Chair of Computational Modeling and Simulation
TUM School of Engineering and Design
Technical University of Munich

# Accelerating Topology Optimization Using a Combination of Conventional Methods and Neural Networks

Scientific work to obtain the degree

**Master of Science (M.Sc.)**

at the TUM School of Engineering and Design
of the Technical University of Munich.

**Supervised by** PD Dr.-Ing. habil. Stefan Kollmannsberger
M.Sc. Leon Herrmann
Chair of Computational Modeling and Simulation

Christian Vogl
BMW Group

**Submitted by** Viola Muning Li [redacted]

**Submitted on** 29. April 2023

## Abstract

Topology optimization is a tool to design lightweight structures with maximum performance. However, their scope of application is limited due to the high computational cost associated with large-scale problems. This work presents new approaches that improve the efficiency of existing methods by combining conventional acceleration methods with neural networks. Based on the standard solid isotropic material with penalization method, a higher-order multi-resolution scheme is employed, which utilizes higher-order shape functions and different resolutions for the finite element mesh and the density mesh. A parametric model order reduction method is proposed to condense the internal modes out of the stiffness matrix. Additionally, neural networks are used to reparameterize the density field in the optimization process. In an alternative approach, a UNet++ is trained to predict near-optimal density distributions based on local stress and strain fields. All methods are applied to the Messerschmitt-Bolkow-Blohm beam, where the results are analyzed regarding their solution quality and efficiency. The use of convolutional neural networks for the density parameterization leads to improved designs and the total computation time is reduced by a factor of $2.5$. Similarly, designs with lower compliance and faster convergence are obtained by integrating predictions from the UNet++ into the optimization process.

# Contents

# Acronyms

| | |
|---|---|
| **CA** | Combined Approximations |
| **CNN** | Convolutional Neural Network |
| **CSR** | Compressed Sparse Row |
| **DOFs** | Degrees of Freedom |
| **FCNN** | Fully Connected Neural Network |
| **FE** | Finite Element |
| **GAN** | Generative Adversarial Network |
| **GPU** | Graphics Processing Unit |
| **MBB** | Messerschmitt-Bolkow-Blohm |
| **ML** | Machine Learning |
| **MOR** | Model Order Reduction |
| **NN** | Neural Network |
| **OC** | Optimality Criteria |
| **ReLU** | Rectified Linear Unit |
| **SIMP** | Solid Isotropic Material with Penalization |
| **TO** | Topology Optimization |

# Chapter 1

# Introduction

Topology Optimization (TO) is a structural design method for creating lightweight structures with maximum performance. It can be defined as an optimal material distribution problem, in which the TO algorithm determines the location where material should be present and where it can be removed. Essentially, it determines the number, position, and shape of holes within a given design space. The goal is to find the optimal material layout with respect to some objective function, certain boundary conditions, loads, and other constraints. Constraints may relate to the maximum amount of material in the final design or the maximum stress that occurs in the structure. The resulting lightweight structures not only benefit from a reduced material cost but can also reduce the impact on the environment by decreasing material waste and emissions from lightweight vehicles. TO is therefore increasingly used in almost every engineering field, including aerospace, automotive, and medical engineering. It is a powerful tool that is becoming an integral part of early design processes and that helps the designer make informed decisions during development.

Many different TO approaches have been introduced in the past few decades and their development is still an active field of research. The most popular method is probably the Solid Isotropic Material with Penalization (SIMP) method (Bendsøe, 1989; M. Zhou & Rozvany, 1991), a density-based approach in which the design domain is discretized by a fixed mesh. The topology is then found by varying the virtual densities of each element, which define whether the corresponding element represents a solid or a void. These densities are thus the design variables of the underlying optimization problem. In the most simple case, a minimum compliance problem is considered. Here, the goal is to minimize the compliance of the structure, which is equivalent to maximizing the global stiffness. The optimization problem is constrained by a simple volume constraint that prescribes the maximum amount of material allowed in the final structure.

As the computing power of modern computers continues to increase, TO is becoming increasingly accessible. Still, conventional TO methods are facing some challenges that limit their scope of application. One important aspect is the manufacturability of the optimized designs. Designs obtained from TO often exhibit organic or lattice-like shapes that are difficult to fabricate. In order to prevent a time-consuming post-processing by the engineer, manufacturing constraints can be integrated into the optimization process (Liu & Ma, 2016). Due to the recent advancement of additive manufacturing techniques and their capabilities to manufacture geometries with complex shapes, the focus has been shifted towards linking TO with additive manufacturing constraints (Jihong et al., 2021). This new trend presents new possibilities since more complex designs can be realized.

The second issue, which is also the focus of this thesis, is the high computational cost associated with TO. TO methods are based on iterative design updates, either with gradient-based methods, such as the Optimality Criteria (OC) method, or gradient-free methods, like genetic algorithms. In order to decide how the design variables should be updated, the current design performance needs to be evaluated. This is most commonly done with the Finite Element (FE) method. The FE method is an established numerical method that can return accurate and reliable results as long as the underlying problem is modeled with sufficient accuracy. The drawback is that a rather involved FE analysis has to be performed in every iteration and standard TO methods often require hundreds of iterations. This can lead to extremely high computational costs, especially when large-scale problems are considered. The optimization process of a three-dimensional, dynamic problem with millions of Degrees of Freedom (DOFs) can take days, if not weeks, on a supercomputer to finish. This is one of the main reasons why the use of TO is still limited in industry. Especially in the design and development phase, in which requirements on the structure change frequently, repeated TO is simply unfeasible.

Many approaches addressing the issue of the high computational cost have been introduced in the literature. The majority focuses on the FE analysis, as it constitutes most of the computation time. Common approaches make use of high-performance computing methods, Model Order Reduction (MOR) methods, or multi-grid methods. The overall goal is to reduce the number of time-consuming operations by introducing approximations or simpler models with less DOFs while still maintaining high accuracy. Nguyen et al. (2013) proposed to separate the FE mesh from the density mesh, making the FE analysis on a coarser mesh less expensive. They also used higher-order shape functions to increase the analysis quality. Similarly, Groen et al. (2017) integrated the finite cell method into TO and used static condensation to reduce the number of DOFs even further. Recent advances in Machine Learning (ML) methods have opened new opportunities also in the context of TO. In the past few years, many contributions in the literature focused on the potential of Neural Networks (NNs) to accelerate different parts of the TO process. One of the most popular approaches is to directly predict the final or near-final design without any expensive iterations (Seo & Kapania, 2023). This is often connected to an extremely large data set and poor generalization capabilities. In another approach, NNs are used to reparameterize the design variables of the optimization problem (Chandrasekhar & Suresh, 2021). Variants of both approaches will be investigated in this thesis.

The goal of this thesis is to develop a TO algorithm that can further improve the efficiency of existing methods. The proposed algorithms combine conventional acceleration methods with NNs, which can be mutually beneficial. As reference example, the conventional SIMP method will be used to solve the minimum compliance problem of the static, two-dimensional Messerschmitt-Bolkow-Blohm (MBB) beam, a well-known benchmark example. In this thesis, two different approaches are introduced. The first one combines the higher-order multi-resolution scheme proposed by Nguyen et al. (2013) with a parametric MOR method, and NNs are used to parameterize the material distribution (Chandrasekhar & Suresh, 2021; Hoyer et al., 2019). Both the use of Fully

Connected Neural Networks (FCNNs) and Convolutional Neural Networks (CNNs) will be investigated. It will be shown that the parametric MOR leads to disconnections and artifacts in the final design, but NNs can eliminate this problem, allowing the use of this highly efficient reduction method. The NN thus benefits from an efficient FE analysis, and the FE analysis benefits from the reparameterization by the NN. Especially the use of CNNs returns a wide range of promising results and the average computation time of the complete optimization process is significantly reduced to less than half of the reference time. In the second approach, a NN, more specifically a UNet++ (Z. ZHOU et al., 2018), is employed to accelerate the TO process by locally predicting near-optimal density values. These predictions are based on physical quantities of the structure, which are obtained from a FE analysis. In the scope of this thesis, the integration of the UNet++ is limited to the conventional SIMP method only. Although the total number of iterations can not be reduced with this approach, a faster convergence can be achieved, leading to improved designs already in the early stage of the optimization process. All methods will be applied to the MBB beam problem and the results will be analyzed with respect to the solution quality and the computation time.

The rest of the thesis is structured as follows: Chapter 2 gives a brief literature review of the main approaches for accelerating TO. In chapter 3 the basics of TO are summarized and the theoretical background of the conventional SIMP method is explained in detail. This will serve as reference method in the final evaluation. The method is then extended to a higher-order multi-resolution scheme in chapter 4. The corresponding stiffness matrix integration as well as the condensation methods will also be covered in this chapter. Chapter 5 is divided into two main parts. The first part covers the basics of NNs and their usage as a representation tool for the density distribution. In the second part, the proposed method to predict near-final density fields is explained. This section also covers the data generation process as well as the architecture and training of the UNet++. In chapter 6 the results obtained from the different methods are compared and evaluated with respect to their quality and efficiency. Finally, some concluding remarks and recommendations for further work are summarized in chapter 7.

# Chapter 2

# Literature Review

Since the introduction of the homogenization method for TO (BENDSØE & KIKUCHI, 1988), great effort has been made to develop further, more efficient approaches. These include for example, level set methods (ALLAIRE et al., 2004; M. Y. WANG et al., 2003), phase field methods (WALLIN et al., 2012) and methods using evolutionary algorithms (XIE & STEVEN, 1993). Over the past decades, the density approach, more specifically the SIMP method (BENDSØE, 1989; M. ZHOU & ROZVANY, 1991), has gained large popularity and is today the common method of choice. For a comparative review of different TO approaches the reader is referred to the paper of SIGMUND and MAUTE (2013).

In general, structural TO problems suffer from high computational costs associated with the repeated FE analysis in each iteration. Various techniques have been proposed to accelerate the procedure for large-scale problems. These can be roughly categorized into the following categories:

**High Performance Computing** The computational power of modern computers is increasing rapidly. However, large-scale problems can still lead to unreasonable optimization times, when only one processing unit is used. The use of multiple cores and parallel architectures is a powerful tool that can significantly improve computation times when the underlying problems are independent and parallelizable. BORRVALL and PETERSSON (2001) were likely the first ones to introduce parallel computing to TO. They used a combination of physical domain decomposition and an iterative solver to parallelize algorithms both in the FE part and in the optimization part. VEMAGANTI and LAWRENCE (2005) presented parallel algorithms based on the SIMP method and the OC method. The use of Graphics Processing Units (GPUs) for general problems in scientific computing is also becoming more and more popular due to their high computing capacity. The first contribution in the context of TO was by WADBRO and BERGGREN (2009), who utilized a GPU to accelerate the complete TO process of heat conduction problems. Later, MARTÍNEZ-FRUTOS et al. (2017) proposed a multi-granular GPU implementation based on the SIMP and OC method. The use of one GPU can be further extended to the use of multiple GPUs (HERRERO-PÉREZ & CASTEJÓN, 2021), leading to higher speed-ups and increased memory availability.

