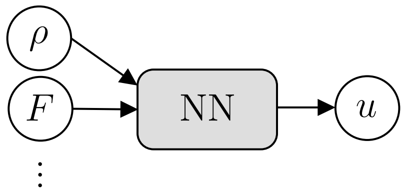


Figure 4.4: Example of a learned PDE solver for TO. The NN takes the current density and relevant problem specifications as input and generates the corresponding displacements as output.

### 4.2.2 Application to Topology Optimization

The primary computational bottleneck in classical TO lies in the repeated solution of the PDE for linear elasticity. There has been a growing interest in leveraging learned surrogate models for traditional numerical solvers to address this challenge. In practical terms, the objective is usually not to enhance prediction quality but rather to speed up the computational process. This results in a trade-off between speed and accuracy that requires careful balancing.

Learned surrogate models have been widely adopted in structural mechanics for predicting mechanical responses [245, 246, 247]. Recent efforts have been made to incorporate the PDE for linear elasticity as a constraint within the loss function, akin to the methodology used in PINNs [248, 249, 250, 251]. However, a common limitation in these approaches is the need for retraining or transfer learning when modifying input parameters, such as loads, boundary conditions, and material properties. Recent publications have addressed this limitation using DeepONets [252, 253] and FNOs [238].

The aforementioned literature solves the PDE for linear elasticity without a direct connection to TO. Notable exceptions are [254, 255], who employ PDE learning as a direct part of the TO pipeline using PINNs and DeepONets, respectively. Both papers employ a dual-model approach, with a primary network calculating a material density at an arbitrary spatial coordinate and a secondary network determining the stresses or displacements at that point. Hence, the authors combine learning the PDE solver with an end-to-end learning approach (see Section 4.1) and neural reparameterization (see Section 4.3). Similarly, some authors [256, 257, 258] employ a more direct approach by replacing the sensitivity and objective computation with a NN.

As explored in Sections 4.1 and 4.6, end-to-end learning approaches often face challenges due to the requirement for extensive training data and limitations in generalization. An alternative and potentially more promising strategy involves substituting SIMP's traditional PDE solver with a NN, eliminating its primary computational bottleneck and contributing to faster optimization (see Figure 4.4). It is important to note that the unique requirements of SIMP influence our choice of NO models. SIMP operates on a fixed voxel grid, making convolution and FNO-based models a natural fit. This rationale explains why we do not further investigate using DeepONets for SIMP, as these models explicitly avoid a fixed-grid voxel representation of the design domain.

Expanding the training dataset to encompass intermediate SIMP densities provides the advantage of acquiring a more extensive collection of training samples with diverse characteristics. Consequently, data generation is often more cost-effective than for direct design models, presenting an opportunity to naturally capture greater input diversity. However, incorporating SIMP into the DL pipeline introduces an additional challenge. Not only must the solution of the PDE be accurately determined, but also the sensitivities that guide SIMP's iterative density updating scheme. Researchers have tackled this challenge in recent literature by training a secondary network that produces sensitivities for the primary network, as demonstrated by Qian and Ye [259] and Lunz et al. [260]. Nevertheless, exploring methods for obtaining accurate gradients without employing a secondary network seems more desirable and less cost-intensive.

In our publication *Equivariant Neural Operators for Gradient-Consistent Topology Optimization*, we present the innovation of applying NOs to SIMP. We introduce and compare different FNO architectures as surrogates for SIMP's PDE solver and show that enforcing group equivariance (see Section 4.5) drastically improves the quality of predictions. We employ a novel loss summand that enforces correct sensitivities for the SIMP updating scheme without using a dual-model approach. Further, we demonstrate that without this loss summand, learned PDE solvers fail entirely when incorporated into an iterative gradient-based optimization routine like SIMP. This presents an important finding since it demonstrates that merely learning an accurate PDE surrogate model is insufficient in real-world applications involving gradient-based optimization.

## 4.3 Neural Reparameterization

The goal of *reparameterization* for TO is to represent a structure via an *implicit representation.* For classical reparameterization methods like the level-set method (see Section 3.2.3) and topology description functions (see Section 4.3.2), the new representation often requires fewer design variables, thereby reducing the computational load of the optimization procedure. Today, reparameterization is commonly achieved using NNs in what is called *neural reparameterization*, most notably used in *neural radiance fields* (NeRFs) [261] and *deep image prior* (DIP) [262, 263] approaches. The typical approach involves constructing a network as a direct surrogate for the optimization process, where training the network is equivalent to solving the

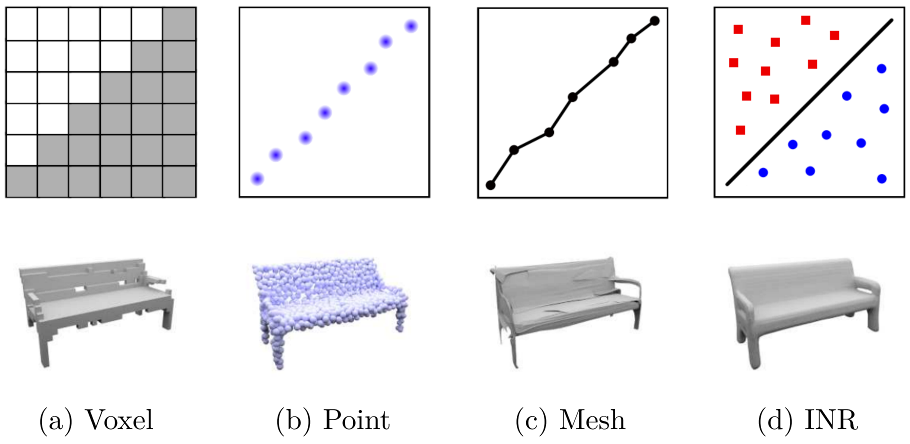


Figure 4.5: 3D representations are categorized according to their discretization of the output space. Image from [264].

optimization problem. This is realized by interpreting the network parameters as the new design variables of the problem.

In the following, we introduce implicit neural representations in Section 4.3.1, followed by the application to TO in Section 4.3.2.

### 4.3.1 Implicit Neural Representations

In contrast to conventional grid-based approaches, *implicit neural representations* (INRs) [265, 266, 267, 268, 269] (also referred to as *neural fields* or *coordinate-based NNs*) offer a continuous and differentiable framework for signal recovery [270]. An INR is a NN representing a function $x \mapsto y$ that maps a spacial coordinate $x$ to a quantity $y$. INRs are a novel research field that utilizes NNs to create approximate representations of surfaces or shapes. INRs have primarily been employed for reconstructing continuous surfaces from unstructured and incomplete 3D point clouds. They have rapidly gained prominence in diverse domains, such as medical imaging [271], computer graphics [261, 272], and robotics [273] and offer a unique approach to encoding complex data. INRs have seen widespread success in representing and generating a variety of signals, including shapes [264, 265, 274], scenes [261], images [275] and videos [269].

Based on their output format, NN-based representation methods can be broadly classified into *discrete-form* and *continuous-form* approaches. Before the emergence of INRs, shape and surface representations were predominantly in discrete forms, exemplified by point-based [276, 277], voxel-based [278], and mesh-based [279, 280] methods. In contrast to representation methods utilizing discrete forms, INRs dispense with the need for voxel or mesh discretizations. Further, compared to discrete

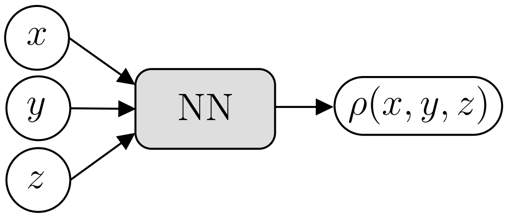


Figure 4.6: Neural reparameterization for TO. The NN receives a 3D coordinate pair $x, y, z$ as input and outputs the corresponding density at that specific location. Therefore, the NN serves as an INR for the density field.

representations, INRs are significantly more memory-efficient while providing higher fidelity, continuity, and analytic differentiability. Implicit representations effectively embody advantageous properties of NNs, including compactness and smoothness [281]. These attributes theoretically enable achieving continuous infinite-resolution function representations in a mesh-free manner. Outcomes of different data representation methods based on NNs are shown in Figure 4.5.

There have recently been substantial efforts to improve INRs. For instance, it has been found that the traditionally popular ReLU activation function fails to represent features in the high-frequency domain well [282], leading to several improvements in network architectures. This includes sinusoidal activation functions introduced in SIREN [269], wavelet transforms like WIRE [283], and a mixture of sinusoidal and exponential activations in TRIDENT [284].

## 4.3.2 Application to Topology Optimization

Researchers have explored new approaches involving DL to overcome the limitations associated with data-driven TO. One such paradigm involves neural reparameterization via INRs, which can be applied in various ways.

Typically, the NN takes the input coordinates $(x, y, z)$ of a point within the design domain and generates the density value $\rho(x, y, z)$ at that specific location (see Figure 4.6). Alternatively, an INR could output the value of the *signed distance function* (SDF) for the same point. The SDF defines the distance from a given spatial point $x$ to the nearest surface, with the sign indicating whether the point is inside (negative) or outside (positive) of the boundary surface [265]. Consequently, a surface can be implicitly represented by the zero cut-off level of the SDF. Since the INR can be assessed at any point, the derivation of the density distribution is completely mesh-independent. This characteristic leads to the designation of neural reparameterization approaches as being of *infinite resolution*.

While neural reparameterization via SDFs shares similarities with the level-set method introduced in Section 3.2.3, it should be noted that INRs do not rely on the solution of the HJ equation (3.16). See [285, 286] for an explanation of similarities and differences between INRs and traditional level-set methods.

In 2004, De Ruiter and Van Keulen proposed a TO method utilizing *topology description functions* (TDFs) [285]. The core idea involves a function $f : \Omega \to \mathbb{R}$, which is defined by a set of basis functions $h_i$,

$$f(x) = \sum_{i=1}^{N} h_i(x).$$

The parameters affecting the properties of this set of basis functions are the design variables for TO. For instance, when employing a Gaussian probability distribution for constructing the TDF, the design variables are the height and width of each kernel function [285]. Subsequently, the assignment of material for a given point $x$ hinges on the application of a Heaviside function with a predefined cut-off level $c \in \mathbb{R}$; specifically, if $f(x) > c$, the point $x$ is designated as solid. The TDF approach has recently been developed further by Zhang et al. [287], who employ a NN with periodic activation functions for $f$, similar to the SIREN network proposed by Sitzmann et al. [269].

In 2019, Hoyer et al. [288] introduced a physics-driven neural reparameterization framework using a DIP [262, 263] for TO. While Hoyer et al. [288] map latent vectors to discrete grid densities, Chandrasekhar and Suresh [289] use MLPs to learn continuous INRs that map spatial locations to density values. Deng and To [290] present a reparameterization approach similar to that of Chandrasekhar and Suresh [289] but with an increased focus on enabling the representation of detailed 3D geometries. Further, Zhang et al. [2] and Jeong et al. [291] incorporate physics information into the loss function based on PINNs [196]. In 2024, Berzins et al. [292] introduced *geometry-informed NNs*, which learn an INR for TO solely based on optimization constraints and without the need for training data, akin to PINNs. Notably, the authors introduced a novel differentiable loss term that promotes the connectedness of structures, drawing inspiration from Morse theory [293]. Other recent examples of continuous INRs applied to TO include [294, 295].

It should be noted that, despite their mesh-independent nature, physical response analysis of the structures generated by neural reparameterization still requires the solution of the PDE for linear elasticity. If a classical solver is used, like FDM or FEM, this imposes a fixed grid discretization of the domain. However, unlike traditional grid-based TO, this mesh discretization is only used to solve the PDE and is completely decoupled from the design variables of the optimization problem.

## 4.4 Other Approaches

In this section, we investigate other approaches at the intersection of DL and TO that do not fit into any of the categories discussed above. This includes generative models and approaches for reinforcement learning. For an in-depth review of generative models in engineering design, see [296].

### 4.4.1 Generative Models

The network architectures discussed in this chapter are mainly geared towards *discriminative* supervised training for tasks like regression or classification with known output features. In contrast, *generative* models focus on understanding an underlying data distribution. The most straightforward application of generative models in computational mechanics is generating new data based on existing examples.

The *variational autoencoder* (VAE) [297] is an example of a generative model class, which seeks to efficiently represent data by compressing it into a latent vector and then decoding it back to its original format. Although it shares an encoder-decoder structure with the standard U-Net model [32], the VAE's primary objective is effective dimensionality reduction. Trained VAEs enable the generation of new data instances by sampling in the latent space and applying the decoding process [298]. Hence, the latent vector operates as the new design parameter [299, 300]. One major drawback of VAEs is that they often yield blurry samples when generating new data due to their learned average data representation in the latent space.

*Generative adversarial networks* (GANs) [177] take a different approach, pairing a generator network with a discriminator in a min-max game. The generator strives to enhance its ability to create realistic data samples, mimicking the available training data, while the discriminator learns to distinguish between generated and real input samples. GANs can be applied to generate structures optimized for specific objectives, with the generator crafting plausible layouts and the discriminator flagging inappropriate ones. The generator can then be used within a framework for diversifying given design options [176, 184, 301, 302, 303].

Other noteworthy generative models include *normalizing flows* [304, 305] and *diffusion models* [306]. While a small number of publications exist on diffusion models for TO [307, 308], to our knowledge, no publications address normalizing flows in the context of TO.

### 4.4.2 Reinforcement Learning

*Reinforcement Learning* (RL) [309] is a strategy for discovering optimal policies for selecting a sequence of actions that guide a system's evolution from an initial state to a predefined goal. The learning process employs a reward-punishment framework, where the quality of an action is assessed through a designated loss function. Unlike unsupervised learning, RL necessitates the definition of possible system states and actions in advance.

RL is particularly valuable in two situations. First, it may be helpful in cases where differentiable physics is not feasible, as, for example, in crash simulations [310]. Second, it is a natural fit for optimization tasks that can be conceptualized as Markov decision processes, such as binary optimization of trusses using evolutionary strategies [311] (see Section 3.2.2 on the ESO method). In this context, the system states represent truss structures composed of voxel elements. Possible actions involve removing specific voxel elements, and the reward or punishment is determined by whether the chosen action results in a constraint violation. The objective is to iteratively eliminate structural elements to achieve an optimized structure that adheres to constraints. The trained model guides the selection of structural elements to remove at each iteration [3]. This learning strategy can also be employed to explore the design space by choosing different parameter settings for SIMP or ESO [312, 313].

Currently, RL in TO faces significant challenges due to the necessity of addressing large-scale combinatorial optimization problems. In the presence of differentiable solvers, gradient-based methods are still state-of-the-art [81] and generally preferred over RL methods [314].

## 4.5 Equivariant Neural Networks

An enduring challenge in many DL methods for TO lies in effectively incorporating physical information into the model pipeline. A viable strategy to infuse DL models with enhanced physical insight is to impose global properties that align with the expectations of a physically accurate predictor. Specifically, a correct model should demonstrate behavior such that reflecting or rotating the TO problem results in corresponding reflections or rotations in the solution, a property referred to as *equivariance*. Unfortunately, standard NNs do not inherently exhibit this type of behavior. The direct incorporation of such knowledge into the model pipeline can significantly improve the learning process by freeing up capacity for other factors of variation [315]. This section presents a mathematical definition of equivariance and subsequently explores its integration with NNs.

Equivariance is the property of a function or operator to commute with the actions of a symmetry group that acts on both its domain and range. For a given transformation group $G$ we call an operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ *(G-)equivariant* if and only if

$$\mathcal{G}(T_g^{\mathcal{A}}[a]) = T_g^{\mathcal{U}}[\mathcal{G}(a)] \qquad \forall g \in G,\, a \in \mathcal{A}, \tag{4.3}$$

where $T_g^{\mathcal{A}}$ and $T_g^{\mathcal{U}}$ denote linear *group actions* in the corresponding spaces $\mathcal{A}$ and $\mathcal{U}$ [39]. In simpler terms, when transforming an input function $a \in \mathcal{A}$ by a group transformation $g \in G$ and passing it through $\mathcal{G}$, the result should be the same as first applying $\mathcal{G}$ to $a$ and then transforming the output $\mathcal{G}(a)$. Convolutional operators are inherently equivariant with respect to translations. By appropriately choosing $G$, we can additionally enforce equivariance with respect to any desired group transformation.

As $G$ is a group, it includes an identity transformation and a unique inverse $g^{-1}$ for each $g \in G$. Thus, we can restate the initial equivariance condition (4.3) as

$$\mathcal{G}(a) = T_{g^{-1}}^{\mathcal{U}}\left[\mathcal{G}\left(T_g^{\mathcal{A}}[a]\right)\right].$$

This reformulation facilitates the implementation of equivariance through a technique known as *group averaging* [316, 317]. Based on group averaging, we can define an *equivariance wrapper* $\mathcal{G}^G$ as

$$\mathcal{G}^G(a) := \frac{1}{|G|}\sum_{g\in G} T_{g^{-1}}^{\mathcal{U}}\left[\mathcal{G}(T_g^{\mathcal{A}}[a])\right].$$

$\mathcal{G}^G$ is equivariant with respect to the transformation group $G$ since it satisfies

$$\begin{aligned}
T_{h^{-1}}^{\mathcal{U}}\left[\mathcal{G}^G\left(T_h^{\mathcal{A}}[a]\right)\right] &= \frac{1}{|G|}\sum_{g\in G} T_{gh^{-1}}^{\mathcal{U}}\left[\mathcal{G}\left(T_{gh}^{\mathcal{A}}[a]\right)\right] \\
&= \frac{1}{|G|}\sum_{gh\in G} T_{g^{-1}}^{\mathcal{U}}\left[\mathcal{G}\left(T_g^{\mathcal{A}}[a]\right)\right] \\
&= \frac{1}{|G|}\sum_{g\in G} T_{g^{-1}}^{\mathcal{U}}\left[\mathcal{G}\left(T_g^{\mathcal{A}}[a]\right)\right] \\
&= \mathcal{G}^G(a)
\end{aligned}$$

for all $g, h \in G$.

The prevalent approach for enforcing group equivariance in DL involves ensuring invariance of the network's convolutional filters and activation functions [39]. However, employing group averaging offers several advantages. It is easy to implement, and its plug-and-play nature allows seamless application to any finite transformation group $G$ and any model $f$. Notably, this flexibility permits direct comparisons between non-equivariant models and their equivariant counterparts.

For our applications, natural choices of transformation groups are the dihedral symmetry group $D_4$ and the octahedral symmetry group $O_h$, describing combinations of 90° rotations and reflections in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. Thus, the equivariance wrapper applies 8 transformations in the $D_4$ and 48 in the $O_h$ case. Notably, group actions on vector fields impact spatial locations and vector directions, making equivariance wrappers particularly suitable and flexible.

Enforcing group equivariance via an equivariance wrapper differs fundamentally from data augmentation, where the dataset is augmented with rotations and reflections. While augmentation averages over the objective, the wrapper averages over the model. This distinction endows the wrapper with several advantages, particularly that the model does not need to learn equivariance since it is hard-coded. Consequently, the wrapper is not only approximately equivariant but exactly so, even for out-of-distribution inputs that differ significantly from the training data.

Since 2023, only a handful of publications employ equivariant DL for PDE learning [318, 319, 320, 321]. To our knowledge, we are the first to apply equivariant networks to TO. In both our publications *SELTO: Sample-Efficient Learned Topology Optimization* and *Equivariant Neural Operators for Gradient-Consistent Topology Optimization*, we were able to show that by enforcing group equivariance, we can drastically improve the sample efficiency of our models, i.e., reduce the required number of training samples.

## 4.6 Conclusion and Discussion

In the past few years, there has been a growing trend in publications utilizing DL frameworks to reduce the computational costs of TO. Many of these proposed frameworks draw inspiration from the analogy between element-based material distributions and images, building on advancements in DL for pattern recognition and image analysis. Despite the increased attention, substantial progress remains elusive. The existing body of literature reveals various dead ends, where opaque presentations of results exaggerate the potential of model architectures, setting unrealistic expectations. Some works treat NNs as enigmatic black boxes with seemingly superhuman capabilities, disregarding well-established limitations in the field of DL [3]. The notion of achieving iteration-free end-to-end TO through DL is particularly prominent but becomes problematic when models are oblivious to the underlying physics. Moreover, the cost associated with producing high-resolution training samples and the increased computational complexity linked to meshes at higher resolutions are determining factors influencing why most studies limit their scope to 2D low-resolution meshes [180, 322, 323]. This contrasts with classical state-of-the-art TO methods employing 3D meshes with up to two billion voxel elements [194].

As an additional hindrance, many papers in the field do not present performance metrics that allow for a fair and quantifiable comparison of results. Various reasons contribute to the inadequacy of the presented results for assessment. First, many studies showcase results for only a select few problems from the test set without showing metric scores averaged over all test samples. Second, the choice of performance metrics often focuses solely on voxel-wise density errors, neglecting to assess the actual structural performance of the obtained designs. Finally, the comparison to conventional benchmarks is often skewed. Many works compare gray-scale results obtained by their models to binary structures obtained by SIMP with a large filter radius. This comparison is not representative, as compliance significantly worsens when designs are thresholded to a solid-void design (see [3] for further explanations). Moreover, a large filter radius results in structures with fewer fine features, potentially impacting the structural performance of the design.

Due to a potentially high time investment required for training and the necessity of sufficient training data, employing DL for TO is particularly advantageous in two main scenarios. First, it proves beneficial when dealing with numerous similar components that require individual optimization, as seen in aerospace engineering, where optimizing thousands of mounting brackets for a single space rocket is essential. In such instances, the significance of sample-efficient models cannot be overstated, as they reduce the required number of training samples. Second, DL becomes advantageous when the training process can be delegated to periods outside regular working hours. Despite potentially lengthy training times, models can be trained without constant supervision, such as over the weekend. This offers a more efficient alternative for engineering companies compared to the intermittent waiting periods inherent in classical SIMP methodologies during engineers' regular working hours.

Finally, another notable shortcoming in the field stems from an insufficient research infrastructure. This is particularly based on two crucial aspects:

1. The absence of publicly available 3D TO datasets. This is problematic for several reasons. First, it is hard to reproduce other published results. Second, it requires each researcher to generate their own datasets, which is time-consuming and computationally demanding. Lastly, the lack of established benchmark datasets impedes the comparability of results throughout the community.

2. The absence of an open-source code base that is both reliable and flexible. Existing libraries for TO are either integrated into commercial frameworks or suffer from severe limitations and an unintuitive non-modular implementation [324]. Furthermore, we are unaware of any library for TO that allows easy integration with NNs in Python.

Our research efforts have addressed many of the critical points discussed above. First, we have released two 3D TO datasets, namely the *disc dataset* and *sphere dataset*, comprising nearly $10,000$ samples of mechanical mounting brackets. The datasets are publicly accessible on Zenodo [86], and our publication *SELTO: Sample-Efficient Learned Topology Optimization* provides a comprehensive introduction and analysis.

Second, we have introduced *DL4TO* (short for *deep learning for topology optimization*), a Python library for 3D TO based on PyTorch [211]. DL4TO facilitates easy integration with NNs, enabling research at the intersection of TO and DL. The library includes features addressing classical TO, such as a custom FDM solver for linear elasticity and an implementation of the SIMP algorithm for various objective functions. The SIMP implementation utilizes PyTorch's automatic differentiation [211] and employs the adjoint method for efficient backpropagation. Regarding DL for TO, the library provides a framework for learned TO with any NN architecture built in PyTorch [211]. Further, it supports our datasets mentioned above [86] and the simple generation of custom datasets. In our conference paper *DL4TO: A Deep Learning Library for Sample-Efficient Topology Optimization*, we introduced the library to an international audience and provided practical use cases. To date, DL4TO has been cloned by over 300 unique researchers worldwide.

Finally, we have published two papers that aim to reduce the dependency on large training datasets via equivariant DL. In *SELTO: Sample-Efficient Learned Topology Optimization*, we apply a physics-inspired end-to-end learning approach to TO. Further, in *Equivariant Neural Operators for Gradient-Consistent Topology Optimization*, we compare different FNO architectures to replace SIMP's PDE solver. Both publications showed that enforcing group equivariance and adding physical information into the learning pipeline can drastically improve sample efficiency.

# Bibliography

[1] Martin Philip Bendsøe and Noboru Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer methods in applied mechanics and engineering*, 71(2):197–224, 1988.

[2] Zeyu Zhang, Yu Li, Weien Zhou, Xiaoqian Chen, Wen Yao, and Yong Zhao. Tonr: An exploration for a novel way combining neural network with topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 386:114083, 2021.

[3] Rebekka V Woldseth, Niels Aage, J Andreas Bærentzen, and Ole Sigmund. On the use of artificial neural networks in topology optimisation. *Structural and Multidisciplinary Optimization*, 65(10):294, 2022.

[4] Alexander G Voronovich. *Wave scattering from rough surfaces*, volume 17. Springer Science & Business Media, 2013.

[5] David J Griffiths. Introduction to electrodynamics, 2005.

[6] Constantin Piron. On the foundations of quantum physics. In *Quantum Mechanics, Determinism, Causality, and Particles: An International Collection of Contributions in Honor of Louis de Broglie on the Occasion of the Jubilee of His Celebrated Thesis*, pages 105–116. Springer, 1976.

[7] Peter A Forsyth and George Labahn. Numerical methods for controlled hamilton-jacobi-bellman pdes in finance. *Journal of Computational Finance*, 11(2):1, 2007.

[8] Elizabeth E Holmes, Mark A Lewis, JE Banks, and RR Veit. Partial differential equations in ecology: spatial interactions and population dynamics. *Ecology*, 75(1):17–29, 1994.

[9] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[10] Walter A Strauss. *Partial differential equations: An introduction.* John Wiley & Sons, 2007.

[11] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Number 44 in Cambridge Texts in Applied Mathematics. Cambridge university press, 2009.

[12] A Salih. Classification of partial differential equations and canonical forms. *Lecture Notes, Department of Aerospace Engineering, Indian Institute of Space Science and Technology, Thiruvananthapuram*, 2014.

[13] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[14] David Gottlieb and Steven A Orszag. *Numerical analysis of spectral methods: theory and applications*. SIAM, 1977.

[15] Lloyd N Trefethen. *Spectral methods in MATLAB*. SIAM, 2000.

[16] Jiun-Shyan Chen, Michael Hillman, and Sheng-Wei Chi. Meshfree methods: progress made after 20 years. *Journal of Engineering Mechanics*, 143(4):04017001, 2017.

[17] Martin Neumüller and Olaf Steinbach. Refinement of flexible space–time finite element meshes and discontinuous galerkin methods. *Computing and visualization in science*, 14:189–205, 2011.

[18] Czesław I Bajer. Triangular and tetrahedral space-time finite elements in vibration analysis. *International journal for numerical methods in engineering*, 23(11):2031–2048, 1986.

[19] Qian Zhang, Jie Lu, and Yaochu Jin. Artificial intelligence in recommender systems. *Complex & Intelligent Systems*, 7(1):439–457, 2021.

[20] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[21] Steven Dewitte, Jan P Cornelis, Richard Müller, and Adrian Munteanu. Artificial intelligence revolutionises weather forecast, climate monitoring and decadal prediction. *Remote Sensing*, 13(16):3209, 2021.

[22] Siyu Teng, Xuemin Hu, Peng Deng, Bai Li, Yuchen Li, Yunfeng Ai, Dongsheng Yang, Lingxi Li, Zhe Xuanyuan, Fenghua Zhu, et al. Motion planning for autonomous driving: The state of the art and future perspectives. *IEEE Transactions on Intelligent Vehicles*, 2023.

[23] Shahin Atakishiyev, Mohammad Salameh, Hengshuai Yao, and Randy Goebel. Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for future research directions. *arXiv preprint arXiv:2112.11561*, 2021.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[26] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[27] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[28] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

[29] Xingxing Xie, Gong Cheng, Jiabao Wang, Xiwen Yao, and Junwei Han. Oriented r-cnn for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3520–3529, 2021.

[30] Kuang Liu, Mingmin Zhang, and Zhigeng Pan. Facial expression recognition with cnn ensemble. In *2016 international conference on cyberworlds (CW)*, pages 163–166. IEEE, 2016.

[31] Baris Kayalibay, Grady Jensen, and Patrick van der Smagt. Cnn-based segmentation of medical imaging data. *arXiv preprint arXiv:1701.03056*, 2017.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[33] Wenjing Tong, Li Song, Xiaokang Yang, Hui Qu, and Rong Xie. Cnn-based shot boundary detection and video annotation. In *2015 IEEE international symposium on broadband multimedia systems and broadcasting*, pages 1–5. IEEE, 2015.

[34] Zhongwen Xu, Yi Yang, and Alex G Hauptmann. A discriminative cnn video representation for event detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1798–1807, 2015.

[35] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[36] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[37] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[39] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[40] WF Bauer. The monte carlo method. *Journal of the Society for Industrial and Applied Mathematics*, 6(4):438–451, 1958.

[41] L eon Bottou. Online learning and stochastic approximations. *Online learning in neural networks*, 17(9):142, 1998.

[42] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[43] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[44] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[45] Raul Rojas and Raúl Rojas. The backpropagation algorithm. *Neural networks: a systematic introduction*, pages 149–182, 1996.

[46] Michael A Nielsen. *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA, 2015.

[47] James Edward Gordon. *Structures: or why things don't fall down*. Da Capo Press, 2009.

[48] Anthony George Maldon Michell. Lviii. the limits of economy of material in frame-structures. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 8(47):589–597, 1904.

[49] Mehdi Tavakoli, Amin Tavakoli, Mehdi Taheri, and Hossein Saghafifar. Design, simulation and structural optimization of a longitudinal acoustic resonator for trace gas detection using laser photoacoustic spectroscopy (lpas). *Optics & Laser Technology*, 42(5):828–838, 2010.

[50] Nima Aghajari and Michael Schäfer. Efficient shape optimization for fluid–structure interaction problems. *Journal of Fluids and Structures*, 57:298–313, 2015.

[51] Rupali Sahu, Vrushang Patel, Sandeep Kumar Singh, and BS Munjal. Structural optimization of a space mirror to selectively constrain optical aberrations. *Structural and Multidisciplinary Optimization*, 55:2353–2363, 2017.

[52] Francesco Lucchini, Riccardo Torchio, Vincenzo Cirimele, Piergiorgio Alotto, and Paolo Bettini. Topology optimization for electromagnetics: A survey. *IEEE Access*, 10:98593–98611, 2022.

[53] Peter W Christensen and Anders Klarbring. *An introduction to structural optimization*, volume 153. Springer Science & Business Media, 2008.

[54] Xavier Oliver Olivella and Carlos Agelet de Saracibar Bosch. Continuum mechanics for engineers. theory and problems, 2017.

[55] Daniel Yago, Juan Cante, Oriol Lloberas-Valls, and Javier Oliver. Topology optimization methods for 3d structural problems: a comparative study. *Archives of Computational Methods in Engineering*, 29(3):1525–1567, 2022.

[56] Jan Rychlewski. On hooke's law. *Journal of Applied Mathematics and Mechanics*, 48(3):303–314, 1984.

[57] Kyung K Choi and Nam-Ho Kim. *Structural sensitivity analysis and optimization 1: linear systems.* Springer Science & Business Media, 2004.

[58] Boris Desmorat and Rodrigue Desmorat. Topology optimization in damage governed low cycle fatigue. *Comptes Rendus Mecanique*, 336(5):448–453, 2008.

[59] Matteo Bruggi. On an alternative approach to stress constraints relaxation in topology optimization. *Structural and multidisciplinary optimization*, 36:125–141, 2008.

[60] Chau Le, Julian Norato, Tyler Bruns, Christopher Ha, and Daniel Tortorelli. Stress-based topology optimization for continua. *Structural and Multidisciplinary Optimization*, 41:605–620, 2010.

[61] Erik Holmberg. *Stress and fatigue constrained topology optimization.* PhD thesis, Linköping University Electronic Press, 2013.

[62] Nicolas Boullé and Alex Townsend. A mathematical guide to operator learning. *arXiv preprint arXiv:2312.14688*, 2023.

[63] Thomas Borrvall and Joakim Petersson. Large-scale topology optimization in 3d using parallel computing. *Computer methods in applied mechanics and engineering*, 190(46-47):6201–6229, 2001.

[64] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[65] Stanley C Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM Journal on Scientific and Statistical Computing*, 2(1):1–4, 1981.

[66] Pieter Wesseling. Introduction to multigrid methods. Technical report, 1995.

[67] Timothy A Davis. Algorithm 832: Umfpack v4. 3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):196–199, 2004.

[68] Anders Hviid, Steven Rubin, and Kathrin Mühlemann. Mumps. *The Lancet*, 371(9616):932–944, 2008.

[69] Iain S Duff, Albert Maurice Erisman, and John Ker Reid. *Direct methods for sparse matrices*. Oxford University Press, 2017.

[70] Grégoire Allaire, François Jouve, and Hervé Maillot. Topology optimization for minimum stress design with the homogenization method. *Structural and Multidisciplinary Optimization*, 28:87–98, 2004.

[71] Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2003.

[72] Xiaodong Huang and Mike Xie. *Evolutionary topology optimization of continuum structures: methods and applications*. John Wiley & Sons, 2010.

[73] Weihong Zhang and Shiping Sun. Scale-related topology optimization of cellular materials and structures. *International Journal for numerical methods in Engineering*, 68(9):993–1011, 2006.

[74] Joshua D Deaton and Ramana V Grandhi. A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Structural and Multidisciplinary Optimization*, 49:1–38, 2014.

[75] Xu Guo and Geng-Dong Cheng. Recent development in structural design and optimization. *Acta Mechanica Sinica*, 26(6):807–823, 2010.

[76] Ole Sigmund and Kurt Maute. Topology optimization approaches: A comparative review. *Structural and multidisciplinary optimization*, 48(6):1031–1055, 2013.

[77] Robert V Kohn and Gilbert Strang. Optimal design and relaxation of variational problems, i. *Communications on pure and applied mathematics*, 39(1):113–137, 1986.

[78] Robert V Kohn and Gilbert Strang. Optimal design and relaxation of variational problems, ii. *Communications on Pure and Applied Mathematics*, 39(2):139–182, 1986.

[79] Muriel Beckers. Topology optimization using a dual method with discrete variables. *Structural optimization*, 17:14–24, 1999.

[80] Prabhat Hajela and E Lee. Genetic algorithms in truss topological optimization. *International journal of solids and structures*, 32(22):3341–3357, 1995.

[81] Ole Sigmund. On the usefulness of non-gradient approaches in topology optimization. *Structural and Multidisciplinary Optimization*, 43:589–596, 2011.

[82] Martin P Bendsøe. Optimal shape design as a material distribution problem. *Structural optimization*, 1:193–202, 1989.

[83] Ming Zhou and George IN Rozvany. The coc algorithm, part ii: Topological, geometrical and generalized shape optimization. *Computer methods in applied mechanics and engineering*, 89(1-3):309–336, 1991.

[84] George IN Rozvany, Ming Zhou, and Torben Birker. Generalized shape optimization without homogenization. *Structural optimization*, 4:250–252, 1992.

[85] Joakim Petersson. A finite element analysis of optimal variable thickness sheets. *SIAM journal on numerical analysis*, 36(6):1759–1778, 1999.

[86] Sören Dittmer, David Erzmann, Henrik Harms, Rielson Falck, and Marco Gosch. Selto dataset. 10.5281/zenodo.7034898, 2023.

[87] G Allaire and GA Francfort. A numerical algorithm for topology and shape optimization. In *Topology design of structures*, pages 239–248. Springer, 1993.

[88] Grégoire Allaire and RV Kohn. Topology optimization and optimal shape design using homogenization. In *Topology design of structures*, pages 207–218. Springer, 1993.

[89] Michael B Giles and Niles A Pierce. An introduction to the adjoint approach to design. *Flow, turbulence and combustion*, 65:393–415, 2000.

[90] Sriram Jameson and Antony Jameson. Adjoint formulations for topology, shape and discrete optimization. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, page 55, 2007.

[91] Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020.

[92] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)*, 23(3):449–456, 2004.

[93] M Zhou, YK Shyy, and HL Thomas. Checkerboard and minimum member size control in topology optimization. *Structural and Multidisciplinary Optimization*, 21:152–158, 2001.

[94] Thomas Borrvall. Topology optimization of elastic continua using restriction. *Archives of Computational Methods in Engineering*, 8:351–385, 2001.

[95] Ole Sigmund and Joakim Petersson. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural optimization*, 16:68–75, 1998.

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

[97] Tyler E Bruns and Daniel A Tortorelli. Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer methods in applied mechanics and engineering*, 190(26-27):3443–3459, 2001.

[98] Kazem Ghabraie. *Exploring topology and shape optimisation techniques in underground excavations.* PhD thesis, RMIT University, 2009.

[99] James K Guest, Jean H Prévost, and Ted Belytschko. Achieving minimum length scale in topology optimization using nodal design variables and projection functions. *International journal for numerical methods in engineering*, 61(2):238–254, 2004.

[100] James K Guest, Alireza Asadpoure, and Seung-Hyun Ha. Eliminating beta-continuation from heaviside projection and density filter algorithms. *Structural and Multidisciplinary Optimization*, 44:443–453, 2011.

[101] Atsushi Kawamoto, Tadayoshi Matsumori, Shintaro Yamasaki, Tsuyoshi Nomura, Tsuguo Kondoh, and Shinji Nishiwaki. Heaviside projection based topology optimization by a pde-filtered scalar function. *Structural and Multidisciplinary Optimization*, 44:19–24, 2011.

[102] Alessandro Lanza, Serena Morigi, Ivan W Selesnick, and Fiorella Sgallari. Convex non-convex variational models. In *Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging: Mathematical Imaging and Vision*, pages 1–57. Springer, 2022.

[103] JM Martinez. A note on the theoretical convergence properties of the simp method. *Structural and Multidisciplinary Optimization*, 29:319–323, 2005.

[104] Mathias Stolpe and Krister Svanberg. On the trajectories of penalization methods for topology optimization. *Structural and Multidisciplinary Optimization*, 21:128–139, 2001.

[105] Ole Sigmund. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33:401–424, 2007.

[106] George IN Rozvany. Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics. *Structural and Multidisciplinary optimization*, 21:90–108, 2001.

[107] Hans A Eschenauer and Niels Olhoff. Topology optimization of continuum structures: a review. *Appl. Mech. Rev.*, 54(4):331–390, 2001.

[108] George IN Rozvany. A critical review of established methods of structural topology optimization. *Structural and multidisciplinary optimization*, 37:217–237, 2009.

[109] YM Xie and Grant P Steven. Shape and layout optimization via an evolutionary procedure. In *Proceedings of the international conference on computational engineering science*, 1992.

[110] Yi Min Xie and Grant P Steven. A simple evolutionary procedure for structural optimization. *Computers & structures*, 49(5):885–896, 1993.

[111] Y Mike Xie, Grant P Steven, YM Xie, and GP Steven. *Basic evolutionary structural optimization.* Springer, 1997.

[112] Eckart Schnack, Uwe Spörl, and Gabriela Iancu. *Gradientless shape optimization with FEM.* Verein Deutscher Ingenieure, 1988.

[113] Claus Mattheck and S Burkhardt. A new method of structural shape optimization based on biological growth. *International Journal of Fatigue*, 12(3):185–190, 1990.

[114] Pasi Tanskanen. The evolutionary structural optimization method: theoretical aspects. *Computer methods in applied mechanics and engineering*, 191(47-48):5485–5498, 2002.

[115] M Zhou and GIN Rozvany. On the validity of eso type methods in topology optimization. *Structural and Multidisciplinary Optimization*, 21(1):80–83, 2001.

[116] Mathias Stolpe and Martin P Bendsøe. Global optima for the zhou–rozvany problem. *Structural and Multidisciplinary Optimization*, 43:151–164, 2011.

[117] Philip Anthony Browne. *Topology optimization of linear elastic structures.* PhD thesis, University of Bath, 2013.

[118] YM Xie and Grant P Steven. Optimal design of multiple load case structures using an evolutionaryprocedure. *Engineering computations*, 11(4):295–302, 1994.

[119] YM Xie and GP Steven. A simple approach to structural frequency optimization. *Computers & structures*, 53(6):1487–1491, 1994.

[120] Chongbin Zhao, GP Steven, and YM Xie. Evolutionary natural frequency optimization of two-dimensional structures with additional non-structural lumped masses. *Engineering Computations*, 14(2):233–251, 1997.

[121] O Querin, G Steven, and Y Xie. Topology optimisation of structures with material and geometric non-linearities. In *6th Symposium on Multidisciplinary Analysis and Optimization*, page 4116, 1996.

[122] D Manickarajah, YM Xie, and GP Steven. An evolutionary method for optimization of plate buckling resistance. *Finite Elements in Analysis and Design*, 29(3-4):205–230, 1998.

[123] Qing Li, Grant P Steven, Osvaldo M Querin, and YM Xie. Shape and topology design for heat conduction by evolutionary structural optimization. *International Journal of Heat and Mass Transfer*, 42(17):3361–3371, 1999.

[124] Qing Li, Grant P Steven, Osvaldo M Querin, and YM Xie. Structural topology design with multiple thermal criteria. *Engineering Computations*, 17(6):715–734, 2000.

[125] Qing Li, Grant P Steven, and YM Xie. Thermoelastic topology optimization for problems with varying temperature fields. *Journal of Thermal Stresses*, 24(4):347–366, 2001.

[126] W Li, G Steven, and Y Xie. Shape design for elastic contact problems by evolutionary structural optimization. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, page 4851, 1998.

[127] Wei Li, Qing Li, Grant P Steven, and YM Xie. An evolutionary shape optimization for elastic contact problems subject to multiple load cases. *Computer methods in applied mechanics and engineering*, 194(30-33):3394–3415, 2005.

[128] Xiao Ying Yang, Yi Min Xie, Grant P Steven, and OM Querin. Bidirectional evolutionary method for stiffness optimization. *AIAA journal*, 37(11):1483–1488, 1999.

[129] Osvaldo M Querin, Grant P Steven, and Yi Min Xie. Evolutionary structural optimisation (eso) using a bidirectional algorithm. *Engineering computations*, 15(8):1031–1048, 1998.

[130] OM Querin, V Young, GP Steven, and YM Xie. Computational efficiency and validation of bi-directional evolutionary structural optimisation. *Computer methods in applied mechanics and engineering*, 189(2):559–573, 2000.

[131] Q Li, GP Steven, and YM Xie. A simple checkerboard suppression algorithm for evolutionary structural optimization. *Structural and Multidisciplinary Optimization*, 22:230–239, 2001.

[132] Xiaodong Huang and YM Xie. Convergent and mesh-independent solutions for the bi-directional evolutionary structural optimization method. *Finite elements in analysis and design*, 43(14):1039–1049, 2007.

[133] Xiao-Ying Yang, Yi-Min Xie, Jin-Shen Liu, GT Parks, and PJ Clarkson. Perimeter control in the bidirectional evolutionary optimization method. *Structural and Multidisciplinary Optimization*, 24:430–440, 2002.

[134] Xiaodong Huang and Yi Min Xie. Bi-directional evolutionary topology optimization of continuum structures with one or multiple materials. *Computational Mechanics*, 43:393–401, 2009.

[135] David Adalsteinsson and James A Sethian. The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148(1):2–22, 1999.

[136] Stanley Osher and Ronald P Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 1. Springer New York, 2005.

[137] Robert Haber and Martin Bendsoe. Problem formulation, solution procedures and geometric modeling-key issues in variable-topology optimization. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, page 4948, 1998.

[138] Michael Yu Wang, Xiaoming Wang, and Dongming Guo. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192(1-2):227–246, 2003.

[139] Grégoire Allaire, F de Gournay, François Jouve, and A-M Toader. Structural optimization using topological and shape sensitivity via a level set method. *Control and cybernetics*, 34(1):59–80, 2005.

[140] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[141] James A Sethian et al. *Level set methods and fast marching methods*, volume 98. Cambridge Cambridge UP, 1999.

[142] 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.

[143] Peter Donald Dunning. *Introducing loading uncertainty in level set-based structural topology optimisation*. PhD thesis, University of Bath, 2011.

[144] Mei Yulin and Wang Xiaoming. A level set method for structural topology optimization and its applications. *Advances in Engineering software*, 35(7):415–441, 2004.

[145] Zhen Luo, Michael Yu Wang, Shengyin Wang, and Peng Wei. A level set-based parameterization method for structural shape and topology optimization. *International Journal for Numerical Methods in Engineering*, 76(1):1–26, 2008.

[146] Stanley J Osher and Fadil Santosa. Level set methods for optimization problems involving geometry and constraints: I. frequencies of a two-density inhomogeneous drum. *Journal of Computational Physics*, 171(1):272–288, 2001.

[147] Junzhao Luo, Zhen Luo, Liping Chen, Liyong Tong, and Michael Yu Wang. A semi-implicit level set method for structural shape and topology optimization. *Journal of Computational Physics*, 227(11):5561–5581, 2008.

[148] Mojtaba Mohamadian and Saeed Shojaee. Binary level set method for structural topology optimization with mbo type of projection. *International journal for numerical methods in engineering*, 89(5):658–670, 2012.

[149] Grégoire Allaire, François Jouve, and Anca-Maria Toader. A level-set method for shape optimization. *Comptes rendus. Mathématique*, 334(12):1125–1130, 2002.

[150] Grégoire Allaire and François Jouve. A level-set method for vibration and multiple loads structural optimization. *Computer methods in applied mechanics and engineering*, 194(30-33):3269–3290, 2005.

[151] Seung-Hyun Ha and Seonho Cho. Topological shape optimization of heat conduction problems using level set approach. *Numerical Heat Transfer, Part B: Fundamentals*, 48(1):67–88, 2005.

[152] Martin Burger and Stanley J Osher. A survey on level set methods for inverse problems and optimal design. *European journal of applied mathematics*, 16(2):263–301, 2005.

[153] Hans A Eschenauer, Vladimir V Kobelev, and Axel Schumacher. Bubble method for topology and shape optimization of structures. *Structural optimization*, 8:42–51, 1994.

[154] Axel Schumacher. *Topologieoptimierung von Bauteilstrukturen unter Verwendung von Lochpositionierungskriterien.* PhD thesis, Inst. für Mechanik und Regelungstechnik, 1996.

[155] Jan Sokolowski and Antoni Zochowski. On the topological derivative in shape optimization. *SIAM journal on control and optimization*, 37(4):1251–1272, 1999.

[156] Stéphane Garreau, Philippe Guillaume, and Mohamed Masmoudi. The topological asymptotic for pde systems: the elasticity case. *SIAM journal on control and optimization*, 39(6):1756–1778, 2001.

[157] Michael Yu Wang and Xiaoming Wang. Pde-driven level sets, shape sensitivity and curvature flow for structural topology optimization. *Computer Modeling in Engineering and Sciences*, 6:373–396, 2004.

[158] Zhenyu Liu, Jan G Korvink, and Ruoyu Huang. Structure topology optimization: fully coupled level set method via femlab. *Structural and Multidisciplinary Optimization*, 29:407–417, 2005.

[159] Xiang Li, Zhanli Liu, Shaoqing Cui, Chengcheng Luo, Chenfeng Li, and Zhuo Zhuang. Predicting the effective mechanical property of heterogeneous materials by image based modeling and deep learning. *Computer Methods in Applied Mechanics and Engineering*, 347:735–753, 2019.

[160] German Capuano and Julian J Rimoli. Smart finite elements: A novel machine learning application. *Computer Methods in Applied Mechanics and Engineering*, 345:363–381, 2019.

[161] Houpu Yao, Yi Gao, and Yongming Liu. Fea-net: A physics-guided data-driven model for efficient mechanical response prediction. *Computer Methods in Applied Mechanics and Engineering*, 363:112892, 2020.

[162] Zhongqing Han, Suvranu De, et al. A deep learning-based hybrid approach for the solution of multiphysics problems in electrosurgery. *Computer methods in applied mechanics and engineering*, 357:112603, 2019.

[163] Miguel A Bessa, Piotr Glowacki, and Michael Houlder. Bayesian machine learning in metamaterial design: Fragile becomes supercompressible. *Advanced Materials*, 31(48):1904845, 2019.

[164] Hojjat Adeli and Hyo Seon Park. A neural dynamics model for structural optimization—theory. *Computers & structures*, 57(3):383–390, 1995.

[165] Manolis Papadrakakis, Nikos D Lagaros, and Yiannis Tsompanakis. Structural optimization using evolution strategies and neural networks. *Computer methods in applied mechanics and engineering*, 156(1-4):309–333, 1998.

[166] Srinivas Kodiyalam and Ram Gurumoorthy. Neural networks with modified backpropagation learning applied to structural optimization. *AIAA journal*, 34(2):408–412, 1996.

[167] Erva Ulu, Rusheng Zhang, and Levent Burak Kara. 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.

[168] Ivan Sosnovik and Ivan Oseledets. Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 34(4):215–223, 2017.

[169] Liang Xue, Jie Liu, Guilin Wen, and Hongxin Wang. Efficient, high-resolution topology optimization method based on convolutional neural networks. *Frontiers of Mechanical Engineering*, 16(1):80–96, 2021.

[170] Saurabh Banga, Harsh Gehani, Sanket Bhilare, Sagar Patel, and Levent Kara. 3d topology optimization using convolutional neural networks. *arXiv preprint arXiv:1808.07440*, 2018.

[171] Ren Kai Tan, Chao Qian, Dan Xu, and Wenjing Ye. An adaptive and scalable ann-based model-order-reduction method for large-scale to designs. *arXiv preprint arXiv:2203.10515*, 2022.

[172] Fernando V Senhora, Heng Chi, Yuyu Zhang, Lucia Mirabella, Tsz Ling Elaine Tang, and Glaucio H Paulino. Machine learning for topology optimization: Physics-based learning through an independent training strategy. *Computer Methods in Applied Mechanics and Engineering*, 398:115116, 2022.

[173] Diab W Abueidda, Seid Koric, and Nahil A Sobh. Topology optimization of 2d structures with nonlinearities using deep learning. *Computers & Structures*, 237:106283, 2020.

[174] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[175] Nikos Ath Kallioras, Georgios Kazakis, and Nikos D Lagaros. Accelerated topology optimization by means of deep learning. *Structural and Multidisciplinary Optimization*, 62(3):1185–1212, 2020.

[176] Yonggyun Yu, Taeil Hur, Jaeho Jung, and In Gwun Jang. Deep learning for determining a near-optimal topological design without any iteration. *Structural and Multidisciplinary Optimization*, 59(3):787–799, 2019.

[177] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[178] Jiaxiang Luo, Yu Li, Weien Zhou, Zhiqiang Gong, Zeyu Zhang, and Wen Yao. An improved data-driven topology optimization method using feature pyramid networks with physical constraints. *Comput Model Eng Sci*, 128(3):823–848, 2021.

[179] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

[180] Baotong Li, Congjia Huang, Xin Li, Shuai Zheng, and Jun Hong. Non-iterative structural topology optimization using deep learning. *Computer-Aided Design*, 115:172–180, 2019.

[181] Shuai Zheng, Haojie Fan, Ziyu Zhang, Zhiqiang Tian, and Kang Jia. Accurate and real-time structural topology prediction driven by deep learning under moving morphable component-based framework. *Applied Mathematical Modelling*, 97:522–535, 2021.

[182] Van-Nam Hoang, Ngoc-Linh Nguyen, Dat Q Tran, Quang-Viet Vu, and H Nguyen-Xuan. Data-driven geometry-based topology optimization. *Structural and Multidisciplinary Optimization*, 65(2):69, 2022.

[183] Jaydeep Rade, Aditya Balu, Ethan Herron, Jay Pathak, Rishikesh Ranade, Soumik Sarkar, and Adarsh Krishnamurthy. Physics-consistent deep learning for structural topology optimization. *arXiv preprint arXiv:2012.05359*, 2020.

[184] Zhenguo Nie, Tong Lin, Haoliang Jiang, and Levent Burak Kara. Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain. *Journal of Mechanical Design*, 143(3), 2021.

[185] Jun Yan, Qi Zhang, Qi Xu, Zhirui Fan, Haijiang Li, Wei Sun, and Guangyuan Wang. Deep learning driven real time topology optimisation based on initial stress learning. *Advanced Engineering Informatics*, 51:101472, 2022.

[186] Dalei Wang, Cheng Xiang, Yue Pan, Airong Chen, Xiaoyi Zhou, and Yiquan Zhang. A deep convolutional neural network for topology optimization with perceptible generalization ability. *Engineering Optimization*, 54(6):973–988, 2022.

[187] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[188] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[189] Marc Moreno Lopez and Jugal Kalita. Deep learning applied to nlp. *arXiv preprint arXiv:1703.03091*, 2017.

[190] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[191] Sylvain Christin, Éric Hervet, and Nicolas Lecomte. Applications for deep learning in ecology. *Methods in Ecology and Evolution*, 10(10):1632–1644, 2019.

[192] Niels Aage, Erik Andreassen, and Boyan Stefanov Lazarov. Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization*, 51(3):565–572, 2015.

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

[194] Mads Baandrup, Ole Sigmund, Henrik Polk, and Niels Aage. Closing the gap towards super-long suspension bridges using computational morphogenesis. *Nature communications*, 11(1):2735, 2020.

[195] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[196] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 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.

[197] Tianju Xue, Alex Beatson, Sigrid Adriaenssens, and Ryan Adams. Amortized finite element analysis for fast pde-constrained optimization. In *International Conference on Machine Learning*, pages 10638–10647. PMLR, 2020.

[198] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[199] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[200] Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*, 2023.

[201] Bing Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[202] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[203] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

[204] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[205] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

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

[207] Hao Xu, Haibin Chang, and Dongxiao Zhang. Dl-pde: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data. *arXiv preprint arXiv:1908.04463*, 2019.

[208] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

[209] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International conference on machine learning*, pages 3208–3216. PMLR, 2018.

[210] Gert-Jan Both, Subham Choudhury, Pierre Sens, and Remy Kusters. Deepmod: Deep learning for model discovery in noisy data. *Journal of Computational Physics*, 428:109985, 2021.

[211] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

[212] Robert Stephany and Christopher Earls. Pde-read: Human-readable partial differential equation discovery using deep learning. *Neural Networks*, 154:360–382, 2022.

[213] Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):6136, 2021.

[214] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

[215] Christophe Bonneville and Christopher Earls. Bayesian deep learning for partial differential equation parameter discovery with sparse and noisy data. *Journal of Computational Physics: X*, 16:100115, 2022.

[216] Daniel A Messenger and David M Bortz. Weak sindy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021.

[217] Daniel R Gurevich, Patrick AK Reinbold, and Roman O Grigoriev. Robust and optimal sparse regression for nonlinear pde models. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(10), 2019.

[218] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[219] Zhijie Li, Wenhui Peng, Zelong Yuan, and Jianchun Wang. Long-term predictions of turbulence by implicit u-net enhanced fourier neural operator. *Physics of Fluids*, 35(7), 2023.

[220] Wenhui Peng, Zelong Yuan, and Jianchun Wang. Attention-enhanced neural network models for turbulence simulation. *Physics of Fluids*, 34(2), 2022.

[221] Huaiqian You, Quinn Zhang, Colton J Ross, Chung-Hao Lee, and Yue Yu. Learning deep implicit fourier neural operators (ifnos) with applications to heterogeneous material modeling. *Computer Methods in Applied Mechanics and Engineering*, 398:115296, 2022.

[222] Shunyuan Mao, Ruobing Dong, Lu Lu, Kwang Moo Yi, Sifan Wang, and Paris Perdikaris. Ppdonet: Deep operator networks for fast prediction of steady-state solutions in disk–planet systems. *The Astrophysical Journal Letters*, 950(2):L12, 2023.

[223] Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–11, 2023.

[224] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[225] Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-informed deep neural operator networks. In *Machine Learning in Modeling and Simulation: Methods and Applications*, pages 219–254. Springer, 2023.

[226] Derick Nganyu Tanyu, Jianfeng Ning, Tom Freudenberg, Nick Heilenkötter, Andreas Rademacher, Uwe Iben, and Peter Maass. Deep learning methods for partial differential equations and related parameter identification problems. *Inverse Problems*, 39(10):103001, 2023.