**Approximate Reanalysis and Reduced Order Modeling** In the classical TO process, structural responses, such as the displacements, are calculated in every iteration by solving a high-fidelity model. This model is solved using; for example, the FE method, which can become computationally expensive for large problems. The idea of approximate reanalysis is to approximately determine these structural responses without solving the high-fidelity model equations. This is done by finding an appropriate reduced basis based on previous

iterations. KIRSCH and PAPALAMBROS (2001) introduced a unified approach for structural reanalysis based on the Combined Approximations (CA) method and considering different types of topological modifications. Later, AMIR et al. (2009) integrated the CA method for approximate reanalysis into TO problems, considering also the corresponding sensitivities. Since then, several approaches from the field of projection-based reduced order modeling were used to build the set of reduced bases. A promising approach is the principal components analysis based on a singular value decomposition (CHOI et al., 2019; XIAO et al., 2020). In general, the high-fidelity model has to be resolved every few iterations to update the reduced basis vectors in order to achieve good approximations during the complete process.

**Multi-Grid Methods** Another promising technique is the use of multi-grid methods. The idea is to define multiple computational grids with different resolutions so that the costly evaluation of the equilibrium equation on a high-resolution mesh is prevented. Y. Y. KIM and YOON (2000) were the first to suggest a multi-resolution multi-scale TO based on wavelet transforms, where the design progresses from a low-resolution level to a high-resolution level. Similarly, STAINKO (2006) suggested an adaptive mesh refinement at the interface between solid and void regions. AMIR et al. (2014) introduced the use of a multi-grid preconditioned conjugate gradients solver, a multilevel iterative method based on relaxation steps, coarse-grid corrections, and transformations from fine to coarse grids and vice versa. NGUYEN et al. (2010) proposed to completely separate the density mesh from the FE analysis mesh. This results in a cheap analysis performed on the coarse FE mesh while a high-resolution design can be maintained on the fine density mesh. The drawback of many multi-grid approaches is the possible loss of fine features that cannot be represented by the coarse resolution.

The methods summarized in these three categories constitute only a subset of all proposed methods. Often, a combination of the aforementioned methods can decrease the computation time even further. One straightforward example would be to incorporate parallel programming in the other reduction methods. For more detailed insights, the reader is referred to the paper of MUKHERJEE et al. (2021).

Recent advances in ML methods have opened new possibilities in many areas, including engineering. Besides solving traditional tasks in computer vision, ML methods, and especially NNs, are being increasingly used in computational mechanics tasks. The introduction of physics-informed NNs (RAISSI et al., 2019) has motivated researchers to explore the possibilities of replacing conventional solving methods with NNs. The field of structural TO has been no exception and the number of relevant papers is increasing rapidly. The paper of WOLDSETH et al. (2022) gives a comprehensive overview of the use of artificial NNs in TO. Here, only a brief summary is given. According to this paper, the main approaches relevant to this thesis can be classified into the following three categories:

**Direct Design** The goal of direct design methods is to remove the expensive iterations usually needed in TO algorithms and to directly obtain the final optimized design. This

can be achieved by predicting the final design using for example CNNs or Generative Adversarial Networks (GANs). YU et al. (2019) first predict a low-resolution design using a CNN with encoder-decoder architecture, where the input consists of the force and geometric boundary conditions. The low-resolution image is then mapped to a higher resolution by a conditional GAN, which is trained with optimized structures generated both at low and high resolutions. In another approach, D. WANG et al. (2022) tried to directly predict the final design with CNNs. They adopted the famous U-Net architecture (RONNEBERGER et al., 2015) and consider as input the displacements and strains of the initial design as well as the prescribed volume constraint. Similarly, SEO and KAPANIA (2023) used squared strain terms from the initial static analysis as input and extended the network architecture to a UNet++ (Z. ZHOU et al., 2018). One of the main drawbacks of direct design methods is the large amount of training data that is usually necessary to obtain results with sufficient quality. For each data sample, a complete TO has to be performed and thousands of those samples are needed. Despite the huge data set, poor generalization properties to unseen problem settings are often encountered.

**Acceleration** A wide range of approaches falls under the category of acceleration. These approaches aim to accelerate the TO process by either reducing the number of iterations or by eliminating time-consuming computations in each iteration. CHI et al. (2021) used a FCNN to directly predict the sensitivities from the current design variables and the current strains computed on a coarse mesh. The number of high-fidelity FE analyses on the fine mesh is thus greatly reduced. In their work, an online training-and-updating concept is employed, which means that the model is trained and updated during the optimization process with data from earlier iterations. This enables a high generalization ability and eliminates the need for an expensive training data generation process. Another way of accelerating the process is by reducing the total number of iterations and thereby achieving a faster convergence. The general idea is to map an intermediate design to the final or near-final design since it is observed that using the SIMP method the main design changes happen only in the first few iterations. Afterward, the overall silhouette of the design remains almost unchanged. The mapping can be done on the full model (SOSNOVIK & OSELEDETS, 2019), on smaller subdomains (JOO et al., 2021), or individual pixels (KALLIORAS et al., 2020), each affecting the generalization capability of the trained model.

**Reduction** The methods that fall into this category aim to reduce the size of the design space, similar to the idea of MOR methods. This is usually done by reparameterizing the design variables by extracting relevant features or by surrogate modeling. GUO et al. (2018) introduced a low-dimensional design representation by using a variational autoencoder to encode the full design space into a lower-dimensional latent space. Design updates are then conducted in the latent space and interpretable structures are obtained again by the decoder. Alternatively, NNs can be used as a direct surrogate for the design representation. CHANDRASEKHAR and SURESH (2021) and DENG and TO (2020) used FCNNs to parameterize the density field, making it independent of the FE mesh. Similarly,

HOYER et al. (2019) proposed a CNN for the reparameterization of the design variables.
In these latter two approaches, the NN is only used as a parameterization tool.

# Chapter 3

# Conventional Topology Optimization

In this chapter, a conventional approach based on the SIMP method is introduced and demonstrated on a typical example, the MBB beam. The MBB beam is a popular, well-known benchmark example often used to validate new methods in TO. Figure 3.1 shows the general problem setup. The beam is supported at the bottom left and right corners

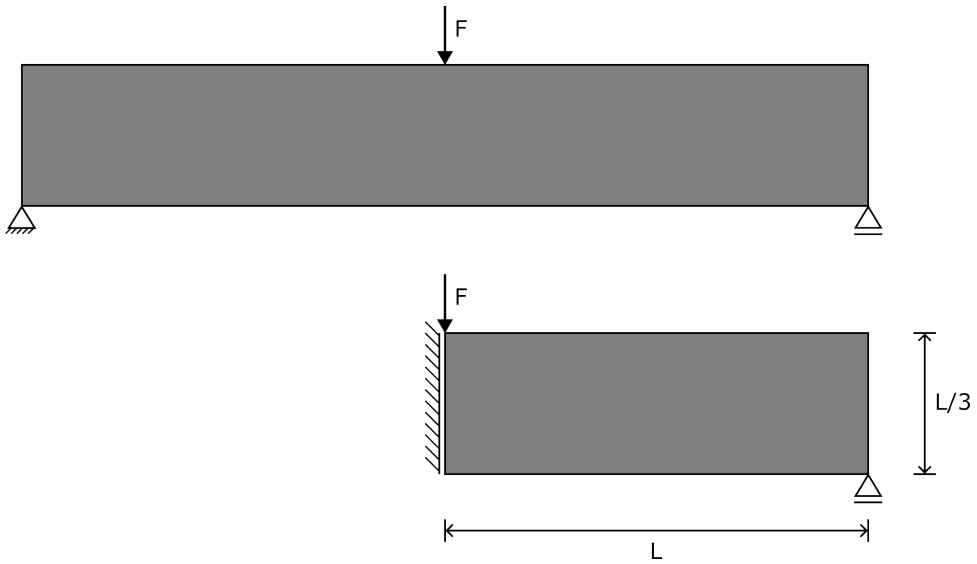



Figure 3.1: The complete MBB beam model on the top and the half MBB beam model used for TO on the bottom.

and a force is applied in the middle of the structure. Due to the symmetry of the problem, only half of the design domain needs to be modeled.

## 3.1 General Problem Formulation

The goal of TO is to find the optimal material layout of a structure with respect to some objective function, within a specified design space and under given support conditions, loads, and constraints. In the simplest case, the objective is to minimize the compliance of the system while a global volume constraint is imposed. The material distribution is described by a virtual density function $\rho(x)$, which can take values between $0$ and $1$. A virtual density value of $\rho(x) = 1$ indicates a material point at location $x$ and a value of $\rho(x) = 0$ represents a void. In the ideal case, the TO should therefore return discrete $0-1$ values. Given the virtual density function $\rho(x)$, the volume constraint takes the form

$$V(\rho) = \int_{\Omega} \rho(x)\, d\Omega \leq V^*, \tag{3.1}$$

where $V^*$ describes the total allowed volume. The variable $\Omega$ represents the design space. In this thesis, only two-dimensional examples with a design space in $\mathbb{R}^2$ are considered.

Minimizing the compliance, or maximizing the global stiffness, is equivalent to minimizing the strain energy of the system. This can be interpreted as finding the optimal choice of the elasticity tensor $C_{ijkl}(x)$ within the design domain $\Omega$ (BENDSOE & SIGMUND, 2003). The equilibrium equation for a displacement $u$ in the weak variational form is

$$a(u,v) = l(v), \quad \forall v \in U, \tag{3.2}$$

where $U$ is the space of kinematically admissible displacement fields and $v$ is an arbitrary, admissible virtual displacement. The energy bilinear form is defined as

$$a(u,v) = \int_\Omega C_{ijkl}(x)\,\epsilon_{ij}(u)\,\epsilon_{kl}(v)\,d\Omega \tag{3.3}$$

with linearized strains $\epsilon_{ij}(u) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$. The load linear form is given by

$$l(u) = \int_\Omega f u\,d\Omega + \int_{\Gamma_T} t u\,ds, \tag{3.4}$$

where $f$ describes the body forces and $t$ the boundary tractions. The general minimum compliance optimization problem thus takes the following form:

$$\left.\begin{aligned} \min: \quad & l(u) \\ \text{s.t.}: \quad & a(u,v) = l(v) \\ & V(\rho) \leq V^* \\ & 0 \leq \rho \leq 1 \end{aligned}\right\}. \tag{3.5}$$

## 3.2 Solid Isotropic Material with Penalization

The equilibrium equation 3.2 introduced in the previous section is commonly solved using numerical methods, such as the FE method. Therefore, the design domain $\Omega$ is discretized into FEs. In the standard density approach, the virtual density distribution is discretized in the same manner as the FE mesh. This means that one density value is assigned to each finite element and the design variables of the optimization problem are the discretized densities collected in the vector $\boldsymbol{\rho}$. This vector can take continuous values in the interval $[0,1]$. In the SIMP, or power-law, approach, the Young's modulus $E_e$ of each element is then defined as

$$E_e(\rho_e) = \rho_e^p E_s, \tag{3.6}$$

where $E_s$ is the Young's modulus of the underlying solid material and $p$ is a penalty factor. Within each element, the material properties are assumed to be constant. In order to avoid singularities during the FE analysis, a small lower bound is imposed on the virtual

densities:

$$0 < \rho_{min} \leq \rho_e \leq 1. \tag{3.7}$$

A common choice is to set $\rho_{min} = 10^{-3}$ for all design variables. As mentioned before, the TO should ideally result in discrete $0-1$ solid-void designs and intermediate gray elements should be suppressed. A penalty factor $p > 1$ is thus introduced in the SIMP method to penalize these gray elements with intermediate values. A common choice for the penalty factor is $p = 3$ as this leads to physically realizable elements for a Poisson ratio of $\nu = 0.3$ (BENDSØE & SIGMUND, 1999). With the definition of the Young's modulus in 3.6, the constitutive matrix of the material under plane stress is defined as

$$\mathbf{C}(\rho_e) = \frac{E(\rho_e)}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} = E(\rho_e)\mathbf{C}_0. \tag{3.8}$$

The corresponding element stiffness matrix is then given by

$$\mathbf{k}_e = \int_{\Omega_e} \mathbf{B}^T\mathbf{C}(\rho_e)\mathbf{B}d\Omega_e = E_e\left(\rho_e\right)\int_{\Omega_e} \mathbf{B}^T\mathbf{C}_0\mathbf{B}d\Omega_e = E_e\left(\rho_e\right)\mathbf{k}_{e,0}, \tag{3.9}$$

where $\mathbf{B}$ is the strain-displacement matrix containing the shape function derivatives and $\mathbf{k}_{e,0}$ is the element stiffness matrix corresponding to a unit Young's modulus. The integral can be solved using numerical integration schemes, such as Gaussian quadrature. If a regular mesh is employed, the strain-displacement matrix $\mathbf{B}$ is the same for all elements and the integral needs to be solved only once. The resulting matrix $\mathbf{k}_{e,0}$ can then be used for all elements in the design domain by simply multiplying it with the actual stiffness value. In the simplest case, quadrilateral elements with bilinear shape functions are employed.