[227] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

[228] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[229] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[230] Beichuan Deng, Yeonjong Shin, Lu Lu, Zhongqiang Zhang, and George Em Karniadakis. Approximation rates of deeponets for learning operators arising from advection–diffusion equations. *Neural Networks*, 153:411–426, 2022.

[231] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[232] Simone Venturi and Tiernan Casey. Svd perspectives for augmenting deeponet flexibility and interpretability. *Computer Methods in Applied Mechanics and Engineering*, 403:115718, 2023.

[233] Samuel Lanthaler, Siddhartha Mishra, and George E Karniadakis. Error estimates for deeponets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022.

[234] Somdatta Goswami, Minglang Yin, Yue Yu, and George Em Karniadakis. A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391:114587, 2022.

[235] P Clark Di Leoni, Lu Lu, Charles Meneveau, George Karniadakis, and Tamer A Zaki. Deeponet prediction of linear instability waves in high-speed boundary layers. *arXiv preprint arXiv:2105.08697*, 2021.

[236] Michael Renardy and Robert C Rogers. *An introduction to partial differential equations*, volume 13. Springer Science & Business Media, 2006.

[237] Yannick Augenstein, Taavi Repan, and Carsten Rockstuhl. Neural operator-based surrogate solver for free-form electromagnetic inverse design. *ACS Photonics*, 2023.

[238] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022.

[239] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *arXiv preprint arXiv:2309.00583*, 2023.

[240] Oscar P Bruno, Youngae Han, and Matthew M Pohlman. Accurate, high-order representation of complex three-dimensional surfaces via fourier continuation analysis. *Journal of computational Physics*, 227(2):1094–1125, 2007.

[241] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

[242] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *The Journal of Machine Learning Research*, 22(1):13237–13312, 2021.

[243] Samuel Lanthaler and Andrew M Stuart. The curse of dimensionality in operator learning. *arXiv preprint arXiv:2306.15924*, 2023.

[244] Maarten V de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M Stuart. The cost-accuracy trade-off in operator learning with neural networks. *arXiv preprint arXiv:2203.13181*, 2022.

[245] Zhenguo Nie, Haoliang Jiang, and Levent Burak Kara. Stress field prediction in cantilevered structures using convolutional neural networks. *Journal of Computing and Information Science in Engineering*, 20(1):011002, 2020.

[246] Daniel A White, William J Arrighi, Jun Kudo, and Seth E Watts. Multiscale topology optimization using neural network surrogate models. *Computer Methods in Applied Mechanics and Engineering*, 346:1118–1135, 2019.

[247] Karl A Kalina, Lennart Linden, Jörg Brummund, Philipp Metsch, and Markus Kästner. Automated constitutive modeling of isotropic hyperelasticity based on artificial neural networks. *Computational Mechanics*, pages 1–20, 2022.

[248] Hamed Bolandi, Gautam Sreekumar, Xuyang Li, Nizar Lajnef, and Vishnu Naresh Boddeti. Physics informed neural network for dynamic stress prediction. *Applied Intelligence*, pages 1–16, 2023.

[249] Salah A Faroughi, Nikhil Pawar, Celio Fernandes, Subasish Das, Nima K Kalantari, and Seyed Kourosh Mahjour. Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing. *arXiv preprint arXiv:2211.07377*, 2022.

[250] Aditya Joglekar, Hongrui Chen, and Levent Burak Kara. Dmf-tonn: Direct mesh-free topology optimization using neural networks. *arXiv preprint arXiv:2305.04107*, 2023.

[251] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[252] Junyan He, Seid Koric, Shashank Kushwaha, Jaewan Park, Diab Abueidda, and Iwona Jasiuk. Novel deeponet architecture to predict stresses in elasto-plastic structures with variable complex geometries and loads. *arXiv preprint arXiv:2306.03645*, 2023.

[253] Lu Lu, Raphaël Pestourie, Steven G Johnson, and Giuseppe Romano. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Physical Review Research*, 4(2):023210, 2022.

[254] Hyogu Jeong, Jinshuai Bai, Chanaka Prabuddha Batuwatta-Gamage, Charith Rathnayaka, Ying Zhou, and YuanTong Gu. A physics-informed neural network-based topology optimization (pinnto) framework for structural optimization. *Engineering Structures*, 278:115484, 2023.

[255] Seid Koric, Asha Viswantah, Diab W Abueidda, Nahil A Sobh, and Kamran Khan. Deep learning operator network for plastic deformation with variable loads and material properties. *Engineering with Computers*, pages 1–13, 2023.

[256] Heng Chi, Yuyu Zhang, Tsz Ling Elaine Tang, Lucia Mirabella, Livio Dalloro, Le Song, and Glaucio H Paulino. Universal machine learning for topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 375:112739, 2021.

[257] Seunghye Lee, Hyunjoo Kim, Qui X Lieu, and Jaehong Lee. Cnn-based image recognition for topology optimization. *Knowledge-Based Systems*, 198:105887, 2020.

[258] Yuyu Zhang, Heng Chi, Binghong Chen, Tsz Ling Elaine Tang, Lucia Mirabella, Le Song, and Glaucio H Paulino. Speeding up computational morphogenesis with online neural synthetic gradients. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[259] Chao Qian and Wenjing Ye. Accelerating gradient-based topology optimization design with dual-model artificial neural networks. *Structural and Multidisciplinary Optimization*, 63(4):1687–1707, 2021.

[260] Sebastian Lunz, Andreas Hauptmann, Tanja Tarvainen, Carola-Bibiane Schonlieb, and Simon Arridge. On learned operator correction in inverse problems. *SIAM Journal on Imaging Sciences*, 14(1):92–127, 2021.

[261] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[262] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[263] Victor Lempitsky, Andrea Vedaldi, and Dmitry Ulyanov. Deep image prior. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9446–9454. IEEE, 2018.

[264] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[265] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[266] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4743–4752, 2019.

[267] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[268] Zhichun Huang, Shaojie Bai, and J Zico Kolter. Implicit2: Implicit layers for implicit representations. *Advances in Neural Information Processing Systems*, 34:9639–9650, 2021.

[269] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[270] Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A structured dictionary perspective on implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19228–19238, 2022.

[271] Jing Zou, Noémie Debroux, Lihao Liu, Jing Qin, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. Homeomorphic image registration via conformal-invariant hyperelastic regularisation. *arXiv preprint arXiv:2303.08113*, 2023.

[272] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020.

[273] Ninad Khargonkar, Neil Song, Zesheng Xu, Balakrishnan Prabhakaran, and Yu Xiang. Neuralgrasps: Learning implicit representations for grasps of multiple robotic hands. In *Conference on Robot Learning*, pages 516–526. PMLR, 2023.

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

[275] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

[276] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[277] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018.

[278] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[279] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67, 2018.

[280] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1886–1895, 2018.

[281] Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinicius da Silva, Helio Lopes, and Luiz Velho. Exploring differential geometry in neural implicits. *Computers & Graphics*, 108:49–60, 2022.

[282] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. Implicit neural representations with levels-of-experts. *Advances in Neural Information Processing Systems*, 35:2564–2576, 2022.

[283] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18507–18516, 2023.

[284] Zhenda Shen, Yanqi Cheng, Raymond H Chan, Pietro Liò, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. Trident: The nonlinear trilogy for implicit neural representations. *arXiv preprint arXiv:2311.13610*, 2023.

[285] MJ De Ruiter and F Van Keulen. Topology optimization using a topology description function. *Structural and Multidisciplinary Optimization*, 26(6):406–416, 2004.

[286] Hao Deng and Albert C To. A parametric level set method for topology optimization based on deep neural network. *Journal of Mechanical Design*, 143(9):091702, 2021.

[287] Zeyu Zhang, Wen Yao, Yu Li, Weien Zhou, and Xiaoqian Chen. Topology optimization via implicit neural representations. *Computer Methods in Applied Mechanics and Engineering*, 411:116052, 2023.

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

[289] Aaditya Chandrasekhar and Krishnan Suresh. Multi-material topology optimization using neural networks. *Computer-Aided Design*, 136:103017, 2021.

[290] Hao Deng and Albert C To. Topology optimization based on deep representation learning (drl) for compliance and stress-constrained design. *Computational Mechanics*, 66(2):449–469, 2020.

[291] Hyogu Jeong, Chanaka Batuwatta-Gamage, Jinshuai Bai, Yi Min Xie, Charith Rathnayaka, Ying Zhou, and YuanTong Gu. A complete physics-informed neural network-based framework for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 417:116401, 2023.

[292] Arturs Berzins, Andreas Radler, Sebastian Sanokowski, Sepp Hochreiter, and Johannes Brandstetter. Geometry-informed neural networks. *arXiv preprint arXiv:2402.14009*, 2024.

[293] Silvia Biasotti, Leila De Floriani, Bianca Falcidieno, Patrizio Frosini, Daniela Giorgi, Claudia Landi, Laura Papaleo, and Michela Spagnuolo. Describing shapes by geometrical-topological properties of real functions. *ACM Computing Surveys (CSUR)*, 40(4):1–87, 2008.

[294] Amin Heyrani Nobari, Giorgio Giannone, Lyle Regenwetter, and Faez Ahmed. Nito: Neural implicit fields for resolution-free topology optimization. *arXiv preprint arXiv:2402.05073*, 2024.

[295] Jonas Zehnder, Yue Li, Stelian Coros, and Bernhard Thomaszewski. Ntopo: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems*, 34:10368–10381, 2021.

[296] Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. Deep generative models in engineering design: A review. *Journal of Mechanical Design*, 144(7):071704, 2022.

[297] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[298] Xu Guo, Weisheng Zhang, and Wenliang Zhong. Doing topology optimization explicitly and geometrically—a new moving morphable components based framework. *Journal of Applied Mechanics*, 81(8):081009, 2014.

[299] Tinghao Guo, Danny J Lohan, Ruijin Cang, Max Yi Ren, and James T Allison. An indirect design representation for topology optimization using variational autoencoder and style transfer. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0804, 2018.

[300] Praveen S Vulimiri, Hao Deng, Florian Dugast, Xiaoli Zhang, and Albert C To. Integrating geometric data into topology optimization via neural style transfer. *Materials*, 14(16):4551, 2021.

[301] Sangeun Oh, Yongsu Jung, Seongsin Kim, Ikjin Lee, and Namwoo Kang. Deep generative design: Integration of topology optimization and generative models. *Journal of Mechanical Design*, 141(11), 2019.

[302] Sharad Rawat and M-H Herman Shen. A novel topology optimization approach using conditional deep learning. *arXiv preprint arXiv:1901.04859*, 2019.

[303] M-H Herman Shen and Liang Chen. A new cgan technique for constrained topology design optimization. *arXiv preprint arXiv:1901.07675*, 2019.

[304] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

[305] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[306] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[307] François Mazé and Faez Ahmed. Diffusion models beat gans on topology optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 9108–9116, 2023.

[308] Giorgio Giannone, Akash Srivastava, Ole Winther, and Faez Ahmed. Aligning optimization trajectories with diffusion models for constrained design generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[309] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[310] Fabian Duddeck. Multidisciplinary optimization of car bodies. *Structural and Multidisciplinary Optimization*, 35:375–389, 2008.

[311] Kazuki Hayashi and Makoto Ohsaki. Reinforcement learning and graph embedding for binary truss topology optimization under stress and displacement constraints. *Frontiers in Built Environment*, 6:59, 2020.

[312] Hongbo Sun and Ling Ma. Generative design by using exploration approaches of reinforcement learning in density-based structural topology optimization. *Designs*, 4(2):10, 2020.

[313] Seowoo Jang, Soyoung Yoo, and Namwoo Kang. Generative design by reinforcement learning: enhancing the diversity of topology optimization designs. *Computer-Aided Design*, 146:103225, 2022.

[314] Leon Herrmann and Stefan Kollmannsberger. Deep learning in computational mechanics: a review. *Computational Mechanics*, pages 1–51, 2024.

[315] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[316] Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv preprint arXiv:1811.01900*, 2018.

[317] Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021.

[318] Jiequn Han, Xu-Hui Zhou, and Heng Xiao. An equivariant neural operator for developing nonlocal tensorial constitutive models. *Journal of Computational Physics*, 488:112243, 2023.

[319] Minkai Xu, Jiaqi Han, Aaron Lou, Jean Kossaifi, Arvind Ramanathan, Kamyar Azizzadenesheli, Jure Leskovec, Stefano Ermon, and Anima Anandkumar. Equivariant graph neural operator for modeling 3d dynamics. *arXiv preprint arXiv:2401.11037*, 2024.

[320] Chaoran Cheng and Jian Peng. Equivariant neural operator learning with graphon convolution. *arXiv preprint arXiv:2311.10908*, 2023.

[321] Jacob Helwig, Xuan Zhang, Cong Fu, Jerry Kurtin, Stephan Wojtowytsch, and Shuiwang Ji. Group equivariant fourier neural operators for partial differential equations. *arXiv preprint arXiv:2306.05697*, 2023.

[322] Xin Lei, Chang Liu, Zongliang Du, Weisheng Zhang, and Xu Guo. Machine learning-driven real-time topology optimization under moving morphable component-based framework. *Journal of Applied Mechanics*, 86(1):011004, 2019.

[323] Shuai Zheng, Zhenzhen He, and Honglei Liu. Generating three-dimensional structural topologies via a u-net convolutional neural network. *Thin-Walled Structures*, 159:107263, 2021.

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

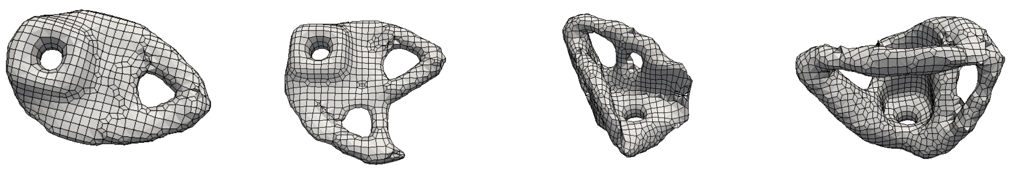
# Part II

# Publications

# SELTO Datasets

Sören Dittmer[1,2], David Erzmann[1], Henrik Harms[1], Rielson Falk[3], and Marco Gosch[3]

[1] *University of Bremen, Germany*
[2] *University of Cambridge, UK*
[3] *ArianeGroup GmbH*



(a) disc simple (b) disc complex (c) sphere simple (d) sphere complex

Figure 1: Ground truth examples from the SELTO disc and sphere dataset [1]. The densities are defined on a voxel grid and are smoothed for visualization purposes using Taubin smoothing [4].

## A Benchmark Dataset for Deep Learning for 3D Topology Optimization

The SELTO datasets [1] represent voxelized 3D topology optimization (TO) problems and solutions. They contain almost 10.000 three-dimensional TO problems and associated ground truth densities. The ground truths have been generated in cooperation with the Ariane Group[1] and Synera[2] using the Altair OptiStruct[3] implementation of SIMP within the Synera software.

The SELTO datasets comprise two distinct 3D datasets for TO, called *disc* and *sphere*, with the names referring to the shape of the respective design spaces. Both datasets can be split into subsets with load cases of one or two points of attack. We call these subsets *simple* and *complex*, respectively. All subsets are further split into a training and a validation dataset. For examples of ground truth densities see Figure 1. The SIMP method does not always provide a physically plausible solution for a TO problem, i.e., some solutions break under their load cases. Therefore, we clean both datasets after the dataset generation process by rejecting failing samples.

The individual samples are discretized on a $n_x \times n_y \times n_z$ voxel grid, where the choice of $n_x, n_y, n_z \in \mathbb{N}$ varies depending on the dataset. Both datasets have fixed Dirichlet boundary conditions but variable force positions and magnitudes. For individual specifications of both datasets, see Table 1.

| dataset | # samples | shape | subsets | # loads | # training samples | # validation samples |
|---|---|---|---|---|---|---|
| disc (combined) | 9246 | 39 × 39 × 4 voxels | simple | 1 | 1509 | 200 |
| | | | complex | 2 | 7337 | 200 |
| sphere (combined) | 602 | 39 × 39 × 21 voxels | simple | 1 | 150 | 36 |
| | | | complex | 2 | 380 | 36 |

Table 1: Overview of the two SELTO datasets *disc* and *sphere*.

[1] https://www.ariane.group
[2] https://www.synera.io
[3] https://altair.com/optistruct

Each TO problem is based on the following information [2]:

1. The number of voxels $(n_x, n_y, n_z)$ and the voxel size in millimeters in each direction.
2. Material properties, given by Young's modulus $E$, Poisson's ratio $\nu$ and a yield stress criterion $\sigma_{\text{ys}}$. We choose $E = 70$ GPa, $\nu = 0.3$ and $\sigma_{\text{ys}} = 450$ MPa for both datasets.
3. A binary $(3 \times n_x \times n_y \times n_z)$-tensor $\Omega_{\text{Dirichlet}}$ to encode the presence of directional homogeneous Dirichlet boundary conditions for every voxel. 1s indicate the presence, and 0s the absence of homogeneous Dirichlet boundary conditions.
4. A real-valued $(3 \times n_x \times n_y \times n_z)$-tensor $F$ to encode external forces, given in N/m$^3$. The three channels correspond to the force magnitudes in each spatial dimension.
5. A $(1 \times n_x \times n_y \times n_z)$-tensor $\Omega_{\text{design}}$ containing values $\in \{-1, 0, 1\}$ to encode design space information. We use 0s and 1s to constrain voxel densities to be 0 or 1, respectively. Entries of $-1$s indicate a lack of density constraints, which signifies that the density in that voxel can be freely optimized. For voxels that have Dirichlet boundary conditions or loads assigned to them, we enforce the density value to be 1 by setting $\Omega_{\text{design}} = 1$.

All tensors are defined voxel-wise, including $\Omega_{\text{Dirichlet}}$ and $F$. This makes our datasets easy to use in DL applications as it allows for a shape-consistent tensor representation. The SELTO datasets can be accessed via Zenodo[4] [1] and the Python library DL4TO[5] [3] can be used to download and access all individual SELTO dataset subsets.

## References

[1] S. Dittmer, D. Erzmann, H. Harms, R. Falck, and M. Gosch. Selto dataset. https://doi.org/10.5281/zenodo.7034898, 2023.

[2] S. Dittmer, D. Erzmann, H. Harms, and P. Maass. Selto: Sample-efficient learned topology optimization. *arXiv preprint arXiv:2209.05098*, 2022.

[3] D. Erzmann, S. Dittmer, H. Harms, and P. Maaß. Dl4to: A deep learning library for sample-efficient topology optimization. In *International Conference on Geometric Science of Information*, pages 543–551. Springer, 2023.

[4] G. Taubin. Curve and surface smoothing without shrinkage. In *Proceedings of IEEE international conference on computer vision*, pages 852–857. IEEE, 1995.

[4] https://doi.org/10.5281/zenodo.7034898
[5] https://github.com/dl4to/dl4to

# DL4TO: A Deep Learning Library for Sample-Efficient Topology Optimization

David Erzmann[1], Sören Dittmer[1,2], Henrik Harms[1], and Peter Maaß[1]

[1] Center for Industrial Mathematics, University of Bremen, Germany
[2] Cambridge Image Analysis, Centre for Mathematical Sciences, University of Cambridge, UK

**Abstract.** We present and publish the DL4TO software library – a Python library for three-dimensional topology optimization. The framework is based on PyTorch and allows easy integration with neural networks. The library fills a critical void in the current research toolkit on the intersection of deep learning and topology optimization. We present the structure of the library's main components and how it enabled the incorporation of physics concepts into deep learning models.

**Keywords:** Topology optimization · Deep learning · Software library.

## 1 Introduction

We begin by briefly introducing the problem of Topology Optimization (TO) and the recent development of applying deep learning to it.

### 1.1 Classical Topology Optimization

The computational discipline of topology optimization (TO) aims to optimize mechanical structures. Since its development in 1988 [4], TO is a powerful tool widely adopted by engineers in a variety of fields, e.g., fluid [6] and solid mechanics [15], acoustics [14,29], and heat transfer [10].

The *Solid Isotropic Material with Penalization* (SIMP) method [5] is widely regarded as the most significant classical approach used in TO. SIMP involves a density-based setup where the density takes values between 0 and 1 over a given design domain. The density represents to which degree material is present in different places. SIMP first voxelizes the domain and density and then employs an iterative optimization scheme to improve structural performance by adjusting voxel densities. In the case of linear elasticity, one evaluates the integrity of the resulting structure via *von Mises stresses*, whose computation involves the corresponding partial differential equation (PDE). The specified objective function and constraints may vary depending on the user's needs. The most common setup for mechanical problems is *compliance minimization* [7], where we minimize a compliance objective subject to volume and possibly stress constraints. See Algorithm 1 for a pseudo-code representation of our SIMP implementation. For more details on SIMP see [5].

**Algorithm 1** Our implementation of the *Solid Isotropic Material with Penalization* (SIMP) algorithm for compliance minimization with volume and stress constraints. We typically initialize $\theta_0$ to be 0.5 everywhere, where it is not enforced otherwise. For the filtering we apply a smoothed Heaviside function $H_\beta(\theta) := 1 - \exp(-\beta\theta) + \theta \exp(-\beta)$ with a smoothing factor $\beta > 0$ which we gradually steepen over the iterations. The SIMP exponent $p$ is commonly chosen as $p = 3$ to further discourage non-binary solutions.

```
  Require: $F$, $\sigma_{ys}$, $N$, $\lambda$, $\mu$        ▷ Forces, yield stress, #iterations and loss weights
  Initialize: $\theta_0$                                    ▷ Start with an initial density
  Set: $p = 3$                                              ▷ Set the SIMP exponent to its typical value
  for $i = 0, \dots, N-1$ do
      $\theta_i \leftarrow$ project($\theta_i$)             ▷ Project density values into the unit interval
      $\theta_i \leftarrow$ smooth($\theta_i$)              ▷ Avoid checkerboard patterns via smoothing
      $\theta_i \leftarrow$ filter($\theta_i$)              ▷ Encourage binary densities via filtering
      $u, \sigma_{vM}$ = pde_solver($F$, ${\theta_i}^p$)    ▷ Solve PDE for current density and exponent $p$
      loss_compl = $F^T u$                                  ▷ Compliance with forces $F$ and displacements $u$
      loss_vol = $\|\theta_i\|_1$                           ▷ Compute volume loss term
      loss_stress = softplus($\sigma_{vM} - \sigma_{ys}$)   ▷ Compute stress constraint loss term
      loss = loss_compl + $\lambda \cdot$ loss_vol + $\mu \cdot$ loss_stress   ▷ Sum up and weight losses
      $\theta_{i+1} \leftarrow$ gradient_step($\theta_i$, loss)   ▷ Update density via gradient descent
  end for
return $\theta_N$                                           ▷ Return final density distribution
```

### 1.2 Neural Networks for Topology Optimization

The iterative nature of density-based methods like SIMP requires repeated solving of the governing PDE. This becomes computationally prohibitive for high voxel mesh resolutions, leading to practical limitations [1]. Recent Deep Learning (DL) research has explored overcoming this challenge. One can broadly classify the advances into four categories [33]:

1. Reduce SIMP iterations: Neural networks map from intermediate SIMP iterations to the final structure, technically performing a deblurring task [2,3,26,28].
2. Eliminate SIMP iterations: Neural networks directly predict the final density distribution without performing any SIMP iterations [21,30,32].
3. Substitute for PDE solver: One replaces classical PDE solvers with neural networks, removing the primary bottleneck [8,20,24].
4. Neural reparameterization: One uses neural networks to reparameterize the density function [11,17,31,33]. However, one usually still requires computationally demanding PDE evaluations for the training.

As advances in computational power and DL have only recently brought the application of DL to TO in the realm of the possible, the literature on it is still in its infancy. As a result, the authors are not aware of any public software framework for TO using DL, requiring every researcher to write their code from scratch.

We want to address this issue by presenting DL4TO, a flexible and easy-to-use python library for three-dimensional TO. The library is open source and based on PyTorch [22], allowing for easy integration of DL and TO methods.

## 2 The DL4TO Framework

### 2.1 Motivation

In this section, we give a basic overview of the DL4TO[3] library. The primary motivation for developing DL4TO is the need for a flexible and easy-to-use basis to conduct DL experiments for TO in Python. The library focuses on linear elasticity on structured three-dimensional grids. DL4TO comes with its own PDE solver, a SIMP implementation, and various objective functions for classical and learned TO. The PyTorch [22]-based implementation smoothly connects the world of TO with the world of DL. To our knowledge, only two Python libraries for TO [16,18] exist, and neither allows for easy integration with neural networks.

### 2.2 Core Classes

In the following we give an overview of how our framework works. Below, we introduce the three main classes that form the core of our library.

- Problem: An important novelty of our framework is how TO problems are defined and processed. This is done via the Problem class, which contains all information of the underlying TO problem one intends to solve. Since we perform optimization on structured grids, all information is either in scalar or in tensor form. This makes data compatible with DL applications since it allows for a shape-consistent tensor representation. Let $(n_x, n_y, n_z)$ be the number of voxels in each spacial direction. We can create a uniquely characterized problem object via

  ```
  problem = Problem(E, ν, σ_ys, h, F, Ω_dirichlet, Ω_design).
  ```

  Here,
  - E, ν and σ_ys denote scalar material properties, namely Young's modulus, Poisson's ratio and yield stress.
  - h is a three-dimensional vector that defines the voxel sizes in meters in each direction.
  - F is a $(3 \times n_x \times n_y \times n_z)$-tensor which encodes external forces given in $\mathrm{N/m}^3$. The three channels correspond to the force magnitudes in each spacial dimension.
  - Ω_dirichlet is a binary $(3 \times n_x \times n_y \times n_z)$-tensor which we use to encode the presence of directional homogeneous Dirichlet boundary conditions for every voxel. 1s indicate the presence, and 0s the absence of homogeneous Dirichlet boundary conditions. Currently, we do not support non-homogeneous Dirichlet boundary conditions since we believe that they are not required for most TO tasks.

[3] The DL4TO library is publicly available at https://github.com/dl4to/dl4to.

  - `Ω_design` is a $(1 \times n_x \times n_y \times n_z)$-tensor containing values $\in \{0, 1, -1\}$ that we use to encode design space information. We use 0s and 1s to constrain voxel densities to be 0 or 1, respectively. Entries of $-1$ indicate a lack of density constraints, which signifies that the density in that voxel can be freely optimized. For voxels that have loads assigned to them we automatically enforce the corresponding density value to be 1.

- `TopoSolver`: This parent class provides different methods for solving TO problems. SIMP, as well as learned methods, are child classes. The initialization arguments slightly differ, depending on the method used. For instance, a SIMP solver for compliance minimization with a volume constraint can be initialized via

  ```
  criterion = Compliance() + λ * VolumeConstraint(vol_fract)
  topo_solver = SIMP(criterion, p, n_iters, lr)
  ```

  with some arbitrary scalar choice of volume fraction `vol_fract` and optimization weight `λ`. The other arguments of `SIMP` denote the SIMP exponent choice `p` (by default, `p` = 3), the number of iterations `n_iters` and the learning rate `lr`.
  Alternatively, for volume minimization with a stress constraint we could set the optimization criterion as follows:

  ```
  criterion = VolumeFraction() + λ * StressConstraint().
  ```

  By default, our framework uses a built-in finite differences method (FDM) solver whenever the PDE for linear elasticity is solved. This is attributed to the regular grid structure, which makes the FDM a suitable and intuitive approach. It is however also possible to include custom PDE solvers, e.g., learned PDE solvers.
  In order to apply a `topo_solver` to a predefined `problem` object, we can simply call it via

  ```
  solution = topo_solver(problem),
  ```

  which returns a `solution` object. Note that this also works with a list of problems, in which case `topo_solver` likewise returns a list of solutions.
  For learned solvers the procedure is similar, with the exception that the initialization of the `topo_solver` object additionally requires a preprocessing as input. This determines how a `problem` object should be converted to neural network compatible input tensors when calling the solver (see Section 4).
  The `topo_solver` is trained via the built-in `train` function:

  ```
  topo_solver.train(dataloader_train, dataloader_val, epochs),
  ```

  where `dataloader_train` and optionally `dataloader_val` are dataloaders for the training and validation dataset.

- `Solution`: Objects of this class define solutions to TO problems. They usually result from calling a `topo_solver` with a `problem` object, but can also be instantiated manually by passing a problem and a density distribution:

```
solution = Solution(problem, θ).
```

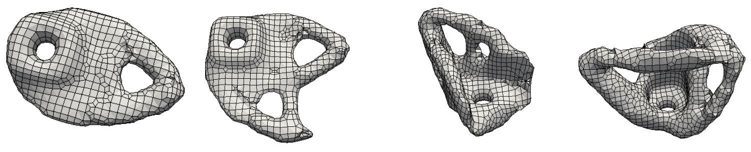
Here, θ is a $(1 \times n_x \times n_y \times n_z)$-tensor that defines a three-dimensional density distribution that solves `problem`. The `Solution` class provides several useful functionalities like logging and plotting.

## 3 Datasets

`DL4TO` is compatible with the SELTO datasets [13] introduced in [12] and publicly available at https://doi.org/10.5281/zenodo.7034898. We want to give a short overview of the two SELTO datasets containing samples of mechanical mounting brackets. Each dataset consists of tuples (`problem`, `solution`), where `solution` is a ground truth density distribution for `problem`. The two datasets are called *disc* and *sphere*, referring to the shape of the corresponding design spaces; see Table 1 for an overview of both datasets and Figure 1 for example samples.

| dataset | shape | # samples |
|---|---|---|
| disc | $39 \times 39 \times 4$ voxels | 9246 (8846 train, 400 val) |
| sphere | $39 \times 39 \times 21$ voxels | 602 (530 train, 72 val) |

Table 1: Overview of our datasets, called *disc* and *sphere*, with the names referring to the shape of their design spaces. Both datasets are split into a training and a validation subset.



(a) Disc dataset (b) Sphere dataset

Fig. 1: Ground truth examples from the disc and sphere dataset [13]. The densities are defined on a voxel grid and smoothed for visualization using Taubin smoothing [27].

## 4 Model Pipeline

We now present how `DL4TO` enables efficient setup of data pipelines. `DL4TO` provides different models and loss functions, e.g., UNets [25] and the weighted binary cross-entropy. Due to the integration with `PyTorch`, optimizers like Adam [19] can be used plug-and-play.

We begin with two critical components of `DL4TO`'s model pipeline: physics-based preprocessing and equivariant architectures, see [12].

- Preprocessing: A suitable input preprocessing strategy is crucial for DL. Following [12], the library provides two preprocessing strategies, easily combined via channel-wise concatenation.
  1. Trivial preprocessing: For each `problem` object, the input of the neural network is a 7-channel tensor which results from the channel-wise concatenation of `Ω_dirichlet`, `Ω_design`, and normalized loads `F`.
  2. PDE preprocessing [12] [32]: We set up a density distribution that is 1 wherever allowed by `problem`. We compute normalized von Mises stresses for that density by solving the PDE for linear elasticity. We use the von Mieses stresses as a 1-channel input to the neural network.
- Equivariance: Equivariance is the property of a function to commute with the actions of a symmetry group. For a given transformation group $G$, a function $f : X \to Y$ is ($G$-)equivariant if

$$f(T_g^X(x)) = T_g^Y(f(x)) \qquad\qquad \forall g \in G,\, x \in X,$$

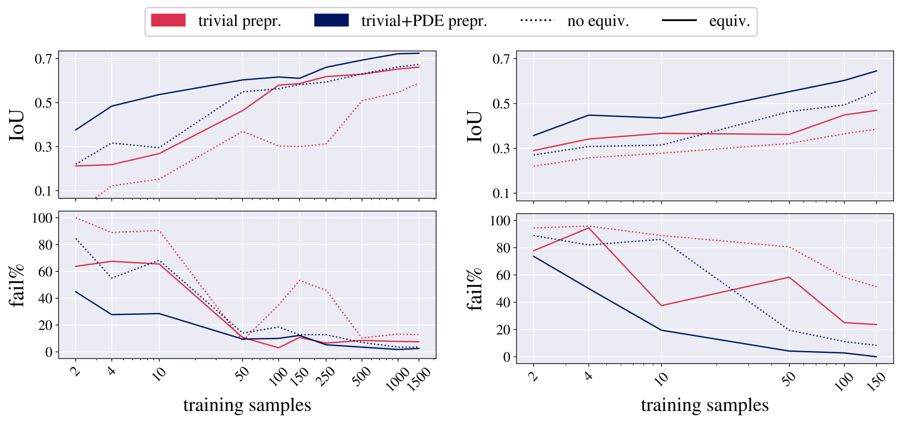
  where $T_g^X$ and $T_g^Y$ denote linear *group actions* in the corresponding spaces $X$ and $Y$ [9]. As shown by [12], mirror and rotation equivariance can drastically improve model performance on TO tasks. We implement equivariance via *group averaging* [23] by defining an equivariance wrapper $F_G^f$ via

$$F_G^f(x) := \frac{1}{|G|} \sum_{g \in G} T_{g^{-1}}^Y \left[ f(T_g^X(x)) \right].$$

  The plug-and-play nature of $F_G^f$ allows effortless applicability to any finite transformation group $G$ and any model $f$.

## 5 Experiments

Generating large datasets is costly; therefore, reducing the required training samples, e.g., by modifying the DL model design, is highly beneficial. Using the `DL4TO` library, [12] investigates the *sample efficiency* of models, i.e., the model's performance when trained on a few training samples. They visualize the sample efficiency of a model via so-called *sample efficiency curves* (SE curves). Each SE curve uses separate instances of a given model setup trained on subsets of the original training dataset of increasing size. One then determines the performance of these models on a fixed validation dataset, e.g., via *Intersection over Union* (IoU) and *fail%* (the percentage of failing model predictions). Figure 2 shows [12]'s results and presents dramatic boosts in the UNet's performance when incorporating physics via equivariance and trivial+PDE preprocessing.



(a) Disc dataset (b) Sphere dataset

Fig. 2: Sample efficiency curves from [12], trained and evaluated via `DL4TO` on the SELTO datasets *disc* (left) and *sphere* (right). The horizontal-axis shows the size of the training dataset for the different models on a logarithmic scale. The vertical-axis shows the performance of the criteria IoU and fail%. Each of the plots shows four different models based on trivial preprocessing (red), trivial+PDE preprocessing (blue), equivariance (solid line), and no equivariance (dashed line).

These improvements are especially visible for low numbers of training samples. For a more in-depth analysis and comparison of the proposed modeling approaches, we refer to [12].

## 6 Conclusion

We presented the `DL4TO` library. The Python library enables research in the intersection of topology optimization and deep learning. Seamlessly integrating with `PyTorch`, `DL4TO` enables a smooth interaction between deep learning models and established algorithms like SIMP. Further, `DL4TO` provides concepts like UNets, SIMP, equivariance, differentiable physics via finite difference analysis, integration with the SELTO datasets [13], as well as three-dimensional interactive visualization. Our library is especially useful for data scientists who want to apply deep learning to topology optimization, as it provides a flexible yet easy-to-use framework. `DL4TO` will continue to be expanded, and the community is welcome to contribute. Documentation and tutorials can be found at https://dl4to.github.io/dl4to/, providing a guide on how to use the library and its features.

## References

1. Aage, N., Andreassen, E., Lazarov, B.S.: Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework. Structural and Multidisciplinary Optimization **51**(3), 565–572 (2015)
2. Abueidda, D.W., Koric, S., Sobh, N.A.: Topology optimization of 2d structures with nonlinearities using deep learning. Computers & Structures **237**, 106283 (2020)
3. Banga, S., Gehani, H., Bhilare, S., Patel, S., Kara, L.: 3d topology optimization using convolutional neural networks. arXiv preprint arXiv:1808.07440 (2018)
4. Bendsøe, M.P., Kikuchi, N.: Generating optimal topologies in structural design using a homogenization method. Computer methods in applied mechanics and engineering **71**(2), 197–224 (1988)
5. Bendsoe, M.P., Sigmund, O.: Topology optimization: theory, methods, and applications. Springer Science & Business Media (2003)
6. Borrvall, T., Petersson, J.: Topology optimization of fluids in stokes flow. International journal for numerical methods in fluids **41**(1), 77–107 (2003)
7. Buhl, T., Pedersen, C.B., Sigmund, O.: Stiffness design of geometrically nonlinear structures using topology optimization. Structural and Multidisciplinary Optimization **19**, 93–104 (2000)
8. Chi, H., Zhang, Y., Tang, T.L.E., Mirabella, L., Dalloro, L., Song, L., Paulino, G.H.: Universal machine learning for topology optimization. Computer Methods in Applied Mechanics and Engineering **375**, 112739 (2021)
9. Cohen, T., Welling, M.: Group equivariant convolutional networks. In: International conference on machine learning. pp. 2990–2999. PMLR (2016)
10. Dede, E.M.: Multiphysics topology optimization of heat transfer and fluid flow systems. In: proceedings of the COMSOL Users Conference. vol. 715 (2009)
11. Deng, H., To, A.C.: Topology optimization based on deep representation learning (drl) for compliance and stress-constrained design. Computational Mechanics **66**(2), 449–469 (2020)
12. Dittmer, S., Erzmann, D., Harms, H., Maass, P.: Selto: Sample-efficient learned topology optimization. arXiv preprint arXiv:2209.05098 (2022)
13. Dittmer, S., Erzmann, D., Harms, H., Falck, R., Gosch, M.: Selto dataset (2023). https://doi.org/10.5281/zenodo.7034898
14. Dühring, M.B., Jensen, J.S., Sigmund, O.: Acoustic design by topology optimization. Journal of sound and vibration **317**(3-5), 557–575 (2008)
15. Eschenauer, H.A., Olhoff, N.: Topology optimization of continuum structures: a review. Appl. Mech. Rev. **54**(4), 331–390 (2001)
16. Ferguson, Z.: Topopt — topology optimization in python. https://github.com/zfergus/topopt (2019)
17. Hoyer, S., Sohl-Dickstein, J., Greydanus, S.: Neural reparameterization improves structural optimization. arXiv preprint arXiv:1909.04240 (2019)
18. Hunter, W., et al.: Topy - topology optimization with python. https://github.com/williamhunter/topy (2017)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Lee, S., Kim, H., Lieu, Q.X., Lee, J.: Cnn-based image recognition for topology optimization. Knowledge-Based Systems **198**, 105887 (2020)
21. Nie, Z., Lin, T., Jiang, H., Kara, L.B.: Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain. Journal of Mechanical Design **143**(3) (2021)

22. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
23. Puny, O., Atzmon, M., Ben-Hamu, H., Smith, E.J., Misra, I., Grover, A., Lipman, Y.: Frame averaging for invariant and equivariant network design. arXiv preprint arXiv:2110.03336 (2021)
24. Qian, C., Ye, W.: Accelerating gradient-based topology optimization design with dual-model artificial neural networks. Structural and Multidisciplinary Optimization **63**(4), 1687–1707 (2021)
25. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
26. Sosnovik, I., Oseledets, I.: Neural networks for topology optimization. Russian Journal of Numerical Analysis and Mathematical Modelling **34**(4), 215–223 (2019)
27. Taubin, G.: Curve and surface smoothing without shrinkage. In: Proceedings of IEEE international conference on computer vision. pp. 852–857. IEEE (1995)
28. Xue, L., Liu, J., Wen, G., Wang, H.: Efficient, high-resolution topology optimization method based on convolutional neural networks. Frontiers of Mechanical Engineering **16**(1), 80–96 (2021)
29. Yoon, G.H., Jensen, J.S., Sigmund, O.: Topology optimization of acoustic–structure interaction problems using a mixed finite element formulation. International journal for numerical methods in engineering **70**(9), 1049–1075 (2007)
30. Yu, Y., Hur, T., Jung, J., Jang, I.G.: Deep learning for determining a near-optimal topological design without any iteration. Structural and Multidisciplinary Optimization **59**(3), 787–799 (2019)
31. Zehnder, J., Li, Y., Coros, S., Thomaszewski, B.: Ntopo: Mesh-free topology optimization using implicit neural representations. Advances in Neural Information Processing Systems **34**, 10368–10381 (2021)
32. Zhang, Y., Peng, B., Zhou, X., Xiang, C., Wang, D.: A deep convolutional neural network for topology optimization with strong generalization ability. arXiv preprint arXiv:1901.07761 (2019)
33. Zhang, Z., Li, Y., Zhou, W., Chen, X., Yao, W., Zhao, Y.: Tonr: An exploration for a novel way combining neural network with topology optimization. Computer Methods in Applied Mechanics and Engineering **386**, 114083 (2021)

# SELTO: SAMPLE-EFFICIENT LEARNED TOPOLOGY OPTIMIZATION[*]

SÖREN DITTMER[†‡§], DAVID ERZMANN[†§], HENRIK HARMS[†], AND PETER MAASS[†]

**Abstract.** Recent developments in Deep Learning (DL) suggest a vast potential for Topology Optimization (TO). However, while there are some promising attempts, the subfield still lacks a firm footing regarding basic methods and datasets. We aim to address both points. First, we explore physics-based preprocessing and equivariant networks to create sample-efficient components for TO DL pipelines. We evaluate them in a large-scale ablation study using end-to-end supervised training. The results demonstrate a drastic improvement in sample efficiency and the predictions' physical correctness. Second, to improve comparability and future progress, we publish the two first TO datasets containing problems and corresponding ground truth solutions.

**Key words.** topology optimization, deep learning, inverse problems

**MSC codes.** 65N21, 68T01, 68U05, 68U07

## 1. Introduction.

The computational discipline of Topology Optimization (TO) generates mechanical structures. Increased computational power made TO an integral tool for engineers in fields ranging from heat transfer [14] and acoustics [17, 51] to fluid [7] and solid mechanics [19]. Still, TO remains computationally costly and often time-consuming [1, 25]. Recently, Deep Learning (DL) approaches tried to address this. While the approaches are promising, the subfield lacks foundations. This paper aims to establish these foundations, focusing on linear elasticity problems.

The lack of foundations shows most in two places. First, we see a need for more involvement of physics priors in the DL pipeline and the evaluation of their efficacy. Second, to the authors' knowledge, no publicly available TO datasets exist. The release of public datasets often proved to be the deciding spark for the flowering of DL subfields. Here we aim to address both points.

We will now give a high-level overview of the classical SIMP [4] method to frame the problem DL methods try to solve. *Solid Isotropic Material with Penalization* (SIMP) [4], arguably the most important classical TO method, uses a density-based setup. Here one discretizes a given design domain into voxels, each having a density value between 0 and 1. These values represent the amount of material in the voxel, thereby defining a mechanical structure. One employs the partial differential equation (PDE) governing the physical phenomenon of interest to determine the performance based on specified constraints and cost functions. One then adjusts the voxel densities via iterative optimization methods to improve performance. From a mathematical perspective, this constitutes a PDE parameter identification problem.

While much progress has been made, classical methods' iterative nature and the in resolution superlinear computational PDE-cost makes classical methods highly computationally demanding – often to the point of practical impossibility [1]. Recent research tries to overcome these challenges using deep learning (DL), i.e., neural networks [5], to speed up and improve the optimization process.

[*]The authors would like to thank the Federal Ministry for Economic Affairs and Climate Action of Germany (BMWK) and the German Aerospace Center (DLR) Space Agency for supporting this work (grant no. 50 RL 2060)

[†]Center for Industrial Mathematics, University of Bremen, Germany (sdittmer@math.uni-bremen.de, erzmann@uni-bremen.de, hharms@uni-bremen.de, pmaass@uni-bremen.de)

[‡]Cambridge Image Analysis, Centre for Mathematical Sciences, University of Cambridge, UK

[§]Equal contribution

While these DL methods can often solve TO problems in less than a second, they still lack a solid foundation regarding architecture, data preprocessing, evaluation criteria, benchmarks, and datasets. Overall, the subdiscipline of applying DL to TO is in its infancy, with most papers focusing on two-dimensional settings [28, 32, 33, 37, 44, 47, 54, 55]. In particular, the lack of datasets drastically limits the evaluation and comparability of new approaches.

This paper establishes and compares a set of tools for DL-based TO, including the choice of network architecture, data preprocessing techniques, and the incorporation of physical priors. The paper also publishes two three-dimensional datasets containing almost 10 000 TO problems and corresponding solutions.

As in DL, not only the amount but also the similarity of the training data to one's problem plays a crucial role; we also study the so-called *generalization* [26] of our tools. Generalization is critical for TO, as large-scale training data generation costs can be prohibitive.

Our main contributions are:

- We develop and evaluate physical priors in the DL pipeline, e.g., PDE-based preprocessing and equivariant models.
- We provide a large-scale ablation study to assess the sample-efficiency of different architectures and preprocessings, i.e., we study the efficacy in the small data setting.
- We provide the first publicly available TO datasets containing problems and corresponding ground truth solutions.

We believe the publication of datasets will improve the field's comparability and incorporating physical laws into the data pipeline marks a significant step toward real-world applicability.

## 2. Preliminaries and related work.

We now briefly introduce classical TO, followed by a review of the current literature on DL for TO.

### 2.1. Density-based topology optimization.

We start by discussing density-based TO, arguably the most common classical TO approach – also used to generate our datasets (see Section 3).

Density-based TO [4] aims to minimize a cost or objective function by adjusting the material's density distribution $\rho : \Omega \to \{\rho_{\min}, 1\}$ over a fixed domain $\Omega \subset \mathbb{R}^d$, typically $d \in \{2, 3\}$. Here, $0 < \rho_{\min} \ll 1$ defines a minimal density value. While the final objective is to obtain binary densities $\rho(\cdot) \in \{0, 1\}$, setting $\rho_{\min} > 0$ is a numerical necessity for solving the governing PDE. Additionally, this optimization is subject to physical constraints. The specified objective function and constraints may vary depending on the user's needs.

This paper focuses on compliance minimization, the most common setting for mechanical problems. The corresponding optimization problem reads as follows:

$$\min_{\rho} \quad F^T u(\rho) \tag{2.1a}$$

$$\text{subject to} \quad K(\rho)u = F, \tag{2.1b}$$

$$\|\rho\|_1 \leq V_{\max}, \tag{2.1c}$$

$$\sigma_{\mathrm{vM}}(u) \leq \sigma_{\mathrm{ys}}. \tag{2.1d}$$

Here (2.1a) is the compliance objective function, $F$ represents the global load distribution, $u$ are the displacement and $K$ is the symmetric positive operator of linear elasticity. $K$ includes the characteristic properties of the used material described by

Young's modulus and Poisson's ratio, which we denote by $E \in \mathbb{R}$ and $\nu \in [0, 0.5]$, respectively. One constrains the amount of allowed material by $V_{\max}$ (2.1c), and one may include a stress constraint (2.1d). This is done to ensure that the maximal von Mises stress $\sigma_{\mathrm{vM}}$ is below the yield stress $\sigma_{\mathrm{ys}}$ of the material. The von Mises stresses are used to predict mechanical yielding and are derived non-linearly from the displacements $u$.

In practice, the SIMP [4] method relaxes the problem. Instead of strictly binary density values, one allows $\rho : \Omega \to [\rho_{\min}, 1]$ and encourages near-binary densities by extending Young's modulus over $\rho$'s interval via $E(\rho) = E_0\rho^p$, where $E_0$ is the original (isotropic) material's modulus. The exponent $p$ controls the penalization of non-binary densities, usually $p = 3$. For implementation purposes, one discretizes the design space $\Omega$ into a regular voxel grid and iteratively updates $\rho$ until a user-specified convergence criterion is met.

As discussed in the introduction, despite several advancements in structural TO, one of the main challenges is the high computational cost. The main bottleneck in classical iterative approaches is that each iteration uses the displacements and stresses for the current density and therefore has to solve the PDE for linear elasticity in (2.1b). Due to this computational challenge, TO at high resolutions, i.e., over many voxels, can take hours, even days [1]. This inspired researchers to develop DL-based TO methods to reduce or eliminate the need to solve PDEs.

**2.2. Neural networks for topology optimization.** One can broadly classify advances in TO using DL into three categories [55].

**Reduce or eliminate SIMP iterations:** The first serious attempts of TO via DL aimed to reduce the number of SIMP iterations [3, 44, 50]. In 2017, Sosnovik et al. [44] interpreted two-dimensional TO problems as image-to-image regression problems and were the first to apply *convolutional neural networks* (CNNs) to TO. Following well-known image processing approaches, they trained a UNet model [41] to map from intermediate SIMP iterations to the final structure. In 2018, Banga et al. [3] transferred these ideas to the three-dimensional case. Xue et al. [50] made each SIMP iteration cheaper by running it on a coarse resolution. They then applied a DL-based super-resolution method to increase the structure's final granularity. In 2020, Abueidda et al. [2] were the first to use *residual neural networks* (ResNets) [56] and to consider two-dimensional nonlinear elasticity.

In 2019, Yu et al. [52] developed the first end-to-end learning routine that directly predicts the final density without performing any SIMP iterations. They also created a generative framework to increase the resolution of their predicted designs. This formed the basis for a series of publications [32, 33, 40, 43] on *generative adversarial network* (GAN)-based TO algorithms [21].

Compared to previous research, Nie et al. [32] and Zhang et al. [54] achieved a better generalization by not directly giving the network the boundary conditions as input. Instead, they passed displacements and von Mises stresses into the network. They argue that neural networks have difficulties extending to previously unseen boundary conditions if the input data is very sparse since the high sparsity of the input matrices leads to high variance of the mapping function.

**Substitute SIMP's PDE solver:** These methods aim to remove classical PDE solvers from the SIMP algorithm, removing its primary bottleneck. Senhora et al. [42] solved the PDE on a coarse grid and learned an upscaling to higher resolutions. Qian et al. [37] proposed a dual-model neural network using a forward model to compute the compliance of the structure and an adjoint model to determine the derivatives with respect to the density of each voxel. Similarly, Chi et al. [11] and Lee et al. [28] used neural networks to replace the gradient and objective function computation.

**Neural reparameterization:** Using implicit neural representation for complex signals is an ongoing research topic in computer vision [10] and engineering [30]. Several TO publications [15, 23, 53, 55] feature neural networks to reparameterize the TO's density field. While Hoyer et al. [23] mapped latent vectors to discrete grid densities, Chandrasekhar & Suresh [9] used multilayer perceptrons to learn a continuous mapping from spatial locations to density values. Since these models are mesh-independent, they can represent the density function at arbitrary resolutions. However, one usually still requires PDE evaluations for training, which is computationally demanding.

While most DL approaches to TO are entirely oblivious to the underlying physics, there is a small number of approaches that have started to include physics-inspired properties into the training process: Banga et al. [3] and Rade et al. [38] augmented the dataset by including rotations and mirrors of given loads and boundary conditions to encourage equivariance. Nie et al. [32] and Zhang et al. [56] involved physics by feeding the network strain and stress information as inputs. Cang et al. [8] and Zhang et al. [55] introduced physics via the loss function design, comparable to *physics-informed neural networks* (PINNs) [39].

To our knowledge, no literature incorporated the problem's underlying physics by modifying the architecture of the DL model itself. We demonstrate that we can dramatically improve *sample efficiency*, i.e., the model's performance when trained on a few training samples, and facilitate geometric reasoning by restricting the hypothesis space to group equivariant models. Cohen & Welling [13] introduced the first group equivariant CNNs in 2016. Nowadays, their application ranges from chemistry [48] and physics [6] to a wide range of tasks in geometric DL [46, 20]. For image processing tasks, Dumont et al. [18] showed that enforcing relevant equivariances can improve generalization performance and

## 3. Datasets.

We find a substantial lack in the availability of public three-dimensional TO datasets, i.e., to the best of our knowledge there is none. This is problematic for several reasons. First, it is hard to reproduce other published results. Second, it requires each researcher to generate their own datasets, which is time-consuming and computationally demanding. Lastly, the lack of established TO datasets impedes the comparability of results throughout the community. To alleviate this issue, we publish two three-dimensional TO datasets containing samples of mounting brackets, which we call *disc dataset* and *sphere dataset*, refering to the shape of their respective design spaces. Both datasets are publicly available at https://doi.org/10.5281/zenodo.7034898 [16].

Each dataset consists of TO problems and associated ground truth density distributions. We generated the samples in cooperation with the ArianeGroup and Synera using the Altair OptiStruct implementation of SIMP within the Synera software. The ArianeGroup designed the mounting brackets in the datasets to be of practical use, though real-world aerospace applications would require more complex load cases. The samples are discretized on a $n_x \times n_y \times n_z$ voxel grid, where the choice of $n_x, n_y, n_z \in \mathbb{N}$ varies depending on the dataset. Both datasets have fixed Dirichlet boundary conditions but variable force positions and magnitudes.

| dataset | # samples | shape | subsets | # loads | # training samples | # validation samples |
|---|---|---|---|---|---|---|
| disc (combined) | 9246 | $39 \times 39 \times 4$ voxels | simple | 1 | 1509 | 200 |
| | | | complex | 2 | 7337 | 200 |
| sphere (combined) | 602 | $39 \times 39 \times 21$ voxels | simple | 1 | 150 | 36 |
| | | | complex | 2 | 380 | 36 |

Table 1: Overview of our datasets, called *disc* and *sphere*, with the names referring to the shape of their design spaces. Both datasets can be split into subsets with load cases of one or two points of attack. We call these subsets *simple* and *complex*, respectively. When we refer to the mixed datasets, i.e., the combination of simple and complex, we sometimes call these *disc combined* and *sphere combined.*

One can uniquely characterize each TO problem via the following properties:

1. The number of voxels $(n_x, n_y, n_z)$ and the voxel size in millimeters in each direction.
2. Material properties, given by Young's modulus $E$, Poisson's ratio $\nu$ and a yield stress criterion $\sigma_{\text{ys}}$. We choose $E = 70$ GPa, $\nu = 0.3$ and $\sigma_{\text{ys}} = 450$ MPa for both datasets.
3. A binary $(3 \times n_x \times n_y \times n_z)$-tensor $\omega_{\text{Dirichlet}}$ to encode the presence of directional homogeneous Dirichlet boundary conditions for every voxel. 1s indicate the presence, and 0s the absence of homogeneous Dirichlet boundary conditions.
4. A real-valued $(3 \times n_x \times n_y \times n_z)$-tensor $F$ to encode external forces, given in N/m$^3$. The three channels correspond to the force magnitudes in each spacial dimension.
5. A $(1 \times n_x \times n_y \times n_z)$-tensor $\omega_{\text{design}}$ containing values $\in \{-1, 0, 1\}$ to encode design space information. We use 0s and 1s to constrain voxel densities to be 0 or 1, respectively. Entries of $-1$s indicate a lack of density constraints, which signifies that the density in that voxel can be freely optimized. This naturally defines the voxel sets $\Omega_{-1}, \Omega_0$, and $\Omega_1$. For voxels that have Dirichlet boundary conditions or loads assigned to them we enforce the density value to be 1 by setting $\omega_{\text{design}} = 1$.

All tensors are defined voxel-wise, including $\omega_{\text{Dirichlet}}$ and $F$. This makes our datasets easy to use in DL applications as it allows for a shape-consistent tensor representation.

The SIMP method does not always provide a physically plausible solution for a TO problem, i.e., some solutions break under their load cases. Therefore, we clean both datasets after the dataset generation process by rejecting failing samples. This leaves a total count of almost 10 000 problem-ground truth pairs. Both datasets can be split into subsets with load cases of one or two points of attack. We call these subsets *simple* and *complex*, respectively. We refer to the combination of the simple and complex dataset as the *disc combined* and *sphere combined* dataset.

See Table 1 for an overview of both datasets and Figure 1 for ground truth examples. More samples can be found in the *ground truth*-columns of Appendix C, where we present a total of 40 randomly chosen samples.

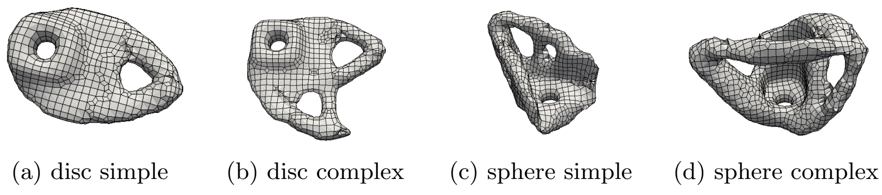


Fig. 1: Ground truth examples from the disc and sphere dataset [16]. The densities are defined on a voxel grid and are smoothed for visualization purposes using Taubin smoothing [45].