In an alternative modified version of the SIMP interpolation scheme (SIGMUND, 2007), the Young's modulus of an element is defined as

$$E_e\left(\rho_e\right) = E_{min} + \rho_e^p\left(E_s - E_{min}\right). \tag{3.10}$$

Here, a lower bound is set directly for the Young's modulus, which is now independent of the underlying solid material and the penalty factor. No lower bound is needed for the virtual densities anymore which can therefore take values of zero. Additionally, the modified formulation covers two-phase design problems and generalizes better to various filtering schemes. In this thesis, the modified SIMP approach is employed with a lower bound of $E_{min} = 10^{-9}$.

The final discretized optimization problem hence takes the following form:

$$\left.\begin{aligned} \min_{\boldsymbol{\rho}}: \quad & c\left(\boldsymbol{\rho}\right) = \mathbf{F}^T\mathbf{u} = \mathbf{u}^T\mathbf{K}\left(\boldsymbol{\rho}\right)\mathbf{u} = \sum_{e=1}^{N} E\left(\rho_e\right)\mathbf{u}_e^T\mathbf{k}_{e,0}\mathbf{u}_e \\ \text{s.t.}: \quad & \mathbf{K}\left(\boldsymbol{\rho}\right)\mathbf{u} = \mathbf{F} \\ & V\left(\boldsymbol{\rho}\right) = \sum_{e=1}^{N} v_e\rho_e \leq V^* \\ & \mathbf{0} \leq \boldsymbol{\rho} \leq \mathbf{1} \end{aligned}\right\}, \tag{3.11}$$

where $c$ is the compliance of the system. The vector $\mathbf{F}$ is the load vector, $\mathbf{K}$ is the global stiffness matrix, and the nodal displacements are collected in the vector $\mathbf{u}$. The variable $v_e$ describes the volume of the individual elements. The volume constraint is often defined by prescribing a volume fraction $f = \frac{V^*}{V_0}$, where $V_0$ is the volume of a completely filled design space.

## 3.3 Sensitivity Analysis

In order to find the best material layout, the optimization problem has to be solved using an appropriate optimization algorithm. Existing optimization algorithms can in general be divided into gradient-based methods and gradient-free methods. Gradient-free methods, such as genetic algorithms, often suffer from a large number of necessary function evaluations. This can become very expensive since a FE analysis has to be performed for each function evaluation. In the context of TO, gradient-based methods have therefore proven to be advantageous. The design variables are updated in an iterative manner using sensitivity information until a termination criterion is met. Thus the sensitivities of the compliance with respect to the design variables need to be computed.

In typical TO problems, the number of objective functions and constraint equations is usually much smaller than the number of design variables. The adjoint method is therefore an efficient method to compute the sensitivities. Following the discussion of BENDSOE and SIGMUND (2003), the objective function is first extended to

$$c = \mathbf{F}^T\mathbf{u} - \tilde{\mathbf{u}}^T\left(\mathbf{K}\mathbf{u} - \mathbf{F}\right), \tag{3.12}$$

where $\tilde{\mathbf{u}}$ is an arbitrary, but fixed vector. Rearranging the terms leads to the following expression:

$$c = \left(\mathbf{F}^T - \tilde{\mathbf{u}}^T\mathbf{K}\right)\mathbf{u} + \tilde{\mathbf{u}}^T\mathbf{F}. \tag{3.13}$$

The partial derivative of this expression with respect to the density of an element is then

$$\frac{\partial c}{\partial \rho_e} = -\tilde{\mathbf{u}}^T\frac{\partial \mathbf{K}}{\partial \rho_e}\mathbf{u} + \left(\mathbf{F}^T - \tilde{\mathbf{u}}^T\mathbf{K}\right)\frac{\partial \mathbf{u}}{\partial \rho_e}. \tag{3.14}$$

When $\tilde{\mathbf{u}}$ satisfies the adjoint equation

$$\mathbf{F}^T - \tilde{\mathbf{u}}^T \mathbf{K} = \mathbf{0}, \tag{3.15}$$

the sensitivities are found as

$$\frac{\partial c}{\partial \rho_e} = -\tilde{\mathbf{u}}^T \frac{\partial \mathbf{K}}{\partial \rho_e} \mathbf{u} \tag{3.16}$$

and the derivatives of the displacements do not need to be computed explicitly. Furthermore, as the problem under consideration is self-adjoint, $\tilde{\mathbf{u}} = \mathbf{u}$ and no additional computations are needed for the adjoint problem. The sensitivities of the minimum compliance problem are therefore simply

$$\frac{\partial c}{\partial \rho_e} = -\mathbf{u}^T \frac{\partial \mathbf{K}}{\partial \rho_e} \mathbf{u}. \tag{3.17}$$

Adding the formulation from the modified SIMP method, see equation 3.9 and 3.10, to equation 3.17 the final expression becomes

$$\frac{\partial c}{\partial \rho_e} = -p\rho_e^{p-1} \left(E_s - E_{min}\right) \mathbf{u}_e^T \mathbf{k}_{e,0} \mathbf{u}_e. \tag{3.18}$$

## 3.4 Optimality Criteria Method

Two very popular gradient-based methods are the OC method and the method of moving asymptotes (SVANBERG, 1987). The method of moving asymptotes is a flexible, general-purpose algorithm that can handle multiple constraints and a large number of different objectives, but is typically more costly than the OC method. On the other hand, the OC method can only be applied to a limited number of structural optimization settings. For the minimum compliance problem considered in this thesis, the OC method is implemented since the very simple computation of the gradients allows a fast updating scheme.

Following the discussion of N. H. KIM et al. (2021), which is based on the update scheme introduced in BENDSØE (1995), the constrained optimization problem from 3.11 is first converted into an unconstrained problem using Lagrange multipliers:

$$L\left(\boldsymbol{\rho}, \lambda\right) = c\left(\boldsymbol{\rho}\right) + \lambda \left(V\left(\boldsymbol{\rho}\right) - fV_0\right). \tag{3.19}$$

Here, only the volume constraint is added to the Lagrange function, as the state equation is fulfilled in every iteration by the FE analysis and the bounds on the design variables can be enforced directly during the design variable update. The corresponding Karush-Kuhn-Tucker conditions are

$$\begin{cases} \frac{\partial L}{\partial \boldsymbol{\rho}} = \frac{\partial c}{\partial \boldsymbol{\rho}} + \lambda \frac{\partial V}{\partial \boldsymbol{\rho}} = \mathbf{0}, \\ \frac{\partial L}{\partial \lambda} = V - fV_0 = 0. \end{cases} \tag{3.20}$$

The update scheme of each design variable is then based on a scaling factor

$$B_e = -\frac{\frac{\partial c}{\partial \rho_e}}{\lambda \frac{\partial V}{\partial \rho_e}}, \tag{3.21}$$

which takes a value of $B_e = 1$ when the first optimality condition is satisfied. When $B_e < 1$, the corresponding density should be decreased, as increasing $\rho_e$ has a larger effect on the volume than on the compliance. On the other hand, when $B_e > 1$, increasing $\rho_e$ is more efficient in reducing the compliance than in increasing the volume and the density should therefore be increased. This leads to the following update scheme:

$$\rho_e^{new} = \begin{cases} \max(\rho_{min}, \rho_e - m), & \text{if} \quad \rho_e B_e^\eta \leq \max(\rho_{min}, \rho_e - m), \\ \rho_e B_e^\eta, & \text{if} \quad \max(\rho_{min}, \rho_e - m) < \rho_e B_e^\eta < \min(1, \rho_e + m), \\ \min(1, \rho_e + m), & \text{if} \quad \min(1, \rho_e + m) \leq \rho_e B_e^\eta, \end{cases} \tag{3.22}$$

where $\eta$ is a numerical damping coefficient and $m$ is the move limit defining the maximum change in each iteration. The typical choice for these parameters is $\eta = 0.5$ and $m = 0.2$.

In order to satisfy the volume constraint, an appropriate value needs to be determined for the Lagrange multiplier $\lambda$. Following SIGMUND (2001), this can be done with an iterative bi-sectioning algorithm, where, starting from an initial interval $\left[0, 10^5\right]$, the interval is halved until the range is smaller than a chosen tolerance. The value of the trial Lagrange multiplier in each iteration is defined as the midpoint of the current interval. If the volume constraint is violated with the current value, the upper half of the interval is chosen for the next iteration; otherwise, the lower half is chosen.

For the scaling factors $B_e$ the gradients of the compliance and the gradients of the volume, both with respect to the design variables, are needed. The previous section 3.3 already covered the computation of the compliance sensitivities $\frac{\partial c}{\partial \rho_e}$. The sensitivities of the volume $\frac{\partial V}{\partial \rho_e}$ depend in general on the discretization mesh and the shape of the individual elements. In this thesis, only rectangular elements of unit length are considered. Therefore, each element has a unit volume and the gradients are simply

$$\frac{\partial V}{\partial \rho_e} = 1. \tag{3.23}$$

## 3.5 Sensitivity Filtering

Without any additional regularization measures, the design variable update entails two significant numerical problems. The first one is the development of so-called checkerboard patterns and the second one is the nonexistence or mesh-dependency of solutions. Checkerboard patterns can occur in the final optimized design and are regions with alternating solid and void elements, as shown in figure 3.2. These patterns arise when low-order finite elements are used since they are unable to accurately model the stiffness of checkerboards (DIAZ & SIGMUND, 1995). Structures that exhibit these patterns are

Figure 3.2: The optimized MBB beam solved with bilinear quadrilaterals and without any regularization method. Checkerboard patterns occur in the final design.

artificially stiff, thus the stiffness of these structures is highly overestimated. The second issue, which is the nonexistence or mesh-dependency of solutions, refers to the problem that the TO will result in different optimal structures dependent on the resolution of the mesh. This is because more holes in the design lead in general to a smaller compliance value. Ideally, the final structure should converge to the same topology, independent of the mesh, and a finer mesh discretization should only result in a higher resolution.