## 4. Methods.

This section introduces the DL pipelines we evaluate and analyze in Section 5. We examine different input preprocessing strategies and the effects of physically motivated group averaging [35]. We use an end-to-end learning approach, i.e., train a neural network $f$ to map preprocessed TO problems to a optimized density distributions provided by the datasets.

**4.1. Preprocessings.** Choosing a suitable input preprocessing strategy is crucial for DL. We now present the two main preprocessings we use in this paper. It is possible to combine these via simple channel-wise concatenation of their outputs.

1. **Trivial preprocessing.** The input of the neural network is a 7-channel tensor which results from the channel-wise concatenation of Dirichlet boundary conditions $\omega_{\text{Dirichlet}}$, loads $F$, and design space information $\omega_{\text{design}}$. Additionally, we normalize each sample's $F$ via the mean $\|F\|_\infty$ over all training samples. This is arguably the most straightforward and intuitive type of preprocessing for learned end-to-end TO.
2. **PDE preprocessing.** As proposed by Zhang et al. (2019) [54], we first define an *initial density distribution* $\rho_{\text{init}}$ that is 1 on $\Omega_1$ and $\Omega_{-1}$, and 0 on $\Omega_0$. For $\rho_{\text{init}}$, we then compute the *initial von Mises stresses*, which we obtain by solving the PDE for linear elasticity. We normalize the resulting tensor analogously to the normalization of $F$ above. These initial von Mises stresses are then used as a 1-channel input to the neural network. Analogously, it would also be possible to use the full initial stress tensor or the initial displacements as network input.

We illustrate both preprocessing strategies and our model pipeline in Figure 2.

**4.2. Architecture.** We choose a UNet [41] as our neural network architecture, which is a convolutional encoder-decoder network. The encoder consists of repeated application of convolutions, each followed by a rectified linear unit (ReLU) activation function and a max pooling operation. During the encoding, the encoder of the UNet reduces the spatial information while it increases the feature information. The decoder then extends the feature and spatial information through convolution and upsampling steps and concatenations with high-resolution features from the encoder. See Appendix A for more details on the UNet architecture.

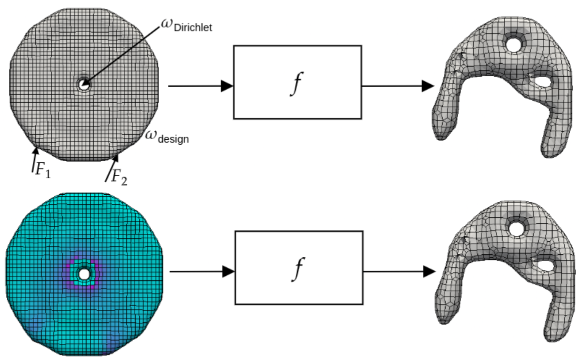


Fig. 2: Illustration of our model pipeline, using an end-to-end learning approach, where the function $f$ represents the neural network. Top: Trivial preprocessing. We concatenate $\omega_{\text{Dirichlet}}$, $\omega_{\text{design}}$, and forces $F = F_1 \cup F_2$ into a 7-channel tensor, which after normalization, we use as input for $f$. Bottom: PDE preprocessing. We compute an initial density distribution $\rho_{\text{init}}$ and corresponding initial displacements and stresses. The image displays the initial von Mises stress distribution, which is a 1-channel tensor.

**4.3. Equivariance.** Equivariance is the property of a function to commute with the actions of a symmetry group acting on its domain and range. For a given transformation group $G$, we say that a function $f : X \to Y$ is ($G$-)equivariant if

$$f(T_g^X(x)) = T_g^Y(f(x)) \qquad \forall g \in G,\ x \in X, \tag{4.1}$$

where $T_g^X$ and $T_g^Y$ denote linear *group actions* in the corresponding spaces $X$ and $Y$ [13]. That is, transforming an input $x$ by a transformation $g$ and then passing it through $f$ should give the same result as first mapping $x$ through $f$ and then transforming the output $f(x)$ (see left image of Figure 3). In many machine learning tasks, we possess prior knowledge about equivariances our predictor should have. Including such knowledge directly into the model can significantly facilitate learning by freeing up model capacity for other factors of variation [49]. Since $G$ is a group, it also contains the identity transformation and a unique inverse transformation $g^{-1}$ for each $g \in G$. Therefore, we can reformulate (4.1) as

$$f(x) = T_{g^{-1}}^Y \left[ f\left(T_g^X(x)\right)\right],$$

allowing the implementation of equivariance via *group averaging* [31, 35] by defining an equivariance wrapper $F_G^f$ as

$$F_G^f(x) := \frac{1}{|G|} \sum_{g \in G} T_{g^{-1}}^Y \left[ f(T_g^X(x))\right].$$

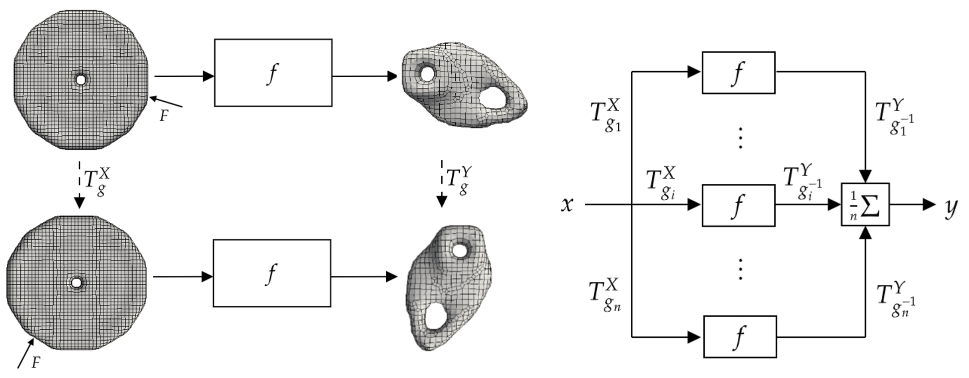


Fig. 3: Left: If $f : X \to Y$ is $G$-equivariant, then for each $g \in G$ applying $f \circ T_g^X$ is equivalent to $T_g^Y \circ f$. Here, $g$ is represented by a 90° clockwise rotation. Right: Illustration of the equivariance wrapper $F_G^f$. We transform the input $x$ via group actions $T_{g_i}^X$ and feed the transformed data into the model $f$. We then reverse the initial transformation on the outputs via $T_{g_i^{-1}}^Y$ and aggregate the results. This process is called *group averaging*. The choice of the (finite) transformation group $G$ is entirely free.

$F_G^f$ is equivariant with respect to $G$ since for each $h \in G$ it holds that

$$\begin{aligned} T_{h^{-1}}^Y \left[ F_G^f \left( T_h^X(x) \right) \right] &= \frac{1}{|G|} \sum_{g \in G} T_{(gh)^{-1}}^Y \left[ f \left( T_{gh}^X(x) \right) \right] \\ &= \frac{1}{|G|} \sum_{gh \in G} T_{g^{-1}}^Y \left[ f \left( T_g^X(x) \right) \right] \\ &= \frac{1}{|G|} \sum_{g \in G} T_{g^{-1}}^Y \left[ f \left( T_g^X(x) \right) \right] \\ &= F_G^f(x). \end{aligned}$$

For an illustration of $F_G^f$ see the right image of Figure 3.

The most popular approach to enforcing group equivariance in DL is enforcing invariance of the network's convolutional filters and activation functions [13]. However, performing group averaging has several advantages. First, it is straightforward to implement. Second, its plug-and-play nature allows the trivial application to any (finite) transformation group $G$ and any model $f$. In particular, the applicability to any $f$ allows for direct comparisons between any non-equivariant model and its equivariant counterpart.

Natural choices of transformation groups for our applications are the dihedral symmetry group $D_4$ and the octahedral symmetry group $O_h$, which describe all combinations of 90° rotations and reflections in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. Thus the equivariance wrapper applies 8 transformations in the $D_4$ and 48 in the $O_h$ case. Note that group actions on vector fields affect spacial locations and vector directions, unlike on scalar fields. The necessity to account for vector directions makes equivariance wrappers especially suited and flexible.

We want to clarify that the equivariance wrapper fundamentally differs from *data augmentation*, i.e., augmenting the dataset with rotations and reflections. Augmentation averages over the objective, whereas the wrapper averages over the model. This difference endows the wrapper with several significant advantages, most notably: The model does not need to learn about equivariance as it is hard-coded. Consequently, the wrapper is not only approximately equivariant but exactly – importantly, this holds for any input, even for inputs drastically different from the training data.

**5. Numerical experiments.** In this section, we conduct several numerical experiments to illustrate the effectiveness of adding physics-based information via PDE preprocessing and equivariance to our DL models.

**5.1. Training.** We train and compare different combinations of preprocessings described in Section 4, each with and without equivariance. Due to the reduced number of voxels in the $z$-direction and for simplicity, we chose the dihedral symmetry group, $D_4$, as the transformation group for all our experiments. All models are implemented in PyTorch [34]. We determine the batch sizes individually, depending on the memory capacity and comparability (see Appendix A). We choose the weighted *binary cross-entropy* (BCE) as our loss function, and calculate the weighting factor based on the training dataset. We use the Adam optimizer [27] with a learning rate of $10^{-3}$ in all our experiments.

We train all models until their improvement on the validation set stalled for 100 epochs and then pick the model corresponding to the best validation epoch; this stopping criterion results in models trained for 100 to 1000 epochs.

We also experimented with different network architectures, transformation groups and preprocessing strategies. However, they led to worse performances than the ones presented here. Regardless, we think that these experiments can still be of interest for the research community (see Appendix B).

**5.2. Evaluation.** We now discuss our evaluation methodology for the analysis of our DL models. In particular, we want to compare the impact of different preprocessing strategies and equivariances on our models. We begin by defining our evaluation criteria.

**5.2.1. Evaluation criteria.** We now introduce the criteria we use to evaluate our models over the validation datasets. Before we apply the criteria we first binarize the densities produced by the DL models, such that they only contain 0s and 1s. We make use of the following two criteria:

- **IoU:** In contrast to our loss, the BCE, which is distribution-based, the *Intersection over Union* (IoU) is region-based. It is defined as

$$\mathrm{IoU} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}},$$

  where TP, FN and FP denote the number of true positive, false negative and false positive voxel predictions. We limit the evaluation of the IoU to the editable design space $\Omega_{-1}$. Following Goodhart's law [22] and established practices in the segmentation community, we use the IoU as our primary evaluation metric [29, 41].
- **Fail percentage:** We consider a prediction failed if the von Mises stress in any voxel exceeds the yield stress by more than 10% or if any voxel with a load case does not connect to one containing Dirichlet conditions. The fail

percentage is the fraction of failed parts. This criterion makes us the first in the DL for TO community who verify their predictions' mechanical integrity.

A typical evaluation criterion in the DL for TO literature is the (balanced) accuracy metric. However, we found that in most cases, IoU yields comparable values but, in general, reflects the quality of its input more appropriately when used across different datasets. Given that we can interpret the binary densities of our structures as segmentation masks, IoU's applicability is not surprising as it is the most common metric for semantic segmentation. Therefore we will use IoU as our main evaluation criterion.

**5.2.2. Sample efficiency.** One of the main impediments to learned supervised TO is the high cost associated with generating new datasets. The necessity to generate thousands of problems with corresponding ground truth solutions often hinders practical applicability in real-world situations since the generation of large datasets can take days, even weeks. Consequently, reducing the number of required training samples is highly beneficial, e.g., by modifying the DL model design. Therefore, we put a particular emphasis on the visualization and measurement of our models' *sample efficiency*, i.e., the model's performance when trained on few training samples.

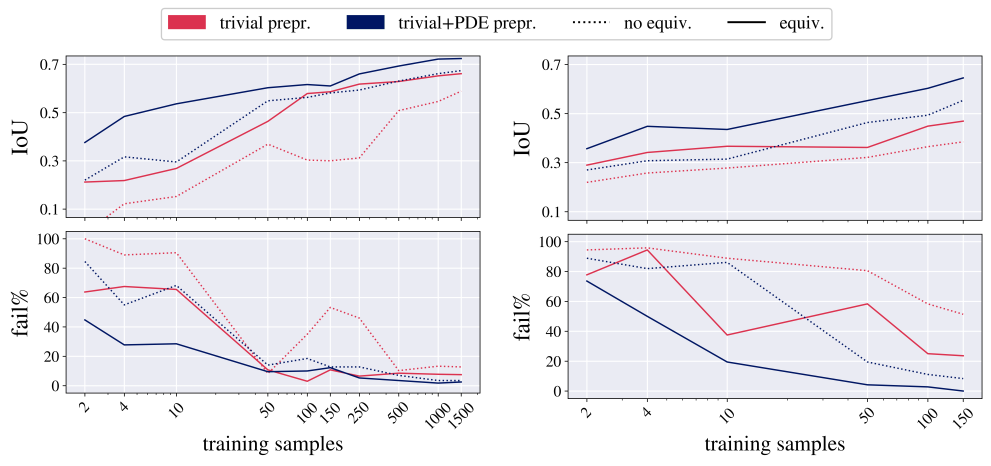
We visualize the sample efficiency of a model with what we call *sample efficiency curves* (SE curves). For each SE curve, we train separate instances of a given model setup on subsets of the original training dataset of varying sizes. We then evaluate and compare the performance of these models on a fixed validation dataset, using IoU and fail percentage as our evaluation criteria. For the disc dataset, we choose training subsets of sizes 2, 4, 10, 50, 100, 150, 250, 500, 1000, and 1500. For the sphere dataset we train on 2, 4, 10, 50, 100, and 150 samples. This way, we obtain an individual SE curve for each evaluation criterion. In order to compare different SE curves of the same criterion, we report two metrics:

1. The normalized *area under the curve* (AUC) of that criterion up to a training sample size of 150, denoted by $\mathrm{AUC}_{150}$. We use this metric to quantify sample efficiency.
2. The *final score*, which is the value of that criterion achieved by the model trained on the largest training subset.

**5.3. Results.** This section gives an overview of our numerical results and compares the performance of different models. We begin with our main results, followed by an analysis of our models' generalization capabilities. Finally, we discuss model alterations that did not yield improvements but might still be helpful for further understanding and research.

**5.3.1. Main results.** As expected, model performance tends to improve with increased training data. We also observe dramatic boosts in the UNet's performance when incorporating physics via equivariance and trivial+PDE preprocessing. These improvements are especially visible for low numbers of training samples; see the SE curves in Figure 4 and $\mathrm{AUC}_{150}$ scores in Table 2. From Figure 4, we observe that using trivial+PDE preprocessing and equivariance in combination leads to a reduction in the required training samples by *two orders of magnitude* while maintaining equivalent IoU scores. Additionally, we notice a reduction in the fail percentage to almost 0%.

The improvements are also evident when visually comparing ground truths with predictions; see Tables 4 to 7. Further, we observe that adding PDE preprocessing and equivariance leads to predictions closer to the ground truth densities, which is espe-



(a) disc combined (b) sphere combined

Fig. 4: Sample efficiency curves, trained and evaluated on disc combined (left) and sphere combined (right). The x-axis shows the size of the training dataset for the different models on a logarithmic scale. On the $y$-axis we see performance of the criteria IoU and fail percentage. For each criterion we evaluate models using trivial preprocessing (red), trivial+PDE preprocessing (blue), equivariance (solid line) and no equivariance (dashed line).

cially noticeable on small training sets. Moreover, applying our equivariance wrapper reduces the necessary number of epochs but increases overall training duration, see Table 8 in Appendix A.

For an unbiased impression of our model predictions and to avoid cherry-picking, we display 20 random samples and predictions from both datasets in Tables 9 and 10 in Appendix C. We show two predictions for each sample, one using trivial preprocessing only and one using trival+PDE preprocessing and equivariance. We trained the UNets on 1500 and 150 samples for disc and sphere, respectively.

**5.3.2. Generalization.** As in all machine learning, generalization is also a problem in its application to TO. The problem led to some publications addressing the issue [32, 54]. We quantify our models' generalizability by cross-evaluating them on different data subsets not used during training, i.e., we evaluate all models trained on either disc simple, disc complex, or disc combined on each of the others. We proceed analogously for the sphere models and datasets.

We compare the $\mathrm{AUC}_{150}$ and final scores for each model based on the corresponding SE curve. For both IoU and fail percentage, we present the results for the $\mathrm{AUC}_{150}$ score in Table 2 and the final score in Table 3. We observe that the addition of PDE preprocessing and equivariance improves the generalization capabilities of our models considerably for both IoU and fail percentage, especially on small training sets.

**6. Conclusion.** We aimed to provide a strong foundation for future research in DL for TO. On the one hand, we proposed and analyzed basic components and principles for designing DL pipelines for TO. On the other hand, we provide two TO datasets enabling the training and comparability of DL methods.

More specifically, method-wise, we focused on physical correctness and sample-efficiency. In practice, the existence of appropriate large-scale training data is costly or simply not realistic; hence it is our conviction that sample-efficiency is a crucial component of any future DL approach to TO.

To achieve this, we developed a physics-inspired approach. We conducted a large-scale ablation study to prove the effectiveness of the two critical components of our approach – on the one hand, PDE-based preprocessing and, on the other, mirror and rotation equivariant architectures. These two key elements drastically improve the overall predictions' physical correctness and the sample efficiency, i.e., the model performance when trained on only a few samples.

On the data side, we publish two three-dimensional TO datasets with a total count of almost 10 000 problem-ground truth pairs. To our knowledge, these are the first publicly available datasets for TO.

Table 2: Tables containing $\text{AUC}_{150}$ scores for different combinations of preprocessings and equivariances. We (cross-)evaluate performance of models trained on the disc and sphere datasets and highlight the row-wise best scores in bold.

| | | trivial prepr. | | trivial+PDE prepr. | |
|---|---|---|---|---|---|
| eval on | train on | no equiv. | equiv. | no equiv. | equiv. |
| disc simple | simple | 0.76 | 0.86 | 0.84 | **0.87** |
| | complex | 0.35 | 0.59 | 0.57 | **0.63** |
| | combined | 0.63 | 0.80 | 0.78 | **0.81** |
| disc complex | simple | 0.35 | 0.40 | 0.43 | **0.44** |
| | complex | 0.28 | 0.44 | 0.48 | **0.58** |
| | combined | 0.33 | 0.45 | 0.48 | **0.56** |
| disc combined | simple | 0.55 | 0.63 | 0.64 | **0.65** |
| | complex | 0.31 | 0.52 | 0.52 | **0.61** |
| | combined | 0.48 | 0.63 | 0.63 | **0.69** |
| sphere simple | simple | 0.28 | 0.41 | 0.50 | **0.61** |
| | complex | 0.28 | 0.32 | 0.39 | **0.44** |
| | combined | 0.30 | 0.37 | 0.43 | **0.56** |
| sphere complex | simple | 0.20 | 0.27 | 0.29 | **0.38** |
| | complex | 0.45 | 0.47 | 0.56 | **0.59** |
| | combined | 0.38 | 0.45 | 0.49 | **0.57** |
| sphere combined | simple | 0.24 | 0.34 | 0.40 | **0.49** |
| | complex | 0.36 | 0.39 | 0.48 | **0.52** |
| | combined | 0.34 | 0.41 | 0.46 | **0.56** |

(a) $\text{AUC}_{150}$ scores for the IoU criterion. Higher values indicate better results.

| | | trivial prepr. | | trivial+PDE prepr. | |
|---|---|---|---|---|---|
| eval on | train on | no equiv. | equiv. | no equiv. | equiv. |
| disc simple | simple | 1.71 | 0.10 | 0.33 | **0.01** |
| | complex | 27.21 | 1.35 | 1.11 | **1.10** |
| | combined | 2.95 | 0.76 | 0.89 | **0.05** |
| disc complex | simple | 39.11 | **10.21** | 22.30 | 14.50 |
| | complex | 49.39 | 16.13 | 9.37 | **5.07** |
| | combined | 37.58 | 17.07 | 14.95 | **8.47** |
| disc combined | simple | 20.41 | **5.16** | 16.31 | 12.26 |
| | complex | 38.30 | 8.74 | 5.24 | **2.69** |
| | combined | 20.27 | 8.92 | 7.92 | **4.26** |
| sphere simple | simple | 48.03 | 12.16 | 9.55 | **4.11** |
| | complex | 59.52 | 36.84 | 8.41 | **6.06** |
| | combined | 56.51 | 24.02 | 27.93 | **7.73** |
| sphere complex | simple | 94.37 | 68.17 | 72.60 | **34.95** |
| | complex | 63.53 | 37.44 | 14.62 | **3.55** |
| | combined | 83.33 | 54.13 | 26.61 | **6.42** |
| sphere combined | simple | 71.20 | 40.17 | 41.08 | **19.53** |
| | complex | 61.52 | 37.14 | 11.51 | **4.80** |
| | combined | 69.92 | 39.08 | 27.27 | **7.08** |

(b) $\text{AUC}_{150}$ scores for the fail% criterion.

Table 3: Tables containing the final scores for different combinations of preprocessings and equivariances. We (cross-)evaluate performance of models trained on the disc and sphere datasets and highlight the row-wise best scores in bold.

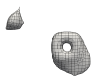
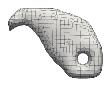
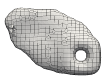
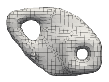
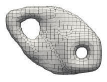
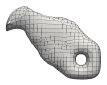
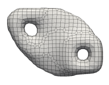
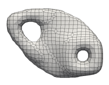
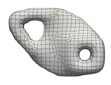
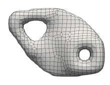
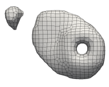
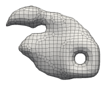
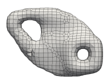
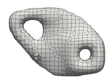
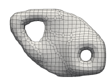
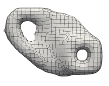
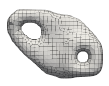
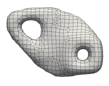
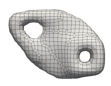
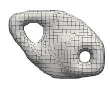
| | | trivial prepr. | | trivial+PDE prepr. | |
|---|---|---|---|---|---|
| eval on | train on | no equiv. | equiv. | no equiv. | equiv. |
| disc simple | simple | 0.87 | 0.89 | 0.90 | **0.91** |
| | complex | 0.52 | 0.61 | 0.61 | **0.62** |
| | combined | 0.77 | 0.84 | 0.83 | **0.85** |
| disc complex | simple | 0.37 | 0.40 | 0.44 | **0.45** |
| | complex | 0.40 | 0.48 | 0.52 | **0.62** |
| | combined | 0.40 | 0.48 | 0.52 | **0.59** |
| disc combined | simple | 0.62 | 0.65 | 0.67 | **0.68** |
| | complex | 0.46 | 0.55 | 0.57 | **0.62** |
| | combined | 0.59 | 0.66 | 0.67 | **0.72** |
| sphere simple | simple | 0.26 | 0.46 | 0.58 | **0.67** |
| | complex | 0.29 | 0.35 | 0.39 | **0.48** |
| | combined | 0.34 | 0.44 | 0.52 | **0.66** |
| sphere complex | simple | 0.18 | 0.23 | 0.35 | **0.40** |
| | complex | 0.46 | 0.55 | 0.64 | **0.67** |
| | combined | 0.43 | 0.50 | 0.58 | **0.63** |
| sphere combined | simple | 0.22 | 0.34 | 0.46 | **0.53** |
| | complex | 0.37 | 0.45 | 0.51 | **0.57** |
| | combined | 0.38 | 0.47 | 0.55 | **0.65** |

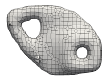
(a) Final scores of the IoU criterion. Higher values indicate better results.

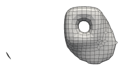
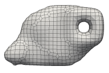
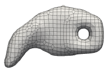
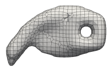
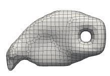
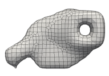
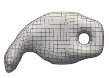
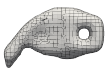
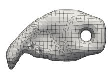
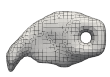
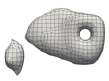
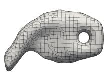
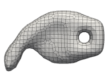
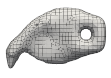
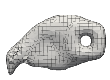
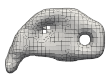
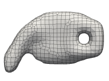
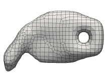
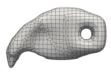
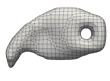
| | | trivial prepr. | | trivial+PDE prepr. | |
|---|---|---|---|---|---|
| eval on | train on | no equiv. | equiv. | no equiv. | equiv. |
| disc simple | simple | **0.00** | **0.00** | **0.00** | **0.00** |
| | complex | **0.00** | **0.00** | **0.00** | **0.00** |
| | combined | **0.00** | **0.00** | **0.00** | **0.00** |
| disc complex | simple | 32.50 | 15.50 | 24.50 | **13.50** |
| | complex | 12.50 | 11.00 | 5.00 | **3.50** |
| | combined | 25.50 | 15.00 | 7.00 | **5.00** |
| disc combined | simple | 16.25 | 7.75 | 12.25 | **6.75** |
| | complex | 6.25 | 5.50 | 2.50 | **1.75** |
| | combined | 12.75 | 7.50 | 3.50 | **2.50** |
| sphere simple | simple | 52.78 | 8.33 | **0.00** | **0.00** |
| | complex | 69.44 | 13.89 | **0.00** | **0.00** |
| | combined | 33.33 | 8.33 | 8.33 | **0.00** |
| sphere complex | simple | 99.97 | 61.11 | 38.89 | **13.89** |
| | complex | 66.67 | 13.89 | **0.00** | **0.00** |
| | combined | 69.44 | 38.89 | 8.33 | **0.00** |
| sphere combined | simple | 76.39 | 34.72 | 19.44 | **6.94** |
| | complex | 68.06 | 13.89 | **0.00** | **0.00** |
| | combined | 51.39 | 23.61 | 8.33 | **0.00** |

(b) Final scores of the fail% criterion. Lower values indicate better results.