Different prevention methods addressing these issues were introduced in the literature. These include for example the perimeter control method (HABER et al., 1996), the monotonicity-based minimum length scale method (POULSEN, 2003), and filtering methods. The filtering methods can be divided into density-based (BOURDIN, 2001; BRUNS & TORTORELLI, 2001) and sensitivity-based (SIGMUND, 1994, 1997) methods and have become very popular due to their simple and efficient implementation. The basic idea of both filtering approaches is to compute the weighted average of the quantity of interest in a mesh-independent neighborhood. In this work, sensitivity filtering is employed. Inspired by image blurring filters, a convolutional filter is applied to the sensitivities:

$$\widehat{\frac{\partial c}{\partial \rho_e}} = \frac{1}{\max\left(\rho_e, \gamma\right) \cdot \sum_{f=1}^{N_e} \hat{H}_{e,f}} \sum_{f=1}^{N_e} \hat{H}_{e,f} \rho_f \frac{\partial c}{\partial \rho_f} \tag{3.24}$$

with

$$\hat{H}_{e,f} = \max\left(0, r_{min} - \operatorname{dist}\left(e, f\right)\right), \tag{3.25}$$

where $r_{min}$ is the filter radius and $\operatorname{dist}\left(e, f\right)$ the center-to-center distance between elements $e$ and $f$. Since the densities can become zero in the modified SIMP formulation, a small positive parameter $\gamma = 10^{-3}$ is introduced to avoid divisions by zero (ANDREASSEN et al., 2011). The filter radius $r_{min}$ imposes a minimum feature size and therefore restricts the formation of arbitrarily many holes. This means that a smaller radius will lead to topologies with finer structures, whereas a larger radius leads to simpler designs. The modified sensitivities from equation 3.24 replace the unfiltered, real sensitivities and are used in the update scheme of the OC method.

Figure 3.3 shows the final topology of the MBB beam for different discretizations $N_x \times N_y$ and different filter radii. The filter radius can be defined in terms of the finite element size $h$ or the length $L$ of the design domain. It can be observed that a finer mesh with a filter

(a) Mesh: $90 \times 30$, $r_{min} = 1.5h = \frac{L}{60}$

(b) Mesh: $180 \times 60$, $r_{min} = 1.5h = \frac{L}{120}$

(c) Mesh: $180 \times 60$, $r_{min} = 3h = \frac{L}{60}$

Figure 3.3: Final topology of the MBB beam for different discretizations and different filter radii.

radius spanning the same amount of finite elements leads to a different, more detailed topology, see figure 3.3a and figure 3.3b. If the filter radius is accordingly increased, such that it stays the same with respect to the length $L$, a topology very similar to the original structure, but with a higher resolution, is obtained, compare figure 3.3a and figure 3.3c. Sensitivity filtering hence does not only prevent the formation of checkerboard patterns but also resolves the issue of mesh-dependent solutions.

## 3.6 General Workflow

The general workflow of the minimum compliance problem using the conventional SIMP method is visualized in figure 3.4. In the first step, the optimization problem is defined and the design variables are initialized. For volume-constrained problems, a straightforward initialization is to set all design variables to the prescribed volume fraction $f$. Afterward, an iterative process is started, where a FE analysis is performed with the current design. The displacements obtained from the FE analysis are then used to compute the corresponding compliance and to perform the sensitivity analysis. In the next step, the sensitivities are modified with the filtering method introduced in the previous section and finally, the design variables are updated following the OC update scheme. These steps are repeated until a termination criterion is met and the design has converged to the optimized, final topology. In this thesis, the optimization is stopped when the maximum change between two iterations is smaller than $\epsilon = 0.01$ (SIGMUND, 2001).

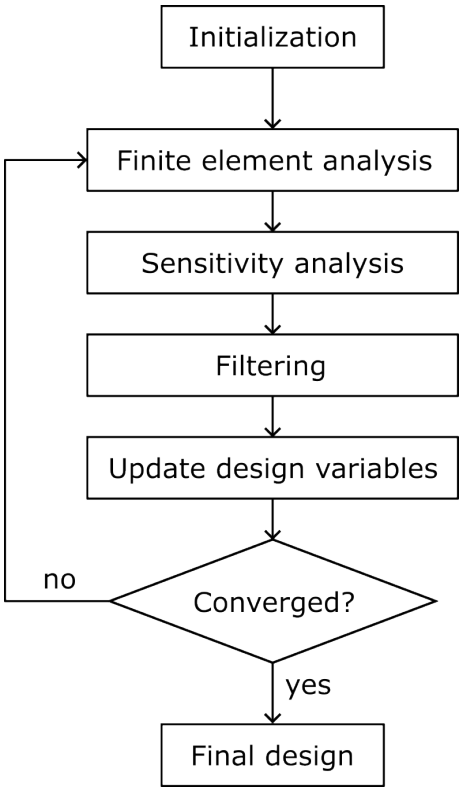



Figure 3.4: The general workflow of a conventional TO.

# Chapter 4

# Higher-Order Multi-Resolution Topology Optimization

In the previous chapter, a conventional and well-studied method for solving minimum compliance problems was introduced. This method has served as basis and reference for several researchers, trying to develop algorithms that further improve the efficiency of TO solution schemes. Especially the computational cost related to the FE analysis can make large-scale TO problems with millions of DOFs unfeasible to solve. In this chapter, a multi-resolution scheme using higher-order shape functions is introduced, which can reduce the cost related to the FE analysis.

## 4.1 Multi-Resolution Scheme

In the conventional SIMP method, the number of design variables determines the resolution of the final design. For many applications, a well-defined solution with a high resolution is desired. Thus, the design domain has to be discretized with a corresponding fine mesh. When the FE mesh is defined in the same manner as the design variable mesh, the large number of DOFs related to this fine mesh lead to a computationally costly FE analysis. To address this problem, NGUYEN et al. (2010) proposed a multi-resolution TO scheme, where different discretization levels are employed. The authors distinguish between a FE mesh used for the analysis, a design variable mesh for performing the optimization, and a density mesh that represents the material distribution. The idea is to have a relatively coarse FE mesh, such that the resulting DOFs of the system are reduced, while maintaining a high-resolution design by employing a fine density mesh. In the paper of NGUYEN et al. (2010) the design and density mesh share the same discretization, but the values do not necessarily coincide. A projection method is used to obtain the densities from the design variables, as this will prevent the formation of checkerboard patterns and also solves the problem of mesh-dependent designs. In this thesis, the density mesh is chosen to be identical to the design variable mesh, so only two distinct meshes are employed. Checkerboard patterns and mesh dependencies are eliminated by applying sensitivity filtering as explained in section 3.5. Figure 4.1 schematically shows the difference between the conventional discretization approach and the multi-resolution scheme. On the left, the number of design variables, and thus the number of density values, is equal to the number of finite elements, and one density value is assigned to each four-noded quadrilateral element. On the right, the multi-resolution scheme is visualized, where the density or design variable mesh is decoupled from the analysis mesh. Here, each four-noded quadrilateral element is described by $6 \times 6$ density variables, thus allowing

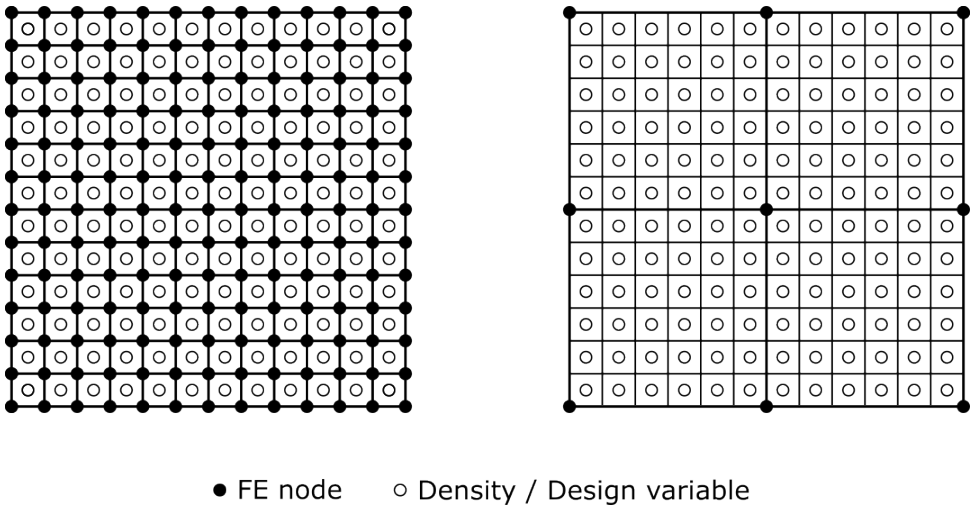



Figure 4.1: Identical discretizations for the FE mesh and the density mesh on the left and the multi-resolution scheme with distinct meshes on the right.

density variations within one finite element. In order to have a clear distinction between the two discretization meshes, an element of the FE mesh will from now on also be referred to as a *cell*, and density elements are henceforth also called *voxels*.

Figure 4.2 shows the result of this multi-resolution scheme applied to the MBB beam. The design domain is discretized with $30 \times 10$ cells, each containing $6 \times 6$ voxels. The topology is thus described by $180 \times 60$ density elements, just as in figure 3.3b. Bilinear shape functions are employed and a sensitivity filter radius of $r_{min} = 1.5h_v$ is chosen, where $h_v$ describes the voxel size. From the figure, it is apparent that artifacts occur in

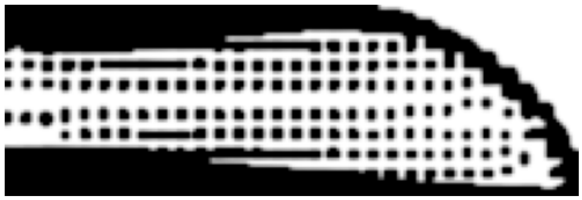


Figure 4.2: The optimized MBB beam solved with bilinear quadrilaterals and the multi-resolution scheme. Sensitivity filtering with a radius of $r_{min} = 1.5h_v$ is applied. QR-patterns occur in the final design.

the final design. These artifacts are referred to as QR-patterns and their properties as well as the reason for their formation were analyzed by GUPTA et al. (2018). QR-patterns are, similar to checkerboard patterns, artificially stiff and occur because of insufficient model capabilities of lower-order finite elements. Filtering techniques, such as sensitivity filtering, can prevent the formation of these QR-patterns when a sufficiently large filter radius is chosen. In traditional TO problems, a filter radius slightly larger than the density element is sufficient to eliminate checkerboard patterns. In the multi-resolution approach, on the other hand, a FE is usually much larger than a density element and a filter radius considerably larger than one voxel is needed to prevent the formation of QR-patterns. Increasing the filter radius can lead to reasonable designs, see figure 4.3, but at the same

time, fine features are lost and less sharp results with more gray elements are obtained. As the formation of QR-patterns is linked to the limitations of lower-order elements, the

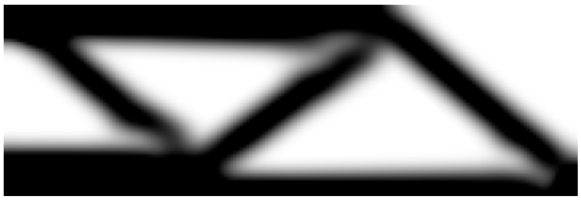


Figure 4.3: The optimized MBB beam solved with bilinear quadrilaterals and the multi-resolution scheme. Sensitivity filtering with a radius of $r_{min} = 7h_v$ is applied.

use of higher-order shape functions can effectively prevent their occurrence. In general, the higher the order, the smaller the necessary filter radius. Higher-order shape functions can thus enable crisp edges and fine features within the multi-resolution scheme.