Table 4: Model predictions of two different problems from the **disc simple** validation dataset, using the UNet with different preprocessings and equivariances. We train the models on subsets of the dataset and vary the training size along the columns of the table. At the boxes below the tables we show the corresponding ground truth density for each problem.

| | | training samples | | | | |
|---|---|---|---|---|---|---|
| prepr. | equiv. | 10 | 50 | 100 | 500 | 1500 |
| trivial | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |
| trivial+PDE | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |



| | | training samples | | | | |
|---|---|---|---|---|---|---|
| prepr. | equiv. | 10 | 50 | 100 | 500 | 1500 |
| trivial | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |
| trivial+PDE | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |

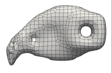
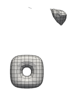
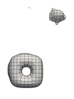
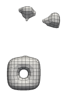
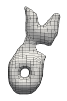
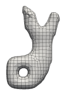
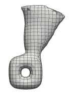
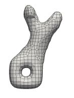
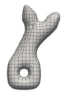
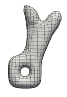
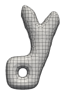
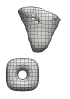
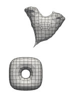
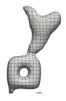
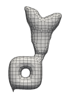
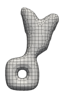
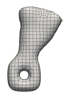
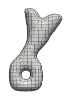
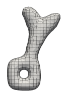
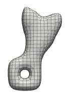
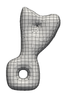
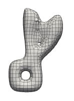
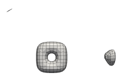
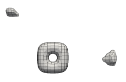
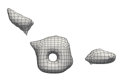
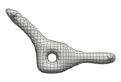
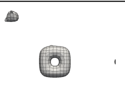
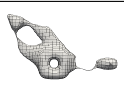
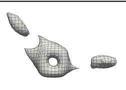
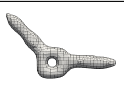
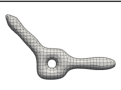
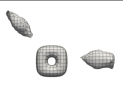
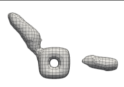
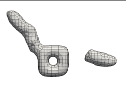
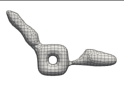
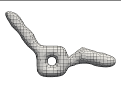
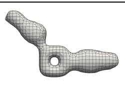
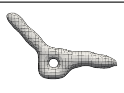
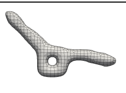
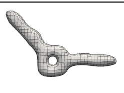
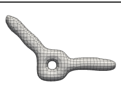


Table 5: Model predictions of two different problems from the **disc complex** validation dataset, using the UNet with different preprocessings and equivariances. We train the models on subsets of the dataset and vary the training size along the columns of the table. At the boxes below the tables we show the corresponding ground truth density for each problem.

| | | training samples | | | | |
|---|---|---|---|---|---|---|
| prepr. | equiv. | 10 | 50 | 100 | 500 | 1500 |
| trivial | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |
| trivial+PDE | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |



| | | training samples | | | | |
|---|---|---|---|---|---|---|
| prepr. | equiv. | 10 | 50 | 100 | 500 | 1500 |
| trivial | |  | |  |  |  |
| | ✓ |  |  |  |  |  |
| trivial+PDE | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |

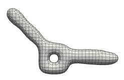
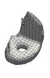
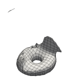
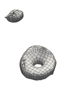
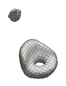
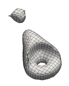
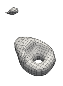
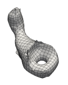
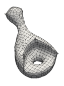
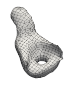
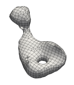
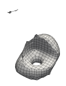
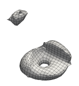
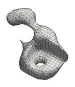
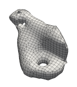
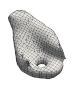
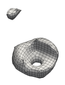
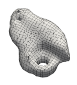
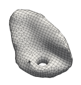
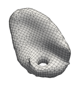
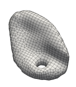
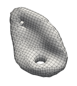
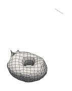
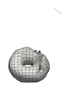
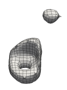
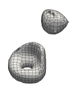
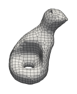
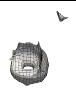
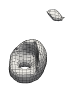
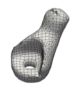
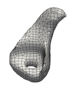
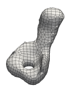
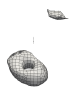
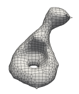
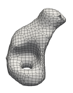
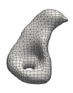
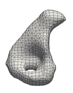
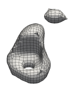
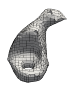
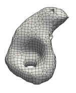
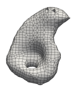
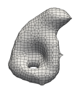


Table 6: Model predictions of two different problems from the **sphere simple** validation dataset, using the UNet with different preprocessings and equivariances. We train the models on subsets of the dataset and vary the training size along the columns of the table. At the boxes below the tables we show the corresponding ground truth density for each problem.

| | | training samples | | | | |
|---|---|---|---|---|---|---|
| prepr. | equiv. | 10 | 50 | 100 | 500 | 1500 |
| trivial | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |
| trivial+PDE | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |



| | | training samples | | | | |
|---|---|---|---|---|---|---|
| prepr. | equiv. | 10 | 50 | 100 | 500 | 1500 |
| trivial | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |
| trivial+PDE | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |

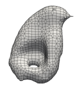
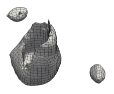
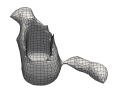
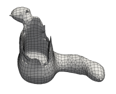
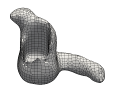
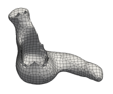
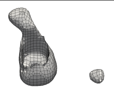
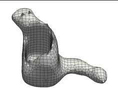
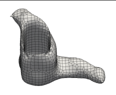
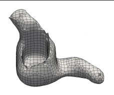
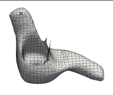
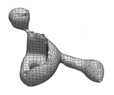
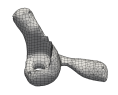
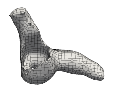
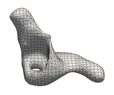
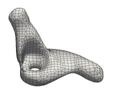
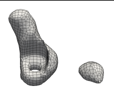
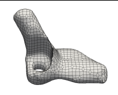
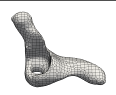
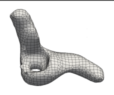
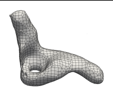
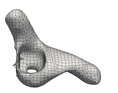
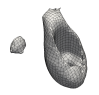
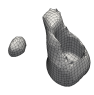
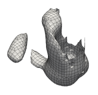
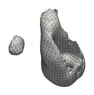
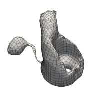
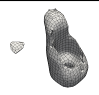
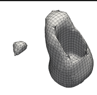
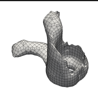
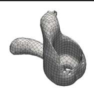
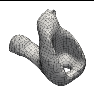
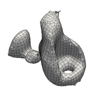
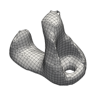
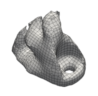
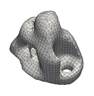
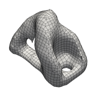
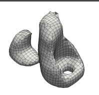
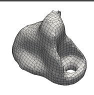
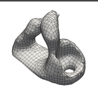
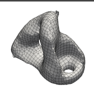
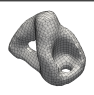
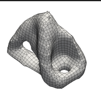


Table 7: Model predictions of two different problems from the **sphere complex** validation dataset, using the UNet with different preprocessings and equivariances. We train the models on subsets of the dataset and vary the training size along the columns of the table. At the boxes below the tables we show the corresponding ground truth density for each problem.

| | | training samples | | | | |
|---|---|---|---|---|---|---|
| prepr. | equiv. | 10 | 50 | 100 | 500 | 1500 |
| trivial | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |
| trivial+PDE | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |



| | | training samples | | | | |
|---|---|---|---|---|---|---|
| prepr. | equiv. | 10 | 50 | 100 | 500 | 1500 |
| trivial | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |
| trivial+PDE | |  |  |  |  |  |
| | ✓ |  |  |  |  |  |



**Acknowledgments.** We thank Marco Gosch, Rielson Falck, Christian Knorr (ArianeGroup) and Daniel Siegel (Synera) for providing the datasets. We also thank Denis Regenbrecht and Susanne Heckrodt (DLR, German Aerospace Center) and Janek Gödeke (University of Bremen) for their support.

## REFERENCES

[1] N. Aage, E. Andreassen, and B. S. Lazarov, *Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework*, Structural and Multidisciplinary Optimization, 51 (2015), pp. 565–572.

[2] D. W. Abueidda, S. Koric, and N. A. Sobh, *Topology optimization of 2d structures with nonlinearities using deep learning*, Computers & Structures, 237 (2020), p. 106283.

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

[4] M. P. Bendsoe and O. Sigmund, *Topology optimization: theory, methods, and applications*, Springer Science & Business Media, 2003.

[5] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*, vol. 1, MIT press Cambridge, MA, USA, 2017.

[6] A. Bogatskiy, B. Anderson, J. Offermann, M. Roussi, D. Miller, and R. Kondor, *Lorentz group equivariant neural network for particle physics*, in International Conference on Machine Learning, PMLR, 2020, pp. 992–1002.

[7] T. Borrvall and J. Petersson, *Topology optimization of fluids in stokes flow*, International journal for numerical methods in fluids, 41 (2003), pp. 77–107.

[8] R. Cang, H. Yao, and Y. Ren, *One-shot generation of near-optimal topology through theory-driven machine learning*, Computer-Aided Design, 109 (2019), pp. 12–21.

[9] A. Chandrasekhar and K. Suresh, *Tounn: topology optimization using neural networks*, Structural and Multidisciplinary Optimization, 63 (2021), pp. 1135–1149.

[10] H. Chen, B. He, H. Wang, Y. Ren, S. N. Lim, and A. Shrivastava, *Nerv: Neural representations for videos*, Advances in Neural Information Processing Systems, 34 (2021), pp. 21557–21568.

[11] H. Chi, Y. Zhang, T. L. E. Tang, L. Mirabella, L. Dalloro, L. Song, and G. H. Paulino, *Universal machine learning for topology optimization*, Computer Methods in Applied Mechanics and Engineering, 375 (2021), p. 112739.

[12] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, *3d u-net: learning dense volumetric segmentation from sparse annotation*, in International conference on medical image computing and computer-assisted intervention, Springer, 2016, pp. 424–432.

[13] T. Cohen and M. Welling, *Group equivariant convolutional networks*, in International conference on machine learning, PMLR, 2016, pp. 2990–2999.

[14] E. M. Dede, *Multiphysics topology optimization of heat transfer and fluid flow systems*, 715 (2009).

[15] H. Deng and A. C. To, *Topology optimization based on deep representation learning (drl) for compliance and stress-constrained design*, Computational Mechanics, 66 (2020), pp. 449–469.

[16] S. Dittmer, D. Erzmann, H. Harms, R. Falck, and M. Gosch, *Selto dataset*. https://doi.org/10.5281/zenodo.7034898, 2023.

[17] M. B. Dühring, J. S. Jensen, and O. Sigmund, *Acoustic design by topology optimization*, Journal of sound and vibration, 317 (2008), pp. 557–575.

[18] B. Dumont, S. Maggio, and P. Montalvo, *Robustness of rotation-equivariant networks to adversarial perturbations*, arXiv preprint arXiv:1802.06627, (2018).

[19] H. A. Eschenauer and N. Olhoff, *Topology optimization of continuum structures: a review*, Appl. Mech. Rev., 54 (2001), pp. 331–390.

[20] J. E. Gerken, J. Aronsson, O. Carlsson, H. Linander, F. Ohlsson, C. Petersson, and D. Persson, *Geometric deep learning and equivariant neural networks*, arXiv preprint arXiv:2105.13926, (2021).

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, Advances in neural information processing systems, 27 (2014).

[22] C. Goodhart, *Problems of monetary management: the uk experience in papers in monetary economics*, Monetary Economics, 1 (1975).

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

[24] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in International conference on machine learning, PMLR, 2015, pp. 448–456.

[25] P. D. L. Jensen, F. Wang, I. Dimino, and O. Sigmund, *Topology optimization of large-scale 3d morphing wing structures*, in Actuators, vol. 10, MDPI, 2021, p. 217.

[26] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, *Generalization in deep learning*, arXiv preprint arXiv:1710.05468, (2017).

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

[28] S. Lee, H. Kim, Q. X. Lieu, and J. Lee, *Cnn-based image recognition for topology optimization*, Knowledge-Based Systems, 198 (2020), p. 105887.

[29] J. Ma, *Segmentation loss odyssey*, arXiv preprint arXiv:2005.13449, (2020).

[30] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L. F. Fei-Fei, J. Tenenbaum, and D. L. Yamins, *Flexible neural representation for physics prediction*, Advances in neural information processing systems, 31 (2018).

[31] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, *Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs*, arXiv preprint arXiv:1811.01900, (2018).

[32] Z. Nie, T. Lin, H. Jiang, and L. B. Kara, *Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain*, Journal of Mechanical Design, 143 (2021).

[33] S. Oh, Y. Jung, S. Kim, I. Lee, and N. Kang, *Deep generative design: Integration of topology optimization and generative models*, Journal of Mechanical Design, 141 (2019).

[34] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in pytorch*, (2017).

[35] O. Puny, M. Atzmon, H. Ben-Hamu, E. J. Smith, I. Misra, A. Grover, and Y. Lipman, *Frame averaging for invariant and equivariant network design*, arXiv preprint arXiv:2110.03336, (2021).

[36] F. Pérez-García, *Pytorch implementation of 2d and 3d u-net (v0.7.5)*. https://github.com/fepegar/unet, 2020.

[37] C. Qian and W. Ye, *Accelerating gradient-based topology optimization design with dual-model artificial neural networks*, Structural and Multidisciplinary Optimization, 63 (2021), pp. 1687–1707.

[38] J. Rade, A. Balu, E. Herron, J. Pathak, R. Ranade, S. Sarkar, and A. Krishnamurthy, *Physics-consistent deep learning for structural topology optimization*, arXiv preprint arXiv:2012.05359, (2020).

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

[40] S. Rawat and M.-H. H. Shen, *A novel topology optimization approach using conditional deep learning*, arXiv preprint arXiv:1901.04859, (2019).

[41] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[42] F. V. Senhora, H. Chi, Y. Zhang, L. Mirabella, T. L. E. Tang, and G. H. Paulino, *Machine learning for topology optimization: Physics-based learning through an independent training strategy*, Computer Methods in Applied Mechanics and Engineering, 398 (2022), p. 115116.

[43] M.-H. H. Shen and L. Chen, *A new cgan technique for constrained topology design optimization*, arXiv preprint arXiv:1901.07675, (2019).

[44] I. Sosnovik and I. Oseledets, *Neural networks for topology optimization*, Russian Journal of Numerical Analysis and Mathematical Modelling, 34 (2019), pp. 215–223.

[45] G. Taubin, *Curve and surface smoothing without shrinkage*, in Proceedings of IEEE international conference on computer vision, IEEE, 1995, pp. 852–857.

[46] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, *Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds*, arXiv preprint arXiv:1802.08219, (2018).

[47] C. Wang, S. Yao, Z. Wang, and J. Hu, *Deep super-resolution neural network for structural topology optimization*, Engineering Optimization, 53 (2021), pp. 2108–2121.

[48] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. S. Cohen, *3d steerable cnns: Learning rotationally equivariant features in volumetric data*, Advances in Neural Information Processing Systems, 31 (2018).

[49] M. Weiler, F. A. Hamprecht, and M. Storath, *Learning steerable filters for rotation equivariant cnns*, (2018), pp. 849–858.

[50] L. Xue, J. Liu, G. Wen, and H. Wang, *Efficient, high-resolution topology optimization method based on convolutional neural networks*, Frontiers of Mechanical Engineering, 16 (2021), pp. 80–96.

[51] G. H. Yoon, J. S. Jensen, and O. Sigmund, *Topology optimization of acoustic–structure interaction problems using a mixed finite element formulation*, International journal for numerical methods in engineering, 70 (2007), pp. 1049–1075.

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

[53] J. Zehnder, Y. Li, S. Coros, and B. Thomaszewski, *Ntopo: Mesh-free topology optimization using implicit neural representations*, Advances in Neural Information Processing Systems, 34 (2021), pp. 10368–10381.

[54] Y. Zhang, B. Peng, X. Zhou, C. Xiang, and D. Wang, *A deep convolutional neural network for topology optimization with strong generalization ability*, arXiv preprint arXiv:1901.07761, (2019).

[55] Z. Zhang, Y. Li, W. Zhou, X. Chen, W. Yao, and Y. Zhao, *Tonr: An exploration for a novel way combining neural network with topology optimization*, Computer Methods in Applied Mechanics and Engineering, 386 (2021), p. 114083.

[56] Z. Zhang, Q. Liu, and Y. Wang, *Road extraction by deep residual u-net*, IEEE Geoscience and Remote Sensing Letters, 15 (2018), pp. 749–753.

**Appendix A. Network architecture and training details.**

Our UNet architecture is inspired by Çiçek et al.[12] and based on the implementation from Pérez-García[36]. Like in Çiçek et al.[12], each layer of our encoder contains two $3 \times 3 \times 3$ convolutions, each followed by batch normalization [24] and a rectified linear unit (ReLU) activation function, and then a max pooling operation. For the sphere dataset, we choose a pooling kernel size of $2 \times 2 \times 2$; for the disc dataset, we choose $2 \times 2 \times 1$ to account for the lower number of voxels in the $z$-direction. In the decoding path, each layer consists of a nearest neighbor upsampling step, followed by two $3 \times 3 \times 3$ convolutions with batch normalization and ReLU activation functions. In the last layer, a $1 \times 1 \times 1$ convolution with a sigmoid activation function reduces the number of output channels to 1 and acts as a binary voxel-wise classifier. We use skip connections to pass information from layers of equal resolution in the encoding path to the decoder and apply a dropout rate of 0.3 to reduce overfitting. We use five encoding and four decoding blocks. Each encoding block doubles the number of channels, starting with 10 output channels for the first block. We choose the padding, so the image resolution stays unaffected by the convolutions. Based on memory availability, we set the batch size to 128 when training on the disc dataset and 28 when training on the sphere dataset. All models have been implemented in PyTorch [34] with a total parameter count of approximately 1.6M. For training we used an Nvidia GeForce GTX 1080 Ti GPU with 11GB VRAM. For the number of training epochs and the approximate training times for different models trained on the largest training subset, see Table 8.

Table 8: The number of training epochs and the approximate training time for UNets with different preprocessings and equivariances, trained on the largest training subsets of the disc and sphere combined datasets.

| preprocessing | equivariance | training epochs | training time (in minutes) |
|---|---|---|---|
| trivial | | 210 | 12 |
| trivial | ✓ | 120 | 34 |
| trivial+PDE | | 230 | 14 |
| trivial+PDE | ✓ | 110 | 32 |

(a) disc combined

| preprocessing | equivariance | training epochs | training time (in minutes) |
|---|---|---|---|
| trivial | | 100 | 5 |
| trivial | ✓ | 80 | 17 |
| trivial+PDE | | 170 | 9 |
| trivial+PDE | ✓ | 140 | 31 |

(b) sphere combined

**Appendix B. Ablations and negative results.** This section discusses several model alterations that did not lead to noticeable improvements in our numerical experiments. We think these results are still valuable to the community.

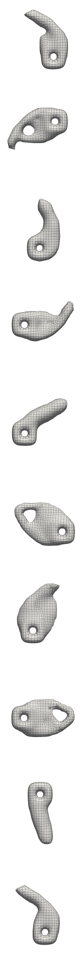
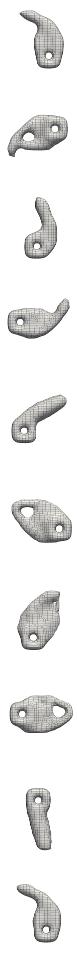
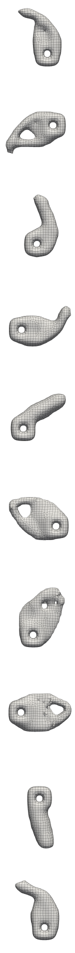
Above, we exclusively used the dihedral symmetry group, $D_4$, in the equivariance wrapper. For the sphere dataset, we also examined the performance of the octahedral symmetry group $O_h$. While still constituting a significant benefit over not utilizing equivariance, this method was inferior to the use of the $D_4$ symmetry group. We suspect this is due to the fixed location of the sphere dataset's Dirichlet condition, negating the benefit of generalizing to more diverse locations of boundary conditions.

In addition to the preprocessings we examined in Subsection 5.3.1, we experimented with various other preprocessing combinations. For instance, for an alternative preprocessing strategy we construct a binary mask to extract all voxels with either Dirichlet boundary or load cases. The *convex hull* of this mask is then given as the set of voxels included in the smallest convex polygon that surrounds all these voxels. In case of one load point and one Dirichlet voxel this results in the simplest possible density distribution that leads to a connected mechanical structure. This convex hull density distribution is then used as a 1-channel input to the neural network. We found that convex hull preprocessing did not lead to any measurable improvements on the validation dataset, neither on its own nor in combination with other preprocessings. We speculate that this is due to the convex hull being invariant to the direction of the applied forces. We also found that applying PDE preprocessing on its own produced similar but somewhat inferior results compared to the trivial+PDE preprocessing. Also, adding the full initial stress tensor and initial displacements to the PDE preprocessing did not improve over the presented PDE preprocessing.

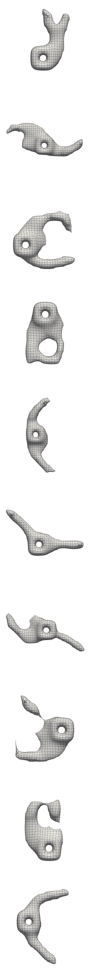
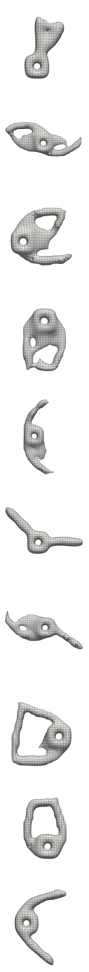
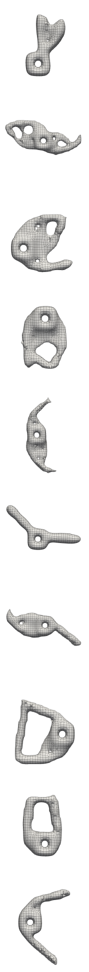
We also experimented with different network architectures. In addition to the UNet, we considered a ResNet [56] of depth 10 with 5-layer CNN blocks. However, despite the UNet requiring less VRAM and computation time, it reliably outperformed the ResNet.

## Appendix C. Examples of model predictions.

Table 9: Random samples from the disc dataset. The first columns displays prediction from the UNet with trivial preprocessing and without equivariance. The second columns shows predictions from the UNet with trivial+PDE preprocessing and equivariance, which is our best model pipeline. All models have been trained on 1500 samples. In the third columns we show the ground truths densities corresponding to each sample.

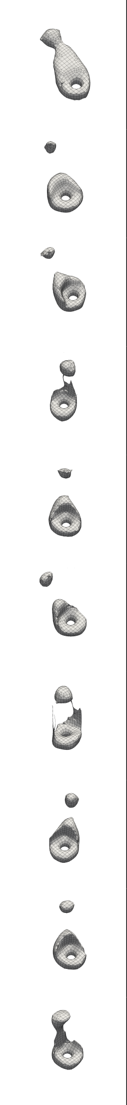
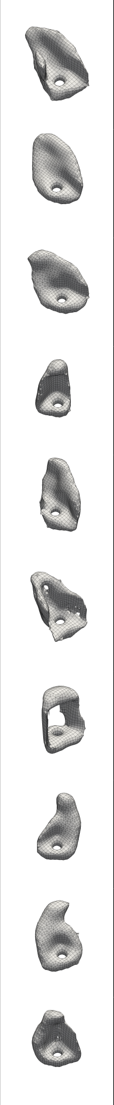
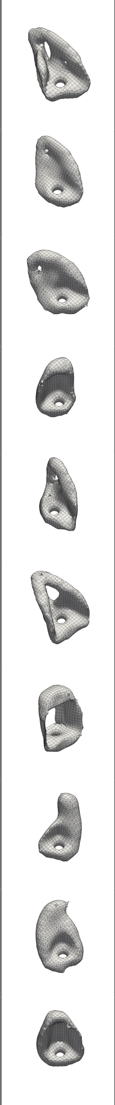
| UNet | UNet +physics | ground truth |
|---|---|---|
|  |  |  |

(a) disc simple

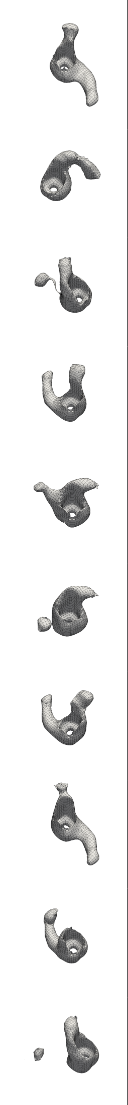
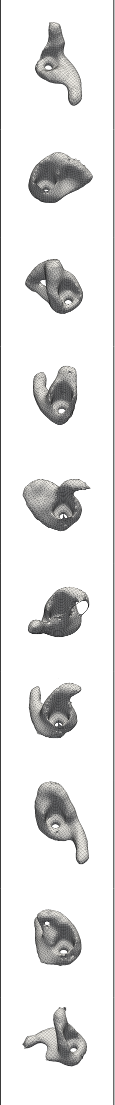
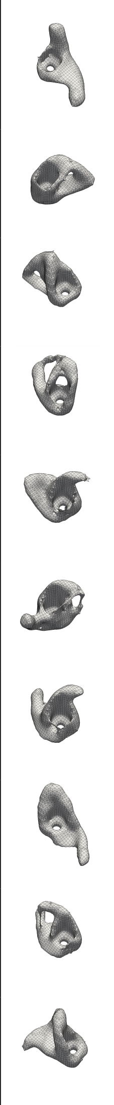
| UNet | UNet +physics | ground truth |
|---|---|---|
|  |  |  |

(b) disc complex

Table 10: Random samples from the sphere dataset. The first columns displays prediction from the UNet with trivial preprocessing and without equivariance. The second columns shows predictions from the UNet with trivial+PDE preprocessing and equivariance, which is our best model pipeline. All models have been trained on 150 samples. In the third columns we show the ground truths densities corresponding to each sample.

| UNet | UNet +physics | ground truth |
|---|---|---|
|  |  |  |

(a) sphere simple

| UNet | UNet +physics | ground truth |
|---|---|---|
|  |  |  |

(b) sphere complex

*Journal of Computational Design and Engineering*, 2024, **00**, 1–10
DOI: 10.1093/jcde/qwae039
Advance access publication date: 19 April 2024
Research Article

# Equivariant neural operators for gradient-consistent topology optimization

David Erzmann[1,*] and Sören Dittmer[1,2]

[1]Center for Industrial Mathematics, University of Bremen, Bibliothekstraße 5, 28359 Bremen, Germany
[2]Cambridge Image Analysis, Centre for Mathematical Sciences, University of Cambridge, Wilberforce Rd, Cambridge CB3 0WA, UK
*Correspondence: erzmann@uni-bremen.de

## Abstract

Most traditional methods for solving partial differential equations (PDEs) require the costly solving of large linear systems. Neural operators (NOs) offer remarkable speed-ups over classical numerical PDE solvers. Here, we conduct the first exploration and comparison of NOs for three-dimensional topology optimization. Specifically, we propose replacing the PDE solver within the popular Solid Isotropic Material with Penalization (SIMP) algorithm, which is its main computational bottleneck. For this, the NO not only needs to solve the PDE with sufficient accuracy but also has the additional challenge of providing accurate gradients which are necessary for SIMP's density updates. To realize this, we do three things: (i) We introduce a novel loss term to promote gradient-consistency. (ii) We guarantee equivariance in our NOs to increase the physical correctness of predictions. (iii) We introduce a novel NO architecture called U-Net Fourier neural operator (U-Net FNO), which combines the multi-resolution properties of U-Nets with the Fourier neural operator (FNO)'s focus on local features in frequency space. In our experiments we demonstrate that the inclusion of the novel gradient loss term is necessary to obtain good results. Furthermore, enforcing group equivariance greatly improves the quality of predictions, especially on small training datasets. Finally, we show that in our experiments the U-Net FNO outperforms both a standard U-Net, as well as other FNO methods.

**Keywords:** deep learning, topology optimization, neural operators, inverse design

## 1. Introduction

Topology optimization (TO) is a cornerstone of mechanical engineering, providing rigorous frameworks for designing mechanical structures that are both lightweight and robust, tailored to meet user-defined objectives and constraints. Recent methods aim to solve TO problems across a diverse spectrum of industries, encompassing aerospace and naval engineering (Aage *et al.*, 2017; Hwang *et al.*, 2023), heat transfer (Ha & Cho, 2005), and medicine (Park *et al.*, 2021). The Solid Isotropic Material with Penalization (SIMP) method (Bendsøe & Sigmund, 2003) is widely regarded as the most significant TO approach. SIMP is an iterative gradient-based method that requires repeated solving of a partial differential equation (PDE). Most traditional methods for solving PDEs involve numerical schemes that can lead to challenges regarding computational cost and handling of complex geometries. The recent advancements in Deep Learning (DL), coupled with the rapid progress in computational power and data storage, have sparked interest in utilizing DL methods to solve PDEs. Notably, the emergence of neural operators (NOs, Kovachki *et al.*, 2021; Li, Kovachki, *et al.*, 2020a, b) promises initiating a paradigm shift in our approach to solving PDEs. One of the compelling advantages of employing such surrogate models to tackle PDE problems is the reduced inference time. Nevertheless, there are trade-offs associated with this acceleration, necessitating thoughtful deliberation as to whether these concessions align with the objectives of a given task. The most apparent drawbacks revolve around generalization and accuracy.

Our primary objective is to develop an NO as a substitute mapping for the PDE solver used in the SIMP framework. We attain strong generalization capabilities from our choice of network architecture, the inclusion of group equivariance, and gradient-consistent training. The paper is structured as follows: we review the current literature in Section 2 and explore NOs in Section 3. Subsequently, in Section 4, we present a comprehensive breakdown of the SIMP method and propose the implementation of NOs for SIMP. Finally, in Section 5, we present numerical experiments that we conducted, along with their results.