## 4.2 Higher-Order Shape Functions

The FE method can in general be divided into two versions, the h-version and the p-version. In the h-version of the FE method, lower-order finite elements are usually employed and the accuracy is improved by refining the mesh. In the p-version, on the other hand, the mesh is fixed while the degree $q$ of the shape functions is increased (BABUSKA et al., 1981). NGUYEN et al. (2013) proposed to integrate the p-version of the FE method into the multi-resolution scheme, taking full advantage of both the high analysis quality and the decoupled meshes. In a similar approach, PARVIZIAN et al. (2012) suggested using the finite cell method in TO problems.

In this section, higher-order Legendre shape functions are introduced and in the next section, it is shown how the corresponding stiffness matrix within the multi-resolution scheme is built. The set of one-dimensional, hierarchical Legendre shape functions on the standard element $\Omega_{st}^{1D} = [-1, 1]$ is given by (SZABÓ & BABUŠKA, 2021):

$$\begin{aligned} N_1^{1D}(\xi) &= \frac{1}{2}(1-\xi), \\ N_2^{1D}(\xi) &= \frac{1}{2}(1+\xi), \\ N_i^{1D}(\xi) &= \sqrt{\frac{2i-3}{2}} \int_{-1}^{\xi} P_{i-2}(t)\, dt, \quad i = 3, 4, ... \end{aligned} \tag{4.1}$$

where $P_i(t)$ are the Legendre polynomials. The Legendre polynomials are solutions of the Legendre differential equation and the first eight polynomials are given in appendix A. The corresponding shape functions are called hierarchic because all lower-order shape functions are contained in the higher-order basis. The one-dimensional shape functions for a polynomial degree of $q = 3$ are visualized in figure 4.4.

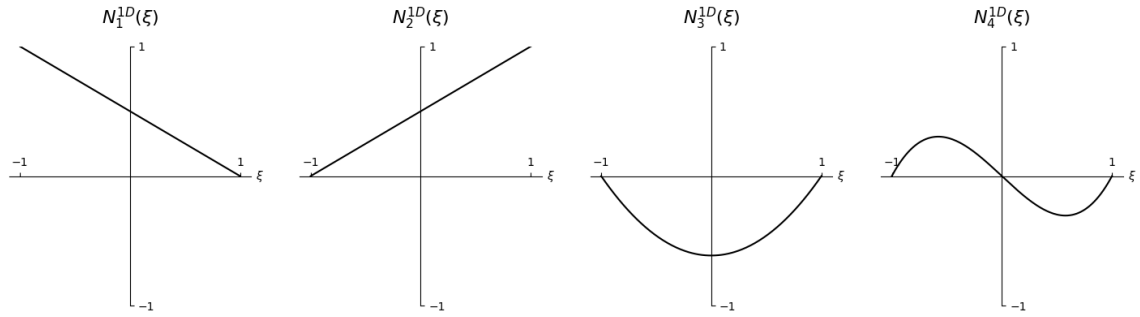
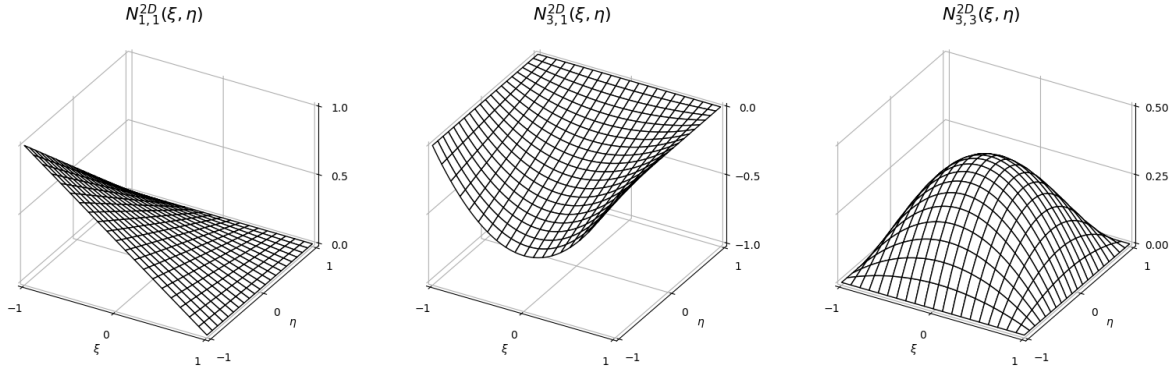



Figure 4.4: Legendre shape functions in one dimension for a polynomial degree of $q = 3$.

In two dimensions, there exist two types of polynomial spaces: the trunk space and the tensor product space. In this thesis, only the tensor product space is considered, which is spanned by the set of all monomials $\xi^k \eta^l$, with $k, l = 0, 1, ..., q$. The polynomial degree in both dimensions is thus assumed to be the same and the two-dimensional shape functions on the standard element $\Omega_{st}^{2D} = ([-1, 1] \times [-1, 1])$ are defined as the tensor product of the one-dimensional shape functions:

$$N_{i,j}^{2D}(\xi, \eta) = N_i^{1D}(\xi) N_j^{1D}(\eta), \quad i, j = 1, 2, ..., q + 1. \tag{4.2}$$

The two-dimensional shape functions can be divided into three different modes, which are also shown in figure 4.5:




(a) Nodal mode (b) Edge mode (c) Internal mode

Figure 4.5: Two-dimensional Legendre shape functions on the standard quadrilateral element.

- Nodal modes: The nodal modes are the bilinear shape functions of the standard four-noded quadrilateral. These four shape functions

$$N_{1,1}^{2D}(\xi, \eta), \quad N_{1,2}^{2D}(\xi, \eta), \quad N_{2,1}^{2D}(\xi, \eta), \quad N_{2,2}^{2D}(\xi, \eta), \tag{4.3}$$

  each take a unit value at one of the four nodes and vanish at all other nodes.

- Edge modes: For the edge modes, a linear shape function is combined with any higher-order shape function:

$$N_{i,j}^{2D}(\xi,\eta) = N_i^{1D}(\xi)\, N_j^{1D}(\eta), \quad \begin{aligned} &i = 1,2, \quad j = 3,4,...,q+1 \\ \text{or} \quad &j = 1,2, \quad i = 3,4,...,q+1. \end{aligned} \tag{4.4}$$

  They are therefore defined separately for each edge, vanishing at all other edges.

- Internal modes: Internal modes are local to one element as they become zero at all four nodes and edges. They are built by combining two higher-order shape functions:

$$N_{i,j}^{2D}(\xi,\eta) = N_i^{1D}(\xi)\, N_j^{1D}(\eta), \quad i,j = 3,4,...,q+1. \tag{4.5}$$

If chosen properly, the use of higher-order shape functions can increase the analysis quality and prevent the formation of QR-patterns. When the filter radius is held fixed, a higher number of voxels per cell also requires shape functions of higher order. In the rest of this thesis, cells with $6 \times 6$ voxels and shape functions of order $q = 4$ are used as recommended by GROEN et al. (2017).

## 4.3 Stiffness Matrix Integration

For the FE analysis, the cell stiffness matrices are needed to assemble the global stiffness matrix of the system. In the multi-resolution scheme, the material of each cell is described by multiple voxels. Thus, the contribution of each voxel needs to be considered in the cell stiffness matrix. Within each voxel, the material is assumed to be constant and the corresponding voxel stiffness matrix is defined in the same manner as in equation 3.9:

$$\mathbf{k}_v = \int_{\Omega_v} \mathbf{B}^T \mathbf{C}(\rho_v) \mathbf{B}\, d\Omega_v = E_v(\rho_v)\, \mathbf{k}_{v,0}. \tag{4.6}$$

This integral can be solved using for example Gauss-Legendre quadrature. For an exact integration, $n_{GP} = q + 1$ Gauss points are required in each direction. This means that for a quadrilateral element with shape functions up to order $q = 4$, a $5 \times 5$ integration scheme is employed for each voxel. This is visualized in figure 4.6. Using the Gauss-Legendre quadrature rule, the integral in equation 4.6 becomes

$$\mathbf{k}_v = \int_{-1}^{1}\int_{-1}^{1} \mathbf{B}^T \mathbf{C}(\rho_v) \mathbf{B} J\, d\xi\, d\eta \approx \sum_{i=1}^{n_{GP}} \sum_{j=1}^{n_{GP}} \left( \left( \mathbf{B}^T \mathbf{C}(\rho_v)\, \mathbf{B} \right) \Big|_{(\xi_i,\eta_j)} w_i w_j J \right), \tag{4.7}$$

where $J$ is the Jacobian, $(\xi_i, \eta_j)$ denote the integration point coordinates and $w_i$ and $w_j$ are the corresponding weights. The stiffness matrix of the cell is then obtained by summing

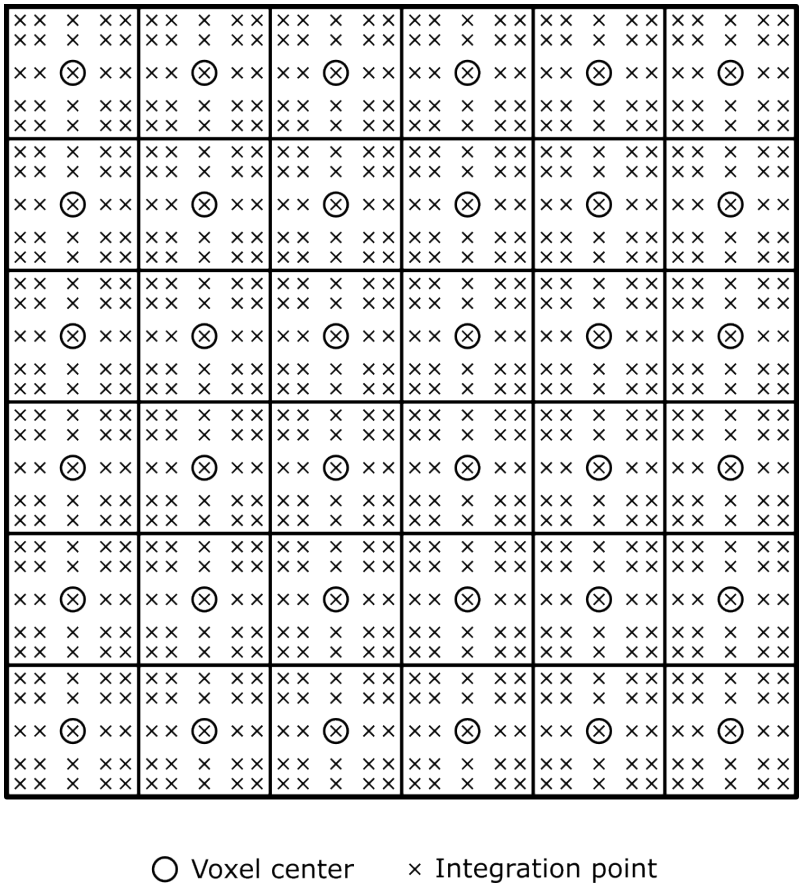



Figure 4.6: A cell containing $6 \times 6$ voxels. The stiffness matrix of each voxel is computed with a $5 \times 5$ integration scheme.

up the contributions of all $N$ voxels contained in the cell:

$$\mathbf{k}_c = \sum_{v=1}^{N} \mathbf{k}_v. \tag{4.8}$$

The expression for the sensitivities is accordingly modified from equation 3.18 to:

$$\frac{\partial c}{\partial \rho_v} = -p\rho_v^{p-1} \left(E_s - E_{min}\right) \mathbf{u}_c^T \mathbf{k}_{v,0} \mathbf{u}_c. \tag{4.9}$$

## 4.4 Static Condensation

The higher accuracy that is achieved with higher-order shape functions also comes with an increasing number of DOFs. For high polynomial degrees $q$, the majority of the DOFs are related to internal modes as their number grows quadratically with increasing $q$. A large number of DOFs will inevitably lead to a computationally expensive FE analysis. When simple static problems are considered, internal modes, which are local to one cell, can be condensed out of the stiffness matrix. This can significantly reduce the number of DOFs and thus the computational cost of the analysis. In this section, the standard Guyan reduction and a variation inspired by parametric MOR are introduced. These reduction

methods are applied on the cell level and the reduced global stiffness matrix is obtained by directly assembling the condensed cell stiffness matrices.