In this paper, we address the following key objectives:

(i) We are the first to use NOs in TO as surrogates for SIMP's PDE solver. This requires gradient-consistency, which is not inherent to NOs, and motivates a novel loss function.
(ii) We introduce a new NO called U-Net Fourier neural operator (U-Net FNO). It combines the U-Net's ability to capture spatial features with the Fourier neural operator (FNO) ability to model operators based on Fourier space representations. We show that the U-Net FNO outperforms a standard U-Net and state-of-the-art FNO architectures.
(iii) We introduce group equivariance to NOs, demonstrating clear benefits to the model's sample efficiency, i.e., in the small-data setting.

**Received:** October 16, 2023. **Revised:** March 14, 2024. **Accepted:** March 15, 2024

## 2. Related Work

This section offers a concise review of the existing literature regarding the utilization of DL techniques in tackling both PDEs and TO.

### 2.1. DL for PDEs

Current approaches for solving PDEs using neural networks can be categorized into two distinct strategies. For a more comprehensive and in-depth analysis and comparison, we refer the reader to Huang *et al.* (2022) and Tanyu *et al.* (2023).

In the first approach, PDEs are directly integrated into the training process as constraints (Lagaris *et al.*, 1998; Raissi *et al.*, 2019; T. Xue *et al.*, 2020). These constraints often stem from prior knowledge and are typically incorporated into the loss function used during training. Notable methods following this paradigm include Physics-Informed Neural Networks (PINNs, Raissi *et al.*, 2019), the Deep Ritz Method (Yu & E, 2018), and the Deep Galerkin Method (Sirignano & Spiliopoulos, 2018). Since PDE-based constraints are typically continuous, training does not involve discretized grids, as traditional numerical methods do (Cuomo *et al.*, 2022). However, any alterations to physical parameters, coefficients, or initial/boundary conditions necessitate re-training, which is a notable drawback.

The second approach involves using DL to learn NOs (Li, Kovachki, *et al.*, 2020b), which can serve as approximators for infinite-dimensional parameter-to-state mappings. Chen and Chen (1995) pioneered the early versions of NO methods based on operator theory. Presently, two prominent instances of this approach are the Deep Operator Network (DeepONet, Lu, Jin, *et al.*, 2019) and the FNO (Li, Kovachki, *et al.*, 2020a), both of which have brought about significant advancements in the field. DeepONet emphasizes parallel network structures based on universal approximation theorems, while FNO relies on the Fourier transformation for its design. The core idea behind the FNO architecture is learning a kernel parameterized in Fourier space, where each Fourier layer in the FNO performs a global convolution on its input.

### 2.2. DL for TO

Density-based methods like SIMP operate iteratively, demanding repeated solutions of the PDE for linear elasticity (see Section 4). This iterative process presents computational challenges, particularly when dealing with high-resolution voxel meshes (Aage *et al.*, 2015). Recent advancements in DL research have been dedicated to addressing this inherent challenge (Shin *et al.*, 2023). These advancements can be broadly categorized into three main areas (Zhang *et al.*, 2021).

Firstly, DL techniques can be employed to reduce or even entirely replace the need for SIMP iterations. For instance, researchers have used DL to map intermediate SIMP iterations to the final density, essentially performing a deblurring operation (Abueidda *et al.*, 2020; Banga *et al.*, 2018; Sosnovik & Oseledets, 2019; L. Xue *et al.*, 2021). Alternatively, some have developed methods that directly predict the final density distribution without relying on SIMP iterations at all (Dittmer *et al.*, 2022; Nie *et al.*, 2021; D. Wang *et al.*, 2022; Yu *et al.*, 2019). Although there exist publications addressing three-dimensional (3D) TO (Banga *et al.*, 2018; Challis *et al.*, 2014; Dittmer *et al.*, 2022; Martínez-Frutos & Herrero-Pérez, 2016; Xiang *et al.*, 2022), a substantial portion of the literature predominantly focuses on the 2D case. Handling 3D data poses significantly greater challenges, primarily due to memory and runtime constraints, making the task considerably more intricate.

Secondly, and our method falls into this category, an alternative approach involves replacing SIMP's traditional PDE solver with a neural network, thereby eliminating the primary computational bottleneck and contributing to faster optimization. For instance, Chi *et al.* (2021) and Lee *et al.* (2020) train a network on a low-resolution grid and then validate it on a finer resolution. Qian and Ye (2021) and Lunz *et al.* (2021) employ a dual-model approach to predict the PDE solution and the adjoint using a secondary network. Recent efforts have been made to incorporate the PDE for linear elasticity as a constraint within the loss function, akin to the methodology used in PINNs (Bolandi *et al.*, 2023; Faroughi *et al.*, 2022; Jeong *et al.*, 2023; Joglekar *et al.*, 2023; Lu, Pestourie, *et al.*, 2021). Furthermore, there are some publications centered around the utilization of DeepONets (Augenstein *et al.*, 2023; He *et al.*, 2023; Lu, Pestourie, *et al.*, 2022) and FNOs (Li, Huang, *et al.*, 2022) for solving the PDE. To our knowledge, there exists no publication that employs NOs as part of the SIMP algorithm. In He *et al.* (2023), the authors learn a PDE solution operator for linear elasticity using a DeepONet approach without enforcing gradient-consistency. We demonstrate that without the incorporation of a loss term that promotes gradient-consistency, NOs will likely encounter difficulties when integrated into an iterative gradient-based algorithm like SIMP.

Lastly, DL can also be utilized to reparameterize the density field (Deng & To, 2020; Hoyer *et al.*, 2019; Zehnder *et al.*, 2021; Zhang *et al.*, 2021). Since these models are mesh-independent, they can represent the density function at arbitrary resolutions. However, one usually still requires computationally demanding PDE evaluations for the training process.

## 3. Neural Operators

In this section, we will explore several different NOs that will serve as surrogates for SIMP's PDE solver. We will start with a preliminary overview, and then proceed to introduce both our U-Net FNO architecture and the concept of group equivariant NOs.

### 3.1. Preliminaries

NOs (Kovachki *et al.*, 2021) aim to learn an infinite-dimensional mapping from a finite dataset of input-output pairs. Let $n_{\mathcal{D}}, n_{\mathcal{A}}, n_{\mathcal{U}} \in \mathbb{N}$ and let $\mathcal{D} \subset \mathbb{R}^{n_{\mathcal{D}}}$ be bounded and closed. We formulate the learning problem on real-valued Banach spaces $\mathcal{A} := \mathcal{A}(\mathcal{D}; \mathbb{R}^{n_{\mathcal{A}}})$ and $\mathcal{U} := \mathcal{U}(\mathcal{D}; \mathbb{R}^{n_{\mathcal{U}}})$, with an operator $\mathcal{G}^\dagger : \mathcal{A} \to \mathcal{U}$ mapping between these two spaces. Given an i.i.d. sequence of data points $\{a_i, u_i\}_{i=1}^N \sim (\mathcal{A}, \mathcal{U})$ with $u_i = \mathcal{G}^\dagger(a_i)$, the objective is to approximate $\mathcal{G}^\dagger$ with a parameterized mapping $\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}$ with parameters $\theta$ from some parameter space $\Theta$.

The NO architecture, as proposed in Li, Kovachki, *et al.* (2020b), employs an iterative approach using a sequence of non-linear operators $\mathcal{G}_i$. These operators consist of linear integral operators followed by point-wise non-linearities. Specifically, for an input function $v_i$ at the ith layer, the computation of $\mathcal{G}_i : v_i \mapsto v_{i+1}$ is defined by

$$\mathcal{G}_i v_i(x) := \sigma\left(\mathcal{K}_\theta(v_i)(x) + W v_i(x)\right), \qquad \forall x \in \mathcal{D}_i,$$

where $\mathcal{K}_\theta$ is a linear operator parameterized by $\theta$, $W$ represents a linear transformation, and $\sigma$ is an element-wise non-linear activation function. We set $\mathcal{D}_0 = \mathcal{D}$ and impose that $\mathcal{D}_i \subset \mathbb{R}^{n_{\mathcal{D}}}$ is a bounded domain for all $i$.

We choose $\mathcal{K}_\theta$ to be a kernel integral transformation:

$$\mathcal{K}_\theta(v_i)(x) := \int \kappa_\theta(x, y) v_i(y)\, dy, \qquad \forall x \in \mathcal{D}_i,$$

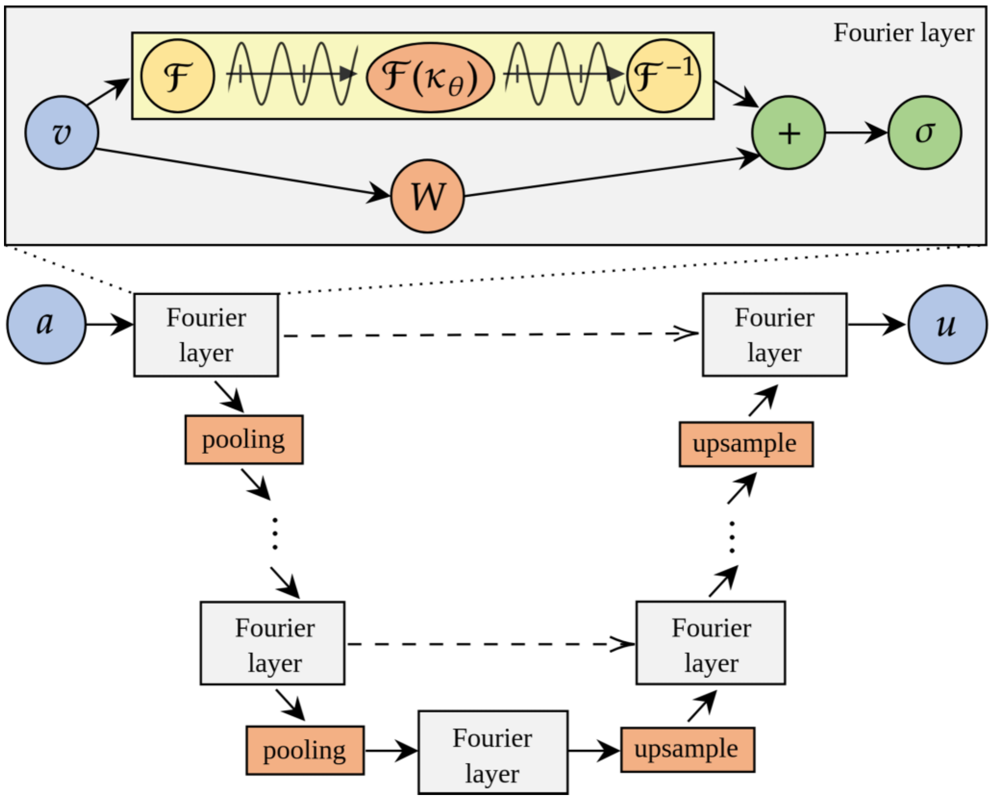


**Figure 1:** Our U-Net FNO model architecture. The architecture is the same as a regular U-Net with skip connections, but Fourier layers replace regular convolutions. Inside the Fourier layers, $\mathcal{F}$ denotes the Fourier transform, W is a linear bias term, and $\sigma$ is an element-wise activation function.

with $\kappa_\theta$ being a neural network parameterized by $\theta$. Here, $\kappa_\theta$ serves as a learnable kernel function. We assume translation equivariance and choose $\kappa_\theta(x, y) = \kappa_\theta(x - y)$. We then obtain

$$\mathcal{K}_\theta(v_i)(x) = (\kappa_\theta * v_i)(x), \qquad \forall x \in \mathcal{D}_i, \tag{1}$$

which in many PDE-related applications is a natural choice from the perspective of fundamental solutions (Renardy & Rogers, 2006).

Typically, NOs include a shallow fully connected neural network $P$ at the start, acting as a local lifting operator to convert the input into a higher dimensional representation, i.e., $v_0(x) = P(a(x))$ for $a \in \mathcal{A}$. Similarly, a local transformation $Q$ is applied to the output of the last layer, acting as a projection operator onto the desired output function space (Kovachki *et al.*, 2021).

FNOs (Li, Kovachki, *et al.*, 2020a) efficiently parameterize the kernel function $\kappa_\theta$ by exploiting the Fourier space representation of $v_i$. From equation (1) and the convolution theorem we derive

$$\mathcal{K}_\theta(v_i)(x) = \mathcal{F}^{-1}\left(\mathcal{F}(\kappa_\theta)\cdot\mathcal{F}(v_i)\right)(x), \qquad \forall x \in \mathcal{D}_i,$$

where $\mathcal{F}$ denotes the Fourier transformation, efficiently calculated using the Fast Fourier Transform (FFT). Assuming periodicity of $\kappa_\theta$, a Fourier series expansion yields discrete Fourier modes. By truncating the series at a maximum of $k_{\max}$ modes, $\mathcal{F}(\kappa_\theta)$ is directly parameterized as a learnable weight tensor with $k_{\max}$ channels. This truncation acts as a low-pass filter, leading to relatively smooth outputs (Augenstein *et al.*, 2023).

If the input domain is uniformly discretized with $m$ sensor points, the vector $\mathcal{F}(\kappa_\theta)$ contains a maximum of $k_{\max} \ll m$ modes. The convolution can then be performed with quasi-linear complexity, requiring $\mathcal{O}(m \log m)$ operations using the FFT. This marks a substantial improvement compared with the $\mathcal{O}(m^2)$ operations needed to compute the integral in equation (1) using a quadrature rule (Boullé & Townsend, 2023). In practice, one can observe that by limiting the number of Fourier modes to $k_{\max} \ll m$, the accuracy of the approximation is not significantly compromised, especially when the input and output functions exhibit smoothness, resulting in rapid decay of the coefficients in their Fourier basis representation. This reduction in Fourier modes contributes to lowering both the computational and training complexity of NOs.

## 3.2. U-Net Fourier Neural Operators

Since their introduction in 2015 (Ronneberger *et al.*, 2015), U-Nets have consistently set the standard in numerous fields of DL. Over the years, the U-Net architecture has seen significant refinements and adaptations (Siddique *et al.*, 2021), marked by notable advancements such as the integration of attention mechanisms (Oktay *et al.*, 2018; Vaswani *et al.*, 2017) and the incorporation of residual connections (Gu *et al.*, 2019; Tong *et al.*, 2018). We introduce the U-Net FNO as a neural network architecture that combines the U-Net's ability to capture spatial features with the FNO's proficiency in modeling operators based on Fourier space representations.

A U-Net is a deep neural network architecture commonly used for image segmentation and related tasks. It consists of an encoding path that gradually reduces spatial dimensions while increasing the number of feature channels. It is followed by a decoding path that upsamples the features back to the original spatial dimensions. Skip connections are employed between the encoding and decoding paths to help retain crucial spatial information. Instead of directly applying convolutional operations, the U-Net FNO utilizes convolutions in Fourier space. This involves

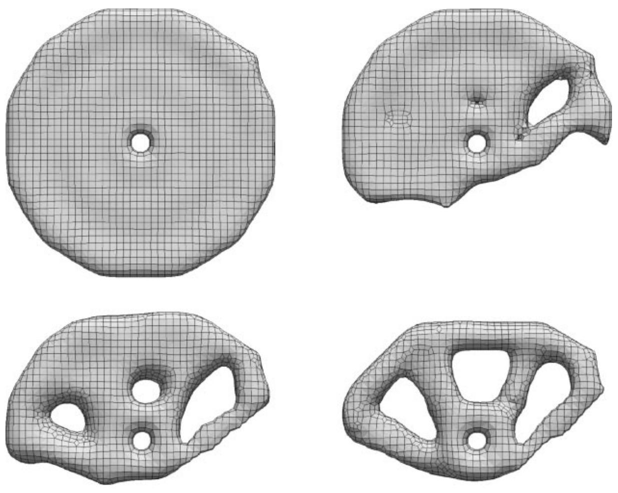


**Figure 2:** Intermediate SIMP solutions at iterations 0, 5, 15, and 60 (from top left to bottom right). The densities are defined on a Cartesian voxel grid and binarized and smoothed for visualization using Taubin smoothing (Taubin, 1995).

applying the Fourier transform, element-wise multiplication in the frequency domain, and then transforming back using the inverse Fourier transform. For a schematic illustration of the U-Net FNO architecture, see Fig. 1. Note that our approach differs from the U-FNO proposed in Wen *et al.* (2022), where the authors append a conventional U-Net path in each Fourier layer without an overall U-Net structure.

### 3.3. Equivariant NOs

In many machine learning tasks, we possess prior knowledge about the equivariances our predictor should have. Including such knowledge directly into the model can significantly enhance learning by freeing up model capacity for other factors of variation (Dittmer *et al.*, 2022; Weiler *et al.*, 2018). Equivariance is the property of a function or operator to commute with the actions of a symmetry group that acts on both its domain and range. For a given transformation group G, we call an operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ *(G-)equivariant* if and only if

$$\mathcal{G}\left(T_g^{\mathcal{A}}[a]\right) = T_g^{\mathcal{U}}[\mathcal{G}(a)] \qquad \forall g \in G, a \in \mathcal{A}, \tag{2}$$

where $T_g^{\mathcal{A}}$ and $T_g^{\mathcal{U}}$ denote linear *group actions* in the corresponding spaces $\mathcal{A}$ and $\mathcal{U}$ (Cohen & Welling, 2016). In simpler terms, when we transform an input function $a \in \mathcal{A}$ by a transformation $g \in G$ and then pass it through $\mathcal{G}$, it should yield the same result as first applying $\mathcal{G}$ to $a$ and then transforming the output $\mathcal{G}(a)$. By construction, convolutional operators are inherently translation equivariant. However, by choosing G accordingly, we can additionally enforce equivariance with respect to any desired group transformation.

Since G is a group, it encompasses an identity transformation and a unique inverse $g^{-1}$ for each $g \in G$. Consequently, we can rephrase the initial equivariance condition (2) as follows:

$$\mathcal{G}(a) = T_{g^{-1}}^{\mathcal{U}}\left[\mathcal{G}\left(T_g^{\mathcal{A}}[a]\right)\right].$$

This formulation enables us to implement equivariance through a technique known as group averaging (Murphy *et al.*, 2018; Puny *et al.*, 2021) by defining an equivariance wrapper (Dittmer *et al.*, 2022) $\mathcal{G}^G$ as

$$\mathcal{G}^G(a) := \frac{1}{|G|}\sum_{g\in G} T_{g^{-1}}^{\mathcal{U}}\left[\mathcal{G}\left(T_g^{\mathcal{A}}[a]\right)\right].$$

$\mathcal{G}^G$ is equivariant with respect to the transformation group G, as can be easily demonstrated (Dittmer *et al.*, 2022).

## 4. SIMP Trajectory Learning

In this section, we apply NOs to TO. To our knowledge, we are pioneering using NOs to replace the PDE solver within the SIMP method.

SIMP uses the notion of a material density function over a predefined design domain that takes values in the interval [0, 1]. This density function indicates the extent to which material is present in each point of the design domain. SIMP iteratively optimizes and updates this density function to achieve the user's objectives (see Figs 2 and 3). We want to make clear that SIMP operates on a discrete voxel grid and can therefore – unlike Level-Set Methods (M. Y. Wang *et al.*, 2003) and Moving Morphable Components (Guo *et al.*, 2014; Rostami *et al.*, 2023) – create non-smooth boundaries in the resulting design.

SIMP's most common optimization objective is compliance, which can be interpreted as a measure of structural integrity. Compliance is defined as the scalar product $\langle u, f\rangle$ between displacements $u$ and external forces $f$. We obtain the displacements as the solution of the PDE for linear elasticity given by

$$-\left[\lambda(\rho) + \mu(\rho)\right]\nabla(\nabla \cdot u) - \mu(\rho)\Delta u = f. \tag{3}$$

In this equation, $\rho$ represents the density distribution, and $\lambda$ and $\mu$ are the Lamé parameters that define the material's properties.

SIMP's main bottleneck is its reliance on solving the PDE for linear elasticity (3) in each iteration to determine the compliance based on the current density configuration $\rho$. We aim to alleviate this bottleneck by learning a NO. Let $\mathcal{U} = L_2(\mathcal{D}; \mathbb{R}^3)$ and $\mathcal{A} = L_2(\mathcal{D}; [0, 1]) \times \mathcal{U} \subset L_2(\mathcal{D}; \mathbb{R}^4)$. We learn the PDE solution operator:

$$\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}$$
$$(\rho, f) \mapsto u,$$

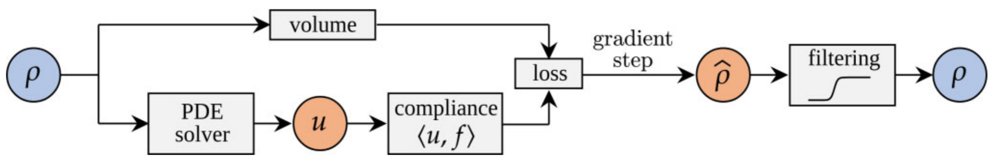


**Figure 3:** Visualization of an iteration of the SIMP algorithm for TO. Each iteration involves solving the PDE for linear elasticity to determine the displacements $u$. Subsequently, we calculate the compliance, a common objective in TO. Gradient-based optimization leads to an updated density distribution, which we filter using a smoothed Heaviside function to discourage non-binary densities.

where $\rho \in L_2(\mathcal{D}; [0, 1])$ is the material density function, $f \in \mathcal{U}$ are the external forces, and $u \in \mathcal{U}$ is the displacement field, which constitutes the solution of the PDE.

We train $\mathcal{G}_\theta$ in a supervised fashion by minimizing the loss function:

$$\mathcal{L}(u_{\text{pred}}, u) := \left\| u_{\text{pred}} - u \right\|_2^2 + \alpha \left\| \partial_\rho \langle u_{\text{pred}}, f \rangle - \partial_\rho \langle u, f \rangle \right\|_2^2, \tag{4}$$

where $u_{\text{pred}} = \mathcal{G}_\theta(\rho, f)$ is the model prediction, $\alpha > 0$ is a weight factor, and $\partial_\rho$ is the partial derivative with respect to the density $\rho$. We include the second summand in equation (4) because accurate function approximations do not ensure accurate gradient approximations, i.e., gradient-consistency. Good gradient approximations are essential for approximating the ground truth SIMP optimization trajectory for compliance minimization, as we will demonstrate in Section 5. While our approach shares similarities with the ones introduced in Lunz *et al.* (2021) and Qian and Ye (2021), we enforce gradient-consistency more directly and avoid using a secondary network for gradient predictions.

## 5. Experiments

We will now conduct a comparative analysis of various NOs, as introduced in Section 3. We compare a standard U-Net architecture with two FNOs and with our U-Net FNO approach (see Section 3.2). For all models, we consider both equivariant and non-equivariant cases. As we will use the SELTO disc dataset (Dittmer *et al.*, 2022, 2023) for our experiments, we follow Dittmer *et al.* (2022) and choose the transformation group G as the group of 2D rotations and reflections.

### 5.1. Setup and training

We will now describe the numerical setup of our experiments. We begin with the architectures used. The U-Net consists of three encoding and three decoding blocks with skip connections. In each block, we use instance normalization (Ulyanov *et al.*, 2016) and Rectified Linear Unit (ReLU) activation functions. We perform all convolutions with a $3 \times 3 \times 3$ filter. For the Fourier-based models, we truncate the number of Fourier modes in x, y, and z direction to $k_{\max} = (6, 6, 3)$. We employ a similar number of learnable parameters of around 500.000 for all models. We run all training on a GeForce RTX 2080 Ti GPU and determine the batch sizes individually based on the VRAM capacity. This leads to batch sizes of 70 for models with equivariance and 140 for models without equivariance. All experiments are implemented in PyTorch (Paszke *et al.*, 2017) using the DL4TO (Erzmann *et al.*, 2023) and the NO (Li, Kovachki, *et al.*, 2020a) software libraries.

For the construction of our datasets, we use subsets of the SELTO disc dataset (Dittmer *et al.*, 2022, 2023), which contains almost 10.000 3D TO problems and associated ground truth densities defined on a $39 \times 39 \times 4$ grid. We conduct training on subsets containing 10, 50, 100, and 200 different TO problems. This enables us to assess sample efficiency (Dittmer *et al.*, 2022), i.e., model performance when trained on smaller training subsets. We run $i_{\max} = 60$ iterations of SIMP for each problem and save the intermediate and final densities, along with their respective displacement tensors that we obtain from solving the PDE for linear elasticity (3). This leads to training sets containing 600, 3000, 6000, and 12000 samples, respectively.

Throughout the experiments, we employ the Adam optimizer (Kingma & Ba, 2014) with a learning rate of $10^{-4}$. We train all models until their improvement on the validation set has not improved for 100 epochs and then pick the model corresponding to the best validation epoch. This stopping criterion results in models being trained for 100 to 1000 epochs, leading to training times between 2 and 14 hours.

### 5.2. Evaluation

After training, we run SIMP with the learned PDE solvers on all problems from the test dataset and compare their performance to the classical SIMP that relies on the Finite Differences Method (FDM). In order to quantify the performance of the learned SIMP algorithm, we use the following two approaches:

(i) We evaluate the quality of SIMP's final predictions via the criteria intersection over union (IoU), fail percentage (fail%, Dittmer *et al.*, 2022), and mean compliance. IoU is the most common region-based metric for binary semantic segmentation. Fail% is the fraction of failed model predictions. We consider a prediction failed if the von Mises stress in any voxel exceeds the limiting yield stress by more than 10% or if at least one voxel with a load case does not connect to one containing Dirichlet conditions. For the computation of the mean compliance we normalize the forces $f$ with the infinity norm $\|f\|_\infty$. Both fail% and compliance are metrics that evaluate the structural integrity of a design. While we favor fail% for its interpretability, the inclusion of compliance in our analysis is motivated by its widespread recognition as a conventional evaluation criterion in the field. As such, both metrics have been incorporated into our investigation to uphold methodological consistency with established scholarly standards.
Another typical evaluation criterion in the DL for TO literature is the (balanced) accuracy metric. However, we found that in most cases, IoU yields comparable values but, in general, reflects the quality of its input more appropriately when used across different datasets (Dittmer *et al.*, 2022). Given that we can interpret the binary densities of our structures as segmentation masks, IoU's applicability is not surprising as it is the most common metric for semantic segmentation.

(ii) We evaluate deviations from the ground truth SIMP optimization trajectory by computing the weighted mean absolute error (WMAE) for each intermediate SIMP iteration. For iteration $i \in \mathbb{N}, 1 < i \leq i_{\max}$, we define the WMAE as follows:

$$\text{WMAE}(i) := \frac{c_i}{N n_e} \sum_{k=1}^{N} \sum_{e=1}^{n_e} \left| \rho_{\text{pred},k}^{e,i} - \rho_k^{e,i} \right|. \tag{5}$$

Here, $\rho_{\text{pred},k}^{e,i}$ and $\rho_k^{e,i}$ denote the predicted and the ground truth density for problem $k$ at SIMP iteration $i$ at voxel element $e$. $N$ and $n_e$ are the number of test samples and elements, respectively. The weighting factor $c_i := \frac{1}{N n_e} \sum_{k=1}^{N} \sum_{e=1}^{n_e} \left| (\rho_k^{e,i_{\max}} - \rho_k^{e,i_{\max}-1}) / (\rho_k^{e,i} - \rho_k^{e,i-1}) \right|$ improves the interpretability by accounting for large changes in the density of the ground truth SIMP trajectory and is scaled such that $c_{i_{\max}} = 1$. For example, to explain the role of $c_i$ we can consider a toy example with a uniform density that changes over the course of iterations, $(\rho_k^{e,i})_{i=1}^5 = (1, 0.5, 0.3, 0.2, 0.25)$ for all voxel elements $e$. In that case, the weighting factors $c_i = (0.1, 0.25, 0.5, 1)$ reduce the contribution of errors due to large relative changes in ground truth density to the WMAE, which leads to less distorted and more intuitive curves with a smoother overall shape.