### 4.4.1 Standard Guyan Reduction

For readability reasons, the subscript $c$, used to denote a cell, is dropped in the following discussion. According to GUYAN (1965), the equilibrium equation $\mathbf{ku} = \mathbf{f}$ of each cell can be rearranged such that

$$\begin{bmatrix} \mathbf{k}_{bb} & \mathbf{k}_{bi} \\ \mathbf{k}_{ib} & \mathbf{k}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{u}_b \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}_b \\ \mathbf{f}_i \end{bmatrix}. \tag{4.10}$$

Here, the subscript $b$ describes entries related to the boundary DOFs and internal DOFs are denoted with the subscript $i$. The boundary DOFs are constituted by nodal and edge modes while the internal DOFs correspond to internal modes. Assuming that no external forces are acting on the internal DOFs, the second equation of 4.10 can be rewritten to

$$\mathbf{u}_i = -\mathbf{k}_{ii}^{-1}\mathbf{k}_{ib}\mathbf{u}_b. \tag{4.11}$$

Inserting the formulation from 4.11 into the first equation of 4.10 leads to the following expression:

$$\mathbf{k}_{bb}\mathbf{u}_b - \mathbf{k}_{bi}\mathbf{k}_{ii}^{-1}\mathbf{k}_{ib}\mathbf{u}_b = \left[\mathbf{k}_{bb} - \mathbf{k}_{bi}\mathbf{k}_{ii}^{-1}\mathbf{k}_{ib}\right]\mathbf{u}_b = \mathbf{f}_b. \tag{4.12}$$

The system of equations is thus reduced to the boundary DOFs and the reduced cell stiffness matrix is defined as

$$\mathbf{k}^{red} = \mathbf{k}_{bb} - \mathbf{k}_{bi}\mathbf{k}_{ii}^{-1}\mathbf{k}_{ib}. \tag{4.13}$$

From the reduced solution vector $\mathbf{u}_b$, the full solution vector $\mathbf{u}$ can be reconstructed by

$$\mathbf{u} = \mathbf{T}\mathbf{u}_b, \tag{4.14}$$

where the matrix $\mathbf{T}$ is the coordinate transformation matrix defined as

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{k}_{ii}^{-1}\mathbf{k}_{ib} \end{bmatrix} \tag{4.15}$$

and $\mathbf{I}$ is the identity matrix. The reduced stiffness matrix can therefore also be expressed in terms of the transformation matrix $\mathbf{T}$ as

$$\mathbf{k}^{red} = \mathbf{T}^T\mathbf{k}\mathbf{T}. \tag{4.16}$$

The reconstructed solution vector is in the static case identical to the solution vector obtained from the full system. The solution quality is therefore not impaired, but for

high polynomial degrees $q$, the inversion of $\mathbf{k}_{ii}$ for each cell can be computationally very expensive, making this approach even less efficient than the standard TO.

### 4.4.2 Parametric Model Order Reduction

Although static condensation can significantly reduce the number of DOFs, a matrix inversion is needed for each transformation matrix $\mathbf{T}$, see equation 4.15. This can make the analysis even more costly, as these transformation matrices have to be calculated anew for each new set of density values, i.e. for each iteration and each cell. The goal of parametric MOR, which is often applied to dynamic systems, is to find a reduced model that retains the parameter dependency. Hence, a parametric model is obtained that can be used for a wide range of different parameter settings without the need to repeat the reduction. In general, it is not possible to find a model that is exact for all parameter settings, so parametric MOR methods are usually accompanied by approximation errors.

One approach is to use the reduced matrices obtained from different parameter sets to interpolate the reduced matrix at a new sample set. Following this idea, simple linear scaling is employed in this thesis. The stiffness matrix of a unit cell with $N$ voxels of unit stiffness, i.e. $E_v = 1$, is

$$\mathbf{k}_{c,0} = \sum_{v=1}^{N} \mathbf{k}_{v,0}. \tag{4.17}$$

With the corresponding transformation matrix $\mathbf{T}_0$, computed as in equation 4.15, the condensed cell stiffness matrix is

$$\mathbf{k}_{c,0}^{red} = \mathbf{T}_0^T \mathbf{k}_{c,0} \mathbf{T}_0 = \mathbf{T}_0^T \left( \sum_{v=1}^{N} \mathbf{k}_{v,0} \right) \mathbf{T}_0 = \sum_{v=1}^{N} \mathbf{T}_0^T \mathbf{k}_{v,0} \mathbf{T}_0. \tag{4.18}$$

For each new set of density values, the reduced cell stiffness matrix is then approximated by scaling each unit voxel contribution with the corresponding Young's modulus:

$$\tilde{\mathbf{k}}_c^{red} = \sum_{v=1}^{N} E_v\left(\rho_v\right) \mathbf{T}_0^T \mathbf{k}_{v,0} \mathbf{T}_0. \tag{4.19}$$

Here, the condensed stiffness matrices of the unit voxels $\mathbf{T}_0^T \mathbf{k}_{v,0} \mathbf{T}_0$ are only computed once for all parameter settings, thus eliminating the need for costly matrix inversions in each iteration.

The proposed reduction scheme can be interpreted as a linear inter- or extrapolation of the reduced matrices with unit and zero stiffness. This is a simplified parametric MOR method and for an overview of more sophisticated state-of-the-art methods, the reader is referred to the survey of BENNER et al. (2015). As it will be shown later, the application of this reduction scheme can lead to designs very similar to those obtained with the conventional SIMP method, but small disconnections will occur in the design. In the next chapter, a new representation method for the density field is proposed, which may overcome this problem.

# Chapter 5

# Topology Optimization Using Neural Networks

The use of ML methods in computational mechanics tasks is increasingly gaining attention and popularity. Especially the potential of NNs is extensively explored in the scientific community. In many existing methods, NNs are used to overcome shortcomings inherent in conventional TO methods. In this chapter, two different approaches are proposed. In the first approach, NNs are used to reparameterize the optimization problem and in the second approach, a UNet++ (Z. ZHOU et al., 2018) is trained to accelerate the TO by predicting near-optimal density values.

## 5.1 Density Parameterization Using Neural Networks

Using NNs instead of discretized grids to represent the density field in TO can come with several advantages. These include a possible reduction of the design space, smooth boundaries in the final topology, and a broader range of solutions. In this section, a short introduction to the basics of deep learning is given and it is demonstrated how NNs, more specifically FCNNs and CNNs, can be used to reparameterize the density field in TO problems. For more in-depth explanations of deep learning principles, the reader is referred for example to the textbook of GOODFELLOW et al. (2016).

### 5.1.1 Fully Connected Neural Networks

#### Fundamentals of Fully Connected Neural Networks

FCNNs, or multilayer perceptrons, form the basis of many ML applications. A FCNN usually consists of many layers in a chain:

$$NN\left(\mathbf{x};\boldsymbol{\theta}\right)=f^{(L)}\left(f^{(L-1)}\left(\dots f^{(2)}\left(f^{(1)}\left(\mathbf{x}\right)\right)\dots\right)\right), \tag{5.1}$$

where $\mathbf{x}$ is the input to the NN and $\boldsymbol{\theta}$ are the learnable parameters of the NN. These learnable parameters include for example the weights $\mathbf{W}$ and biases $\mathbf{b}$ of each layer. An example of a simple FCNN with two input features and one output is visualized in figure 5.1. The shown network consists of five layers: the input layer, the output layer, and three hidden layers with six neurons each. It is called fully connected because every neuron is connected to all neurons in the subsequent layer. In the standard case, each layer $l$ consists of a linear mapping of the output $\mathbf{z}^{(l-1)}$ from the previous layer and an activation

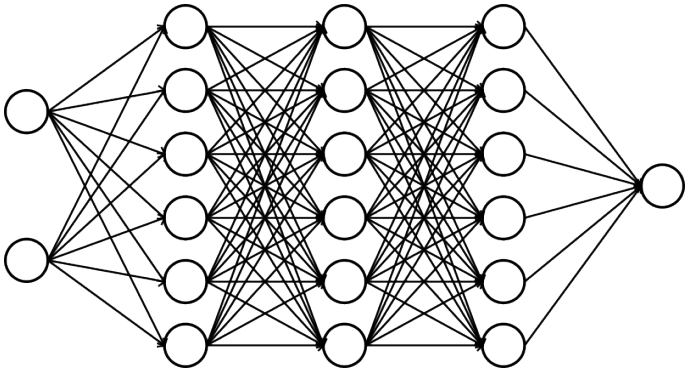


Figure 5.1: A simple FCNN with three hidden layers and six neurons per hidden layer.

function $a^{(l)}$ that introduces nonlinearity to the network. The formulation of layer $l$ is thus defined as

$$f^{(l)}\left(\mathbf{z}^{(l-1)}\right) = a^{(l)}\left(\mathbf{W}^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}\right). \tag{5.2}$$

If no nonlinear activation function is employed in the network, the input to the NN would be mapped linearly to the output, regardless of the number of layers, and the network's learning capabilities would be strongly limited. The right choice of activation functions is therefore an essential part of designing network architectures. One of the most popular activation functions is the Leaky Rectified Linear Unit (ReLU) function, defined as

$$\mathrm{LeakyReLU}\,(x) = \begin{cases} x, & \text{if } x \geq 0, \\ \alpha x & \text{otherwise}, \end{cases} \tag{5.3}$$

where $\alpha$ defines a small negative slope, for example $\alpha = 0.01$. In general, the input to the activation function is a multi-dimensional vector. Consequently, the function is applied element-wise to each vector entry.

The NN is then trained by iteratively updating the learnable parameters $\boldsymbol{\theta}$ of the network such that a loss function $L\,(\boldsymbol{\theta})$ is minimized. For supervised learning, the loss function should quantify the deviation from the current NN output to the desired target output. For the parameter update, an appropriate optimization method needs to be chosen. A popular optimizer is the Adam optimizer (KINGMA & BA, 2014), which implements a stochastic, gradient-based optimization method. Hence, the gradients of the loss function with respect to the network parameters $\frac{\partial L}{\partial \theta_i}$ have to be computed. These can be efficiently computed by automatic differentiation methods, such as backpropagation.

### Overview of the Proposed Method

CHANDRASEKHAR and SURESH (2021) and DENG and TO (2020) proposed to use FCNNs to describe the density field in TO problems. The network thus defines a mapping

$$\rho\,(\mathbf{x}) = NN\,(\mathbf{x}; \boldsymbol{\theta})\,, \tag{5.4}$$

where the density is now parameterized by the learnable parameters $\boldsymbol{\theta}$ of the NN. The input $\mathbf{x}$ consists of the spatial $x$- and $y$-coordinates and the output is the virtual density value $\rho$ at the corresponding position. The NN thus maps a density value to each point in the design domain. The design variables of the optimization problem are now the learnable parameters $\boldsymbol{\theta}$ of the NN, which are updated iteratively during the optimization process.