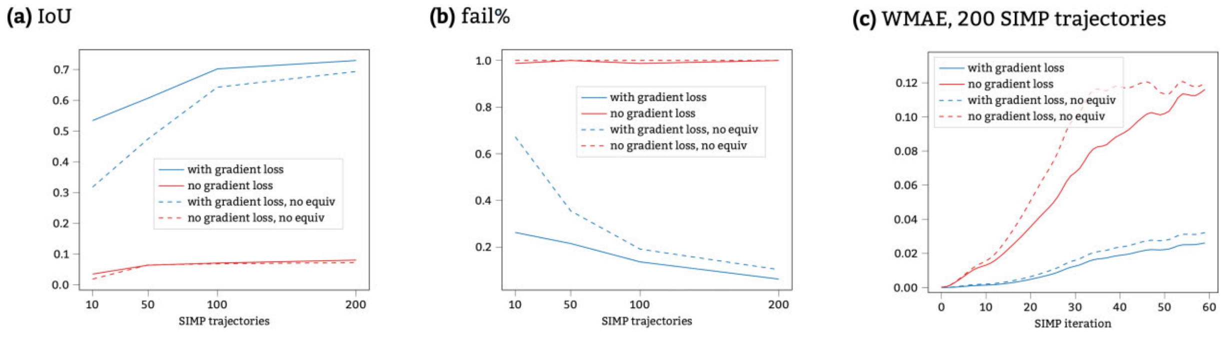


**Figure 4:** IoU, fail%, and WMAE of U-Net FNOs trained with and without the compliance gradient summand added to the loss function (4). Models without equivariance are depicted with dashed lines. We can observe that the gradient loss summand is necessary, and training without it fails entirely.

**Table 1:** IoU, fail%, and mean compliance scores of models with and without equivariance, trained on subsets of 10 to 200 SIMP trajectories. High IoU values are desirable, while lower values are preferable for fail% and compliance.

| # SIMP trajectories | # Training samples | U-Net | | | FNO | | | U-FNO | | | U-Net FNO (ours) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | IoU | Fail% | Compl | IoU | Fail% | Compl | IoU | Fail% | Compl | IoU | Fail% | Compl |
| **(a) Without equivariance** | | | | | | | | | | | | | |
| 10 | 600 | 0.35 | 67.50 | 3.77 | 0.29 | 73.75 | 4.52 | 0.30 | 68.25 | 4.12 | 0.32 | 67.25 | 3.49 |
| 50 | 3000 | 0.43 | 41.50 | 1.43 | 0.37 | 62.50 | 3.71 | 0.38 | 58.10 | 3.11 | 0.47 | 35.50 | 1.21 |
| 100 | 6000 | 0.59 | 21.25 | 1.13 | 0.38 | 46.25 | 2.60 | 0.48 | 43.50 | 2.88 | 0.64 | 19.10 | 1.10 |
| 200 | 12000 | 0.67 | 11.25 | **0.87** | 0.49 | 31.50 | 1.82 | 0.53 | 22.27 | 1.51 | **0.69** | **10.32** | 0.92 |
| **(b) With equivariance** | | | | | | | | | | | | | |
| 10 | 600 | 0.51 | 28.50 | 2.29 | 0.44 | 64.25 | 2.04 | 0.43 | 49.52 | 2.93 | 0.53 | 26.25 | 1.78 |
| 50 | 3000 | 0.61 | 28.75 | 1.40 | 0.48 | 46.50 | 1.38 | 0.55 | 32.55 | 1.06 | 0.63 | 21.50 | 1.03 |
| 100 | 6000 | 0.66 | 18.75 | 1.36 | 0.58 | 23.42 | 0.91 | 0.62 | 21.40 | 0.98 | 0.70 | 13.64 | 0.76 |
| 200 | 12000 | 0.71 | 7.50 | 0.63 | 0.60 | 13.50 | 1.12 | 0.68 | 14.64 | 0.82 | **0.73** | **6.25** | **0.56** |

It is important to underscore that our objective is not centered on enhancing the predictive accuracy of the SIMP method. Instead, our focus is directed towards a speed-up of the classical SIMP algorithm, without sacrificing significant amounts of accuracy.

## 5.3. Results

We now examine the outcomes of our numerical experiments. Specifically, we compare our U-Net FNO to a U-Net with conventional convolutions, with a standard FNO and with the U-FNO proposed by Wen *et al.* (2022). Furthermore, we investigate the impact of group equivariance on our models.

In Fig. 4, we show that including the gradient summand in the loss function (4) is indeed necessary. Running SIMP with a NO trained without the gradient summand fails entirely, with fail% scores of around 100%. Including the loss summand leads to more gradient-consistent predictions that generalize well on the test data. This finding holds significant relevance for future research endeavors within the field.

Table 1 presents the IoU, fail%, and mean compliance scores of different models trained on datasets containing 10, 50, 100, and 200 SIMP trajectories. It is evident that our U-Net FNO model outperforms both the U-Net and the FNO models. This is true for all training dataset sizes. Moreover, incorporating group equivariance enhances performance significantly. For models trained on small datasets, the effect is especially pronounced. This also becomes apparent when comparing individual model predictions, as displayed in Fig. 5. We can observe that equivariance improves the overall quality of predictions and can even find its own solutions with a slightly different topology than the ground truth. We believe that this showcases the strong generalization capabilities of our equivariant models.

The impact of group equivariance and network architecture also becomes evident when analyzing deviations of the learned SIMP optimization trajectories from the ground truth trajectories. Figure 6 illustrates these deviations, with the WMAE criterion (5) serving as our evaluation metric. Notably, the equivariant U-Net FNO model exhibits the least divergence, indicating its superior ability to approximate the ground truth optimization trajectories. However, there is a trade-off between quality and speed, since including the gradient loss summand and equivariance both lead to longer inference times, as demonstrated in Table 2. Nevertheless, all NO models provide at least a fourfold improvement in inference speed compared with the classical SIMP method. This advantage becomes even more pronounced on finer grids, since the computational complexity of the FDM scales quadratically with respect to the number of total voxel elements, while it only scales linearly for NOs. For instance, a single FDM solution on a very fine voxel grid with 160 000 voxel elements takes almost 5 minutes, while a NO takes less than a second.

Despite the promising results presented in this section, it should be noted that the application of DL for TO has its limitations and drawbacks compared with classical PDE solvers. Considering the 2 to 14 hours of training time, coupled with the prerequisite of available training data, we believe that the use of DL in TO is particularly advantageous in two primary scenarios. Firstly, when there is a need for numerous similar components that demand individual optimization – e.g., encountered in aerospace

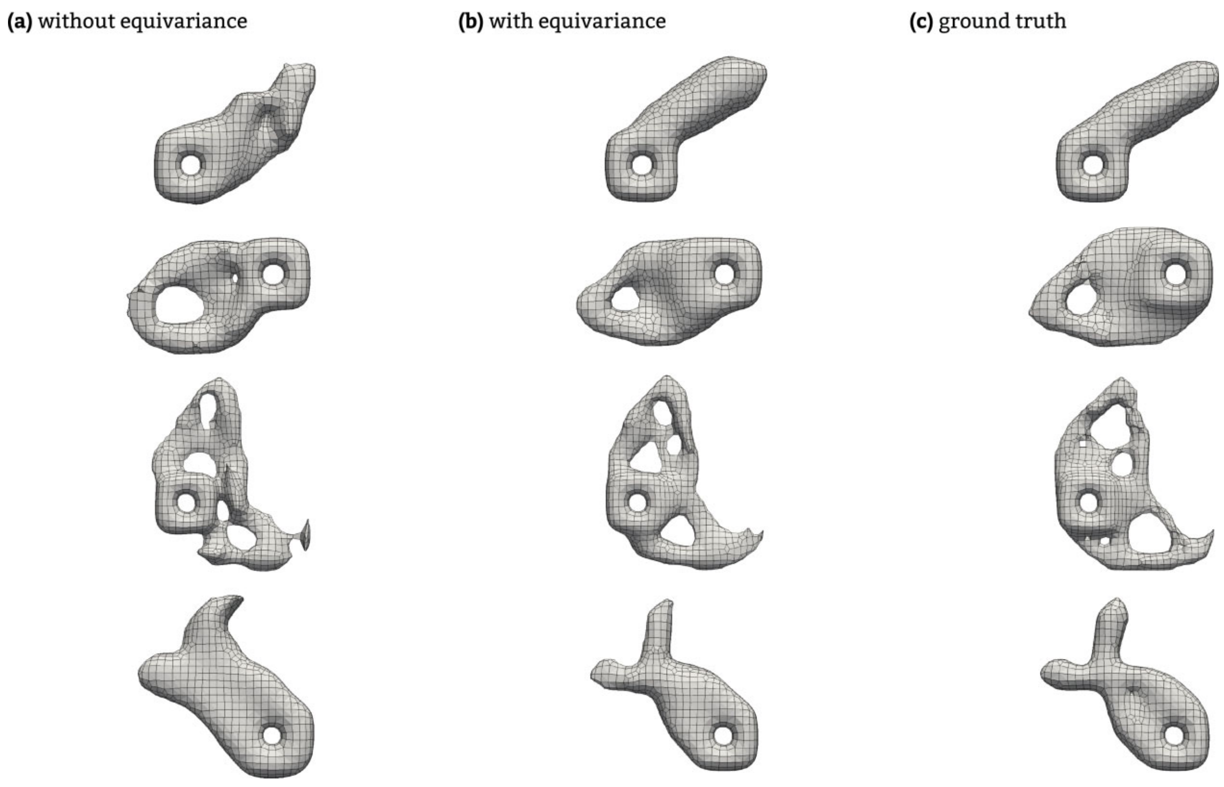


**Figure 5:** U-Net FNO predictions and ground truth solutions from the test dataset, trained on 200 SIMP trajectories. We can see that enforcing equivariance leads to better-looking samples.

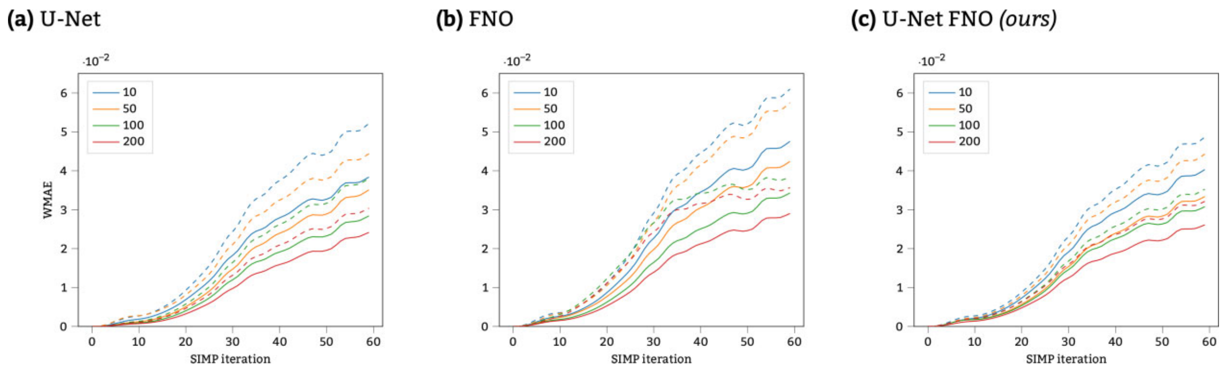


**Figure 6:** Comparison of WMAE (5) for the U-Net, FNO, and our U-Net FNO for each SIMP iteration, averaged over all samples from the test dataset. Models without equivariance are depicted with dashed lines. Different colors signify models trained on 10, 50, 100, and 200 SIMP trajectories. For visualization purposes, all graphs are slightly smoothed using Gaussian smoothing. Notably, the equivariant U-Net FNO model exhibits the lowest WMAE scores, indicating its superior ability to approximate the ground truth optimization trajectories.

**Table 2:** Average inference run times (in seconds) of SIMP with the classical FDM solver and NOs, executed on an Intel Xeon Silver 4210 CPU.

| PDE solver | FDM | Without equivariance | | | | With equivariance | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | U-Net | FNO | U-FNO | U-Net FNO | U-Net | FNO | U-FNO | U-Net FNO |
| duration | 75 | 2 | 3 | 3 | 3 | 15 | 18 | 17 | 19 |

engineering, where a single space rocket necessitates optimization for thousands of mounting brackets. In such cases, the importance of sample-efficient models cannot be overstated, as they contribute to minimizing the required datasets. Secondly, DL becomes advantageous when the training duration can be outsourced. Despite potentially lengthy training times, models can be trained without direct supervision, e.g., over the weekend. This presents a more efficient alternative for engineering companies compared with the intermittent waiting periods during engineers' working hours inherent to classical SIMP methodologies.

## 6. Conclusions

We have demonstrated the potential of DL surrogate models, particularly NOs, to enhance the efficiency of data-driven PDE solving significantly. We introduced and systematically compared several NOs within the context of 3D TO. We demonstrated their capability to replace the PDE solver in the widely used SIMP method. Crucially, we contributed a novel loss function to ensure their gradient-consistency throughout the SIMP optimization process – a key element as without it, NOs fail to work at all within SIMP.

Further, incorporating group equivariance into our approach has proven highly advantageous by diminishing the dependence on large training datasets. Moreover, we introduced and empirically validated the effectiveness of the U-Net FNO. This novel architecture integrates the spatial feature-capturing prowess of U-Nets with the advanced representation capabilities of FNOs.

In future research, we would like to extend the validation of our findings by applying them to diverse datasets. The exploration of multi-resolution data holds particular significance, enabling a thorough evaluation of the models' generalization capabilities across various scales. Additionally, employing a finer grid could offer valuable insights. Enhancing sample efficiency remains a priority, and one avenue to explore involves incorporating additional physical information about the PDE into the training pipeline. For instance, this could be realized through semi-supervised training by incorporating a PINN loss term that leverages information from forward evaluations of the PDE system.

## Conflict of interest statement

None declared.

## References

Aage, N., Andreassen, E., & Lazarov, B. S. (2015). Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization*, **51**(3), 565–572. https://doi.org/10.1007/s00158-014-1157-0.

Aage, N., Andreassen, E., Lazarov, B. S., & Sigmund, O. (2017). Giga-voxel computational morphogenesis for structural design. *Nature*, **550**(7674), 84–86. https://doi.org/10.1038/nature23911.

Abueidda, D. W., Koric, S., & Sobh, N. A. (2020). Topology optimization of 2D structures with nonlinearities using deep learning. *Computers & Structures*, **237**, 106283. https://doi.org/10.1016/j.compstruc.2020.106283.

Augenstein, Y., Repän, T., & Rockstuhl, C. (2023). Neural operator-based surrogate solver for free-form electromagnetic inverse design. *ACS Photonics*, **10**(5), 1547–1557. https://doi.org/10.1021/acsphotonics.3c00156.

Banga, S., Gehani, H., Bhilare, S., Patel, S., & Kara, L. B. (2018). 3D topology optimization using convolutional neural networks. *preprint arXiv:1808.07440*, https://doi.org/10.48550/arXiv.1808.07440.

Bendsøe, M. P., & Sigmund, O. (2003). *Topology optimization: Theory, methods, and applications*. Springer Science & Business Media.

Bolandi, H., Sreekumar, G., Li, X., Lajnef, N., & Boddeti, V. N. (2023). Physics informed neural network for dynamic stress prediction. *Applied Intelligence*, **53** 1–16. https://doi.org/10.48550/arXiv.2211.16190.

Boullé, N., & Townsend, A. (2023). A mathematical guide to operator learning. *preprint arXiv:2312.14688*, https://doi.org/10.48550/arXiv.2312.14688.

Challis, V. J., Roberts, A. P., & Grotowski, J. F. (2014). High resolution topology optimization using graphics processing units (GPUs). *Structural and Multidisciplinary Optimization*, **49**(2), 315–325. https://doi.org/10.1007/s00158-013-0980-z.

Chen, T., & Chen, H. (1995). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, **6**(4), 911–917. https://doi.org/10.1109/72.392253.

Chi, H., Zhang, Y., Tang, T. L. E., Mirabella, L., Dalloro, L., Song, L., & Paulino, G. H. (2021). Universal machine learning for topology optimization. *Computer Methods in Applied Mechanics and Engineering*, **375**, 112739. https://doi.org/10.1016/j.cma.2019.112739.

Cohen, T., & Welling, M. (2016). Group equivariant convolutional networks. In *Proceedings of the International Conference on Machine Learning* (pp. 2990–2999). PMLR.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, **92**(3), 88. https://doi.org/10.1007/s10915-022-01939-z.

Deng, H., & To, A. C. (2020). Topology optimization based on deep representation learning (DRL) for compliance and stress-constrained design. *Computational Mechanics*, **66**(2), 449–469. https://doi.org/10.1007/s00466-020-01859-5.

Dittmer, S., Erzmann, D., Harms, H., & Maass, P. (2022). SELTO: Sample-efficient learned topology optimization. *preprint arXiv:2209.05098*, https://doi.org/10.48550/arXiv.2209.05098.

Dittmer, S., Erzmann, D., Harms, H., Falck, R., & Gosch, M. (2023). *SELTO dataset*. Zenodo. https://doi.org/10.5281/zenodo.7034898.

Erzmann, D., Dittmer, S., Harms, H., & Maaß, P. (2023). DL4TO: A deep learning library for sample-efficient topology optimization. In *Proceedings of the 6th International Conference on Geometric Science of Information* (pp. 543–551). Springer.

Faroughi, S. A., Pawar, N., Fernandes, C., Das, S., Kalantari, N. K., & Mahjour, S. K. (2022). Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing. *preprint arXiv:2211.07377*, https://doi.org/10.48550/arXiv.2211.07377.

Gu, Z., Cheng, J., Fu, H., Zhou, K., Hao, H., Zhao, Y., Zhang, T., Gao, S., & Liu, J. (2019). CE-Net: Context encoder network for 2D medical image segmentation. *IEEE Transactions on Medical Imaging*, **38**(10), 2281–2292. https://doi.org/10.1109/TMI.2019.2903562.

Guo, X., Zhang, W., & Zhong, W. (2014). Doing topology optimization explicitly and geometrically–A new moving morphable components based framework. *Journal of Applied Mechanics*, **81**(8), 081009. https://doi.org/10.1115/1.4027609.

Ha, S.-H., & Cho, S. (2005). Topological shape optimization of heat conduction problems using level set approach. *Numerical Heat Transfer, Part B: Fundamentals*, **48**(1), 67–88. https://doi.org/10.1080/10407790590935966.

He, J., Koric, S., Kushwaha, S., Park, J., Abueidda, D., & Jasiuk, I. (2023). Novel DeepONet architecture to predict stresses in elastoplastic structures with variable complex geometries and loads. *Computer Methods in Applied Mechanics and Engineering*, **415**, 116277. https://doi.org/10.1016/j.cma.2023.116277.

Hoyer, S., Sohl-Dickstein, J., & Greydanus, S. (2019). Neural reparameterization improves structural optimization. *preprint arXiv:1909.04240*, https://doi.org/10.48550/arXiv.1909.04240.

Huang, S., Feng, W., Tang, C., & Lv, J. (2022). Partial differential equations meet deep neural networks: A survey. *preprint arXiv:2211.05567*, https://doi.org/10.48550/arXiv.2211.05567.

Hwang, J.-T., Hong, S.-Y., & Song, J.-H. (2023). Automated topology design to improve the susceptibility of naval ships using geometric deep learning. *Journal of Computational Design and Engineering*, **10**(2), 794–808. https://doi.org/10.1093/jcde/qwad023.

Jeong, H., Bai, J., Batuwatta-Gamage, C. P., Rathnayaka, C., Zhou, Y., & Gu, Y. (2023). A physics-informed neural network-based topology optimization (PINNTO) framework for structural optimization. *Engineering Structures*, **278**, 115484. https://doi.org/10.1016/j.engstruct.2022.115484.

Joglekar, A., Chen, H., & Kara, L. B. (2023). DMF-TONN: Direct mesh-free topology optimization using neural networks. *Engineering with Computers*, 1–14. https://doi.org/10.1007/s00366-023-01904-w.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *preprint arXiv*:1412.6980, https://doi.org/10.48550/arXiv.1412.6980.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2023). Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, **24**, 1–97. http://jmlr.org/papers/v24/21-1524.html.

Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, **9**(5), 987–1000. https://doi.org/10.1109/72.712178.

Lee, S., Kim, H., Lieu, Q. X., & Lee, J. (2020). CNN-based image recognition for topology optimization. *Knowledge-Based Systems*, **198**, 105887. https://doi.org/10.1016/j.knosys.2020.105887.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020a). Fourier neural operator for parametric partial differential equations. *preprint arXiv*:2010.08895, https://doi.org/10.48550/arXiv.2010.08895.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020b). Neural operator: Graph kernel network for partial differential equations. *preprint arXiv*:2003.03485, https://doi.org/10.48550/arXiv.2003.03485.

Li, Z., Huang, D. Z., Liu, B., & Anandkumar, A. (2022). Fourier neural operator with learned deformations for PDEs on general geometries. *preprint arXiv*:2207.05209, https://doi.org/10.48550/arXiv.2207.05209.

Lu, L., Jin, P., & Karniadakis, G. E. (2019). DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *preprint arXiv*:1910.03193, https://doi.org/10.48550/arXiv.1910.03193.

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., & Johnson, S. G. (2021). Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, **43**(6), B1105–B1132. https://doi.org/10.48550/arXiv.2102.04626.

Lu, L., Pestourie, R., Johnson, S. G., & Romano, G. (2022). Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Physical Review Research*, **4**(2), 023210. https://doi.org/10.1103/PhysRevResearch.4.023210.

Lunz, S., Hauptmann, A., Tarvainen, T., Schonlieb, C.-B., & Arridge, S. (2021). On learned operator correction in inverse problems. *SIAM Journal on Imaging Sciences*, **14**(1), 92–127. https://doi.org/10.1137/20M1338460.

Martínez-Frutos, J., & Herrero-Pérez, D. (2016). Large-scale robust topology optimization using multi-GPU systems. *Computer Methods in Applied Mechanics and Engineering*, **311**, 393–414. https://doi.org/10.1016/j.cma.2016.08.016.

Murphy, R. L., Srinivasan, B., Rao, V., & Ribeiro, B. (2018). Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *preprint arXiv*:1811.01900, https://doi.org/10.48550/arXiv.1811.01900.

Nie, Z., Lin, T., Jiang, H., & Kara, L. B. (2021). TopologyGAN: Topology optimization using generative adversarial networks based on physical fields over the initial domain. *Journal of Mechanical Design*, **143**(3), 031715. https://doi.org/10.1115/1.4049533.

Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., Glocker, B., & Rueckert, D. (2018). Attention U-Net: Learning where to look for the pancreas. *preprint arXiv*:1804.03999, https://doi.org/10.48550/arXiv.1804.03999.

Park, S.-M., Park, S., Park, J., Choi, M., Kim, L., & Noh, G. (2021). Design process of patient-specific osteosynthesis plates using topology optimization. *Journal of Computational Design and Engineering*, **8**(5), 1257–1266. https://doi.org/10.1093/jcde/qwab047.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. In *Proceedings of the NIPS 2017 Workshop on Autodiff*.

Puny, O., Atzmon, M., Ben-Hamu, H., Misra, I., Grover, A., Smith, E. J., & Lipman, Y. (2021). Frame averaging for invariant and equivariant network design. *preprint arXiv*:2110.03336, https://doi.org/10.48550/arXiv.2110.03336.

Qian, C., & Ye, W. (2021). Accelerating gradient-based topology optimization design with dual-model artificial neural networks. *Structural and Multidisciplinary Optimization*, **63**(4), 1687–1707. https://doi.org/10.1007/s00158-020-02770-6.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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. https://doi.org/10.1016/j.jcp.2018.10.045.

Renardy, M., & Rogers, R. C. (2006). *An introduction to partial differential equations* (Vol. **13**). Springer Science & Business Media.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: Part III* (pp. 234–241). Springer.

Rostami, S. A. L., Kolahdooz, A., Chung, H., Shi, M., & Zhang, J. (2023). Robust topology optimization of continuum structures with smooth boundaries using moving morphable components. *Structural and Multidisciplinary Optimization*, **66**, 121. https://doi.org/10.1007/s00158-023-03580-2.

Shin, S., Shin, D., & Kang, N. (2023). Topology optimization via machine learning and deep learning: A review. *Journal of Computational Design and Engineering*, **10**(4), 1736–1766. https://doi.org/10.1093/jcde/qwad072.

Siddique, N., Paheding, S., Elkin, C. P., & Devabhaktuni, V. (2021). U-Net and its variants for medical image segmentation: A review of theory and applications. *IEEE Access*, **9**, 82031–82057. https://doi.org/10.1109/ACCESS.2021.3086020.

Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, **375**, 1339–1364. https://doi.org/10.1016/j.jcp.2018.08.029.

Sosnovik, I., & Oseledets, I. (2019). Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling*, **34**(4), 215–223. https://doi.org/10.1515/rnam-2019-0018.

Tanyu, D. N., Ning, J., Freudenberg, T., Heilenkötter, N., Rademacher, A., Iben, U., & Maass, P. (2023). Deep learning methods for partial differential equations and related parameter identification problems. *Inverse Problems*, **39**, 103001. https://doi.org/10.48550/arXiv.2212.03130.

Taubin, G. (1995). Curve and surface smoothing without shrinkage. In *Proceedings of IEEE International Conference on Computer Vision* (pp. 852–857). IEEE.

Tong, G., Li, Y., Chen, H., Zhang, Q., & Jiang, H. (2018). Improved U-Net network for pulmonary nodules segmentation. *Optik*, **174**, 460–469. https://doi.org/10.1016/j.ijleo.2018.08.086.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *preprint arXiv*:1607.08022, https://doi.org/10.48550/arXiv.1607.08022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, **30**, https://doi.org/10.48550/arXiv.1706.03762.

Wang, D., Xiang, C., Pan, Y., Chen, A., Zhou, X., & Zhang, Y. (2022). A deep convolutional neural network for topology optimization with perceptible generalization ability. *Engineering Optimization*, **54**(6), 973–988. https://doi.org/10.1080/0305215X.2021.1902998.

Wang, M. Y., Wang, X., & Guo, D. (2003). A level set method for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering*, **192**(1–2), 227–246. https://doi.org/10.1016/S0045-7825(02)00559-5.

Weiler, M., Hamprecht, F. A., & Storath, M. (2018). Learning steerable filters for rotation equivariant CNNs. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 849–858. https://doi.org/10.48550/arXiv.1711.07289.

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., & Benson, S. M. (2022). U-FNO–An enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, **163**, 104180. https://doi.org/10.1016/j.advwatres.2022.104180.

Xiang, C., Wang, D., Pan, Y., Chen, A., Zhou, X., & Zhang, Y. (2022). Accelerated topology optimization design of 3D structures based on deep learning. *Structural and Multidisciplinary Optimization*, **65**(3), 99. https://doi.org/10.1007/s00158-022-03194-0.

Xue, L., Liu, J., Wen, G., & Wang, H. (2021). Efficient, high-resolution topology optimization method based on convolutional neural networks. *Frontiers of Mechanical Engineering*, **16**(1), 80–96. https://doi.org/10.1007/s11465-020-0614-2.

Xue, T., Beatson, A., Adriaenssens, S., & Adams, R. (2020). Amortized finite element analysis for fast PDE-constrained optimization. In *Proceedings of the International Conference on Machine Learning* (pp. 10638–10647). PMLR.

Yu, B., & E, W. (2018). The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, **6**(1), 1–12. https://doi.org/10.1007/s40304-018-0127-z.

Yu, Y., Hur, T., Jung, J., & Jang, I. G. (2019). Deep learning for determining a near-optimal topological design without any iteration. *Structural and Multidisciplinary Optimization*, **59**(3), 787–799. https://doi.org/10.1007/s00158-018-2101-5.

Zehnder, J., Li, Y., Coros, S., & Thomaszewski, B. (2021). NTOPO: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems*, **34**, 10368–10381. https://doi.org/10.48550/arXiv.2102.10782.

Zhang, Z., Li, Y., Zhou, W., Chen, X., Yao, W., & Zhao, Y. (2021). TONR: An exploration for a novel way combining neural network with topology optimization. *Computer Methods in Applied Mechanics and Engineering*, **386**, 114083. https://doi.org/10.1016/j.cma.2021.114083.

**Received:** October 16, 2023. **Revised:** March 14, 2024. **Accepted:** March 15, 2024