Following the proposed method of CHANDRASEKHAR and SURESH (2021), the new optimization process is schematically visualized in figure 5.2. Firstly, the density values at the

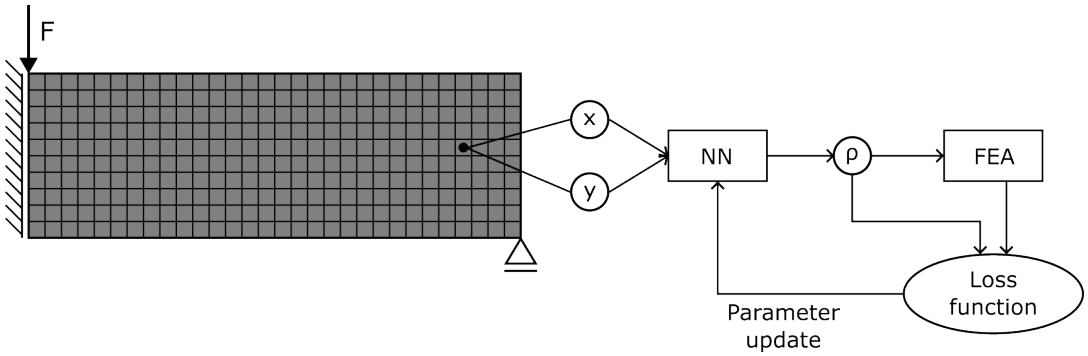



Figure 5.2: Workflow of the proposed method that uses FCNNs to represent the density field.

positions needed for the FE analysis are obtained through the NN. These values are then used to build the stiffness matrix of the system, following the higher-order multi-resolution scheme introduced in the previous chapter. The resulting state equation is solved and the solution is used to compute the current compliance value. The resulting compliance value and the volume constraint are combined to define the loss function $L(\boldsymbol{\rho};\boldsymbol{\theta})$. The sensitivities of the loss function with respect to the design variables $\frac{\partial L}{\partial \theta_i}$ can be obtained with the help of the chain rule and backpropagation. Giving these sensitivities to a suitable optimizer, the design variables are updated iteratively until a termination criterion is met.

For the update of the learnable parameters $\boldsymbol{\theta}$, the Adam optimizer (KINGMA & BA, 2014) is selected with a learning rate of $0.01$. This optimizer is designed for unconstrained problems, so the constrained minimization problem is transformed into an unconstrained one by integrating the volume constraint into the loss function

$$L(\boldsymbol{\rho};\boldsymbol{\theta}) = \frac{c\,(\boldsymbol{\rho})}{c_0} + \alpha \left( \frac{V\,(\boldsymbol{\rho})}{V^*} - 1 \right)^2, \tag{5.5}$$

where $c_0$ is the compliance of the initial system. The volume constraint is integrated by defining a quadratic penalty function, where $\alpha$ is the penalty parameter (NOCEDAL & WRIGHT, 2006). By increasing the penalty parameter $\alpha$ to infinity, the constraint violation is penalized more and more severely, forcing the constraint to be satisfied eventually. According to NOCEDAL and WRIGHT (2006), the penalty parameter $\alpha$ should start at a relatively small initial value $\alpha_0$ and then increase each iteration until a maximum limit is reached.

The gradients of the loss function with respect to each parameter $\theta_i$ can be computed with the chain rule:

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial \rho_v} \frac{\partial \rho_v}{\partial \theta_i}, \tag{5.6}$$

with

$$\frac{\partial L}{\partial \rho_v} = \frac{1}{c_0} \frac{\partial c}{\partial \rho_v} + 2\alpha \left( \frac{V}{V^*} - 1 \right) \frac{\partial V}{\partial \rho_v}. \tag{5.7}$$

The sensitivities of the compliance with respect to the voxel densities $\frac{\partial c}{\partial \rho_v}$ are computed in the same manner as in equation 4.9 and the sensitivities of the volume with respect to the densities $\frac{\partial V}{\partial \rho_v}$ are equal to one as the voxels are defined to have a unit volume. The gradients of the voxel densities with respect to the parameters of the NN $\frac{\partial \rho_v}{\partial \theta_i}$ are obtained through an automatic differentiation technique called backpropagation.

In all previously introduced methods, the optimization process was stopped when the maximum change of the discretized density values fell below a tolerance of $\epsilon = 0.01$. Now that the density is defined as a continuous, global function, CHANDRASEKHAR and SURESH (2021) propose to take the fraction of gray elements as convergence measure. Gray elements are here defined as elements with a density value between $0.05$ and $0.95$. If the fraction of gray elements is less than $\epsilon = 0.05$, the process is terminated.

Comparable to the continuation scheme of the penalty parameter $\alpha$, a continuation scheme is also employed for the penalty factor $p$. Starting from a small initial value $p_0$, the penalty factor is increased in every iteration until the upper limit is reached. This stepwise increase can prevent an early convergence to local minima (SIGMUND & MAUTE, 2013).

DENG and TO (2020) follow a similar approach as CHANDRASEKHAR and SURESH (2021) by using FCNNs to parameterize the density field. In their paper, they focus on the capability of NNs to represent complex geometries and investigate the influence of different activation functions and network architectures. The authors have shown that when the Gaussian function is used as the activation function, highly nonlinear problems can be fitted. The Gaussian function is defined as

$$\text{Gaussian}\,(x) = \exp\left( -\frac{x^2}{2\sigma^2} \right), \tag{5.8}$$

where $\sigma$ is the Gaussian kernel width. The authors have also demonstrated that the kernel width $\sigma$ can control the minimum length scale of features in the final design: A larger value generally leads to simpler topologies and a smaller value to more complex structures with finer features. In contrast to the piecewise linear Leaky ReLU activation function, the Gaussian function is not only nonlinear but also infinitely differentiable at all points. Both the Leaky ReLU and the Gaussian activation function are plotted in figure 5.3.

Another advantage of using FCNNs for the density representation is that the density field is now described by a continuous function independent of the FE mesh. This means that

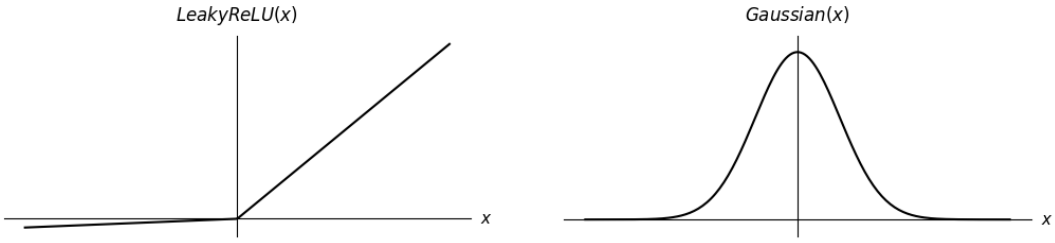



Figure 5.3: The Leaky ReLU function on the left and the Gaussian function on the right.

after the optimization process is finished, the density can be sampled at every location in the design domain, leading theoretically to an infinitely high resolution.

### Network Architecture and Training

The FCNNs employed in this thesis consist of several layers. Starting from two input neurons, which represent the $x$- and $y$-coordinates, the number of neurons in each layer is increased and then decreased to two output neurons. In each layer, except for the final layer, a linear mapping is followed by a batch normalization (IOFFE & SZEGEDY, 2015) and a nonlinear activation function. Batch normalization can accelerate the training process and is a technique, where the output from the previous layer is normalized using the batch mean and batch standard deviation. Usually, NNs are trained in batches so that in each iteration the optimizer is looking at a different subset from the complete data set. In the context of TO, all coordinate points are considered at once in one single batch. In the final layer, a Softmax function is applied directly after the linear mapping to scale the output of the NN to values between $0$ and $1$. The Softmax function is defined as

$$\mathrm{Softmax}\,(x_i) = \frac{\exp\,(x_i)}{\sum_j \exp\,(x_j)} \tag{5.9}$$

and is often implemented in the last layer of classification problems. It takes multiple inputs, where each input is related to a different class. Here, two inputs are considered, where the first one is associated with a solid element and the second one with a void. The output from the Softmax function represents probability values for each class. The second output can thus be disregarded and the first one is interpreted as the density value $\rho$.

Besides batch normalization, other measures can also help the training process. The initialization of the weights in linear layers plays an important role in obtaining stable gradients. Dependent on the selected activation function, a proper initialization method has to be chosen. Common choices are the Xavier normal initialization (GLOROT & BENGIO, 2010) or the Kaiming normal initialization (HE et al., 2015). In the last layer, the weights are initially chosen to be very small and the biases are set to the prescribed volume fraction $f$. In this way, the initial density field is nearly uniform and equal to the prescribed volume fraction. The gradients can be further stabilized by applying gradient norm clipping, where the gradients with a norm exceeding a certain threshold, here set to $0.1$, are scaled down.

### 5.1.2 Convolutional Neural Networks

#### Basics of Convolutional Neural Networks

CNNs are a special kind of NN, where, as the name indicates, convolutions are performed in at least one layer. This kind of NN has been applied very successfully to image-processing tasks. The improved performance is owed to sparse interactions, parameter sharing, and equivariant representations (GOODFELLOW et al., 2016). In general, the input to a CNN is a multi-dimensional tensor. If the input is for example an RGB image, then the input consists of three channels, where each channel is a two-dimensional array of the same size as the image size. The values in the three channels describe the pixel-wise intensity of the three primary colors red, green, and blue. A convolutional operator is then applied to the input, which is schematically visualized in figure 5.4. Here, a $3 \times 3$ kernel

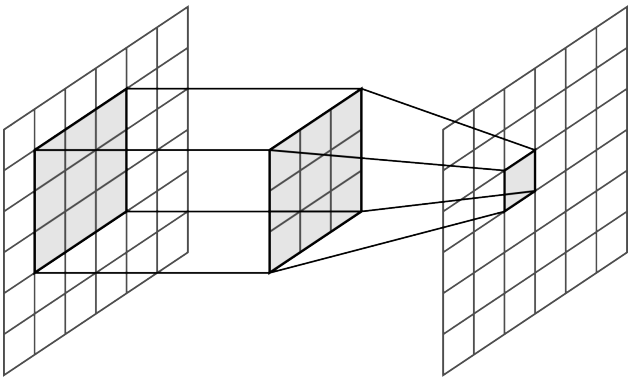


Figure 5.4: Schematic representation of a convolutional operator.

is applied to an input with a single channel. The same filter strides over the complete input, which is usually much larger than the kernel size. For each output value, the entries of the corresponding input window marked in gray are multiplied element-wise with the kernel entries and then summed up. Additionally, a bias can be defined for each filter that is added on top of the sum. If the output dimensions should remain the same as the input dimensions, a padding can be applied to the input, which increases its spatial dimensions. The entries of the kernel represent learnable parameters of the NN, which are updated during the optimization process. Usually, multiple filters are applied to the same input so that different features can be learned. Each filter results in an additional feature map, represented by the channels of the output.

A typical layer in a CNN consists of different operations. After the convolutional operator, a nonlinear activation function is usually applied, just like in fully connected layers. Often, a pooling function modifies the feature maps even further. The most commonly used pooling method is max pooling, where each output is defined as the maximum value within a rectangular neighborhood. Max pooling can therefore be seen as a feature selection process, picking the strongest activation in a neighborhood and thereby downsampling the corresponding input. When the spatial dimensions need to be increased again, for example to perform pixel-wise image segmentation, the corresponding feature maps are upsampled with an appropriate upsampling algorithm.

### Overview of the Proposed Method

In the context of TO, the use of CNNs can be advantageous as they are suited for spatially correlated data. The method described in section 5.1.1 is hence modified to replace the FCNN with a CNN. The rest of the workflow, including the definition of the loss function and the update scheme, stays the same. In a similar approach, HOYER et al. (2019) reparameterized the TO problem with a CNN and used the limited-memory BFGS method for the optimization. For a detailed description of the limited-memory BFGS method, the interested reader is referred to the book of NOCEDAL and WRIGHT (2006).

The input to the CNN is now a two-dimensional array with random values sampled from a uniform distribution in the interval $[0, 1]$. The output has the same size as the input and can be interpreted as the density field given by the current parameters of the NN. The network thus defines a mapping

$$\boldsymbol{\rho} = NN\left(\boldsymbol{\theta}\right), \tag{5.10}$$

where the density field is again represented by discretized density values. The property of an infinitely high resolution, which can be obtained with FCNNs, is therefore lost with CNNs.

### Network Architecture and Training

Similar to the FCNN, the CNN also consists of several layers, where the number of channels is at first increased and then decreased to a single output channel. Since no features are learned from the input, but only the representation capabilities of the CNN are of interest, no pooling is implemented and the spatial dimensions stay the same throughout the entire network. The structure of each layer is comparable to the structure defined in the FCNN, only that the fully connected linear mapping is replaced by a convolutional operator. The convolutional operator is again followed by a batch normalization (IOFFE & SZEGEDY, 2015) and a nonlinear activation function. In the final layer, a Softmax function is applied to scale the output values between $0$ and $1$. The rest of the training procedure, regarding weight initialization and gradient norm clipping, is defined in the same manner as for the FCNN.

## 5.2 Acceleration with Local Stress and Strain Fields

In the previous approach, NNs are only used to parameterize the density distribution, but they do not learn anything. For each TO, the parameters of the NN have to be optimized anew. In this section, another approach is proposed, where a UNet++ (Z. ZHOU et al., 2018) is trained to learn the mapping between the physical properties of an optimization problem and the corresponding near-optimal density distribution.

### 5.2.1 Overview of Proposed Method

The initial analysis of a structure under certain boundary conditions already contains useful information on the final design. Following this idea, different direct design methods and acceleration methods have been proposed that consider these initial conditions as input to a NN. The NN then predicts the final or near-final design, skipping the costly iterations in between. The inputs consist, for example, of the support conditions and loads (YU et al., 2019) or the initial displacements and strains (ZHANG et al., 2019) of the optimization problem. In this thesis, the proposed input by YAN et al. (2022) is combined with the input proposed by SEO and KAPANIA (2023). In both papers, physical quantities obtained from an initial static analysis are used to predict the final design without any iterations. Directly predicting the final design element by element can lead to disconnections and a violation of the volume constraint, assuming that these conditions are not enforced by the NN. Because of this, it is proposed to perform further iterations with the standard SIMP method after the NN prediction. The output from the NN can thus be interpreted as an improved initial guess, closer to the final design than an initially uniform density distribution. The purpose of the NN prediction is therefore to obtain a better starting point for the initial design and thereby to accelerate the convergence of the TO. A similar workflow was also proposed by JOO et al. (2021), where the near-final design prediction from the NN is further optimized in a conventional optimization process. The overall workflow is visualized in figure 5.5.

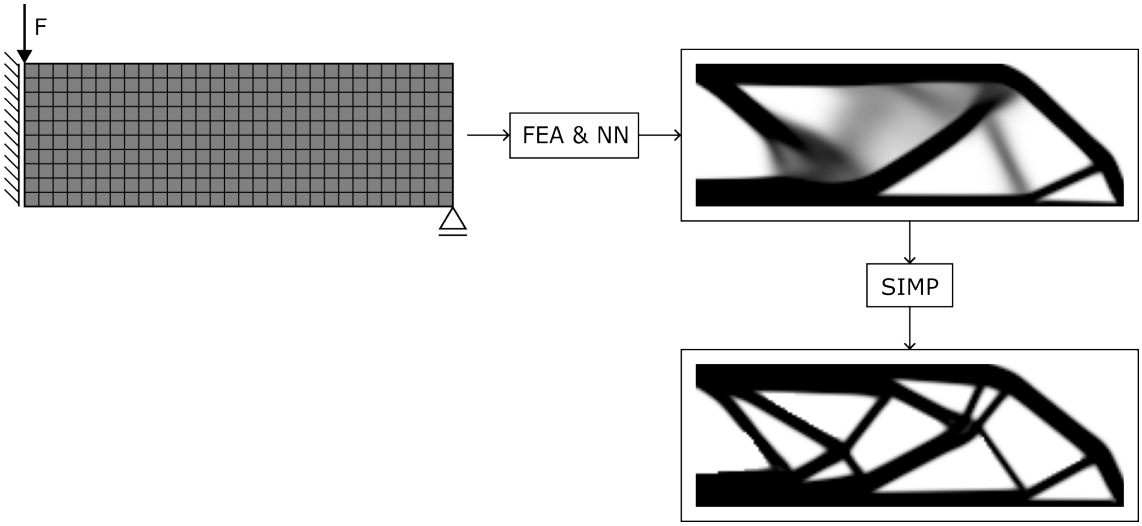



Figure 5.5: General workflow of the proposed method that uses a NN to predict near-optimal density distributions based on an initial static FE analysis. Additional SIMP iterations are performed after the NN prediction.

In minimum compliance TO problems, the internal strain energy of the system represents the objective function that is minimized. In the framework of YAN et al. (2022), the input to the NN thus consists of some strain terms representing this strain energy. Additionally, the initial geometry density is also given as input. In this thesis, a prescribed volume fraction of $f = 0.5$ is considered for all examples. The initial geometry density is thus set to $\rho = 0.5$ for all elements and can be omitted from the input. The internal strain energy of the system

is computed by

$$U = \frac{1}{2} \int_{\Omega} \boldsymbol{\epsilon}^T \mathbf{C} \boldsymbol{\epsilon} \, d\Omega, \tag{5.11}$$

where the strains $\boldsymbol{\epsilon}$ of each element are obtained from a FE analysis performed on the initial structure. Given the nodal displacements $\mathbf{u}$ and the strain-displacement matrix $\mathbf{B}$ of each element, the corresponding strains are found by

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} = \mathbf{B}\mathbf{u}. \tag{5.12}$$

Here and in the rest of this discussion, the subscript $e$, used to denote an element, is dropped for readability reasons. For linear elastic materials, the relationship between the tensor shear strain and the engineering shear strain is given by $\epsilon_{xy} = 0.5\gamma_{xy}$. The internal strain energy density for homogeneous elements can thus be derived as

$$\hat{U} = \frac{E}{2\left(1-\nu^2\right)} \left(\epsilon_x^2 + \epsilon_y^2 + 2\nu\epsilon_x\epsilon_y\right) + \frac{E}{1+\nu}\epsilon_{xy}^2 \tag{5.13}$$

and the input to the NN consists of the four normalized terms $\overline{\epsilon_x^2}$, $\overline{\epsilon_y^2}$, $\overline{\epsilon_x\epsilon_y}$ and $\overline{\epsilon_{xy}^2}$ evaluated at the element center. The strain values at the center of each element are evaluated by substituting $\xi = \eta = 0$ into the corresponding strain-displacement matrix $\mathbf{B}$. Each second-order strain term is then normalized to the interval $[0, 1]$ so that all input features share the same scale. The normalization is calculated with the respective minimum and maximum strain values as

$$\overline{x} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{5.14}$$

In a similar approach, YAN et al. (2022) proposed to learn the optimized topology from the major principal stresses of the initial homogeneous structure. From the calculated strains the stresses are obtained by

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \mathbf{C}\boldsymbol{\epsilon}. \tag{5.15}$$

The major principal stress for each element is then calculated as

$$\sigma_{max} = \left|\frac{\sigma_x + \sigma_y}{2}\right| + \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2}, \tag{5.16}$$

which is again normalized in the same manner as the second-order strain terms.

In order to eliminate singularities caused by point loads or point supports, strain and stress components with a value higher than a respective upper threshold are lowered to the corresponding threshold value. Similarly, values smaller than a lower threshold are raised.

The upper and lower thresholds are defined as the $95$-th percentile, or respectively the $5$-th percentile of the corresponding strain or stress field. These modified fields are then used to compute the second-order strain terms and the major principal stress.

With the strain and stress terms as input, an element-wise classification is employed as suggested by SEO and KAPANIA (2023). The corresponding label is obtained by performing a TO using the conventional SIMP method. In order to obtain discrete $0-1$ labels, a threshold of $\rho_t = 0.5$ is set for the final design. If the final density of an element is smaller than this threshold, it is set to $\rho = 0$ and otherwise to $\rho = 1$. The element-wise output of the NN can then be interpreted as the probability of having a material element and therefore represents the continuous density of that element. The binary cross entropy loss is employed for the classification task at hand and is defined as

$$L = -\frac{1}{N_{spl}} \sum_{s=1}^{N_{spl}} \left( \frac{1}{N_{ele}} \sum_{e=1}^{N_{ele}} \rho_e \cdot \log\left(\hat{\rho}_e\right) + \left(1-\rho_e\right) \cdot \log\left(1-\hat{\rho}_e\right) \right) \Bigg|_s , \tag{5.17}$$

where $N_{spl}$ is the total number of samples given to the NN and $N_{ele}$ is the number of elements per sample. The variable $\hat{\rho}_e$ is the predicted density value of element $e$. The loss value is averaged over all samples and over each element to make it independent of the sample and element size.

Direct design and acceleration methods that aim to predict the final or a near-final design often suffer from poor generalization properties. Large data sets are usually used for the training of the NN, but it is challenging to represent all types of problems. The prediction quality for unseen boundary conditions or mesh resolutions is thus often reduced. The goal of this thesis is to develop an efficient algorithm that is not limited to specific optimization settings. Ideally, the NN should learn local features only dependent on the physical properties within a small observation window. Motivated by this idea, the computed stress and strain fields are divided into subdomains of $32 \times 32$ elements. The input to the NN thus consists of five channels, the four strain terms and the principal stress term, each with a spatial dimension of $32 \times 32$. These $5 \times 32 \times 32$ blocks are independently fed to the NN and the density distribution is predicted separately for each block, as shown in figure 5.6. In a similar approach, JOO et al. (2021) generated their data set by performing optimizations on design spaces with $64 \times 64$ elements. The near-final density distribution of arbitrarily large design spaces is then obtained by decomposing the corresponding design space into a set of these $64 \times 64$ subdomains. In contrast to their approach, the data generated in this thesis is not limited to the design space with only $32 \times 32$ elements, but larger problems, which are divided after the TO, are also considered.

The predictions for the individual subdomains have to be combined to continue the optimization with conventional SIMP iterations. If the original design space cannot be divided exactly by $32$, a higher number of overlapping subdomains is chosen. If the height of the design space is for example discretized with $90$ elements, it can be decomposed into three subdomains, each overlapping the previous subdomain by three elements, see figure 5.7. In general, two different density values will be predicted for each element in the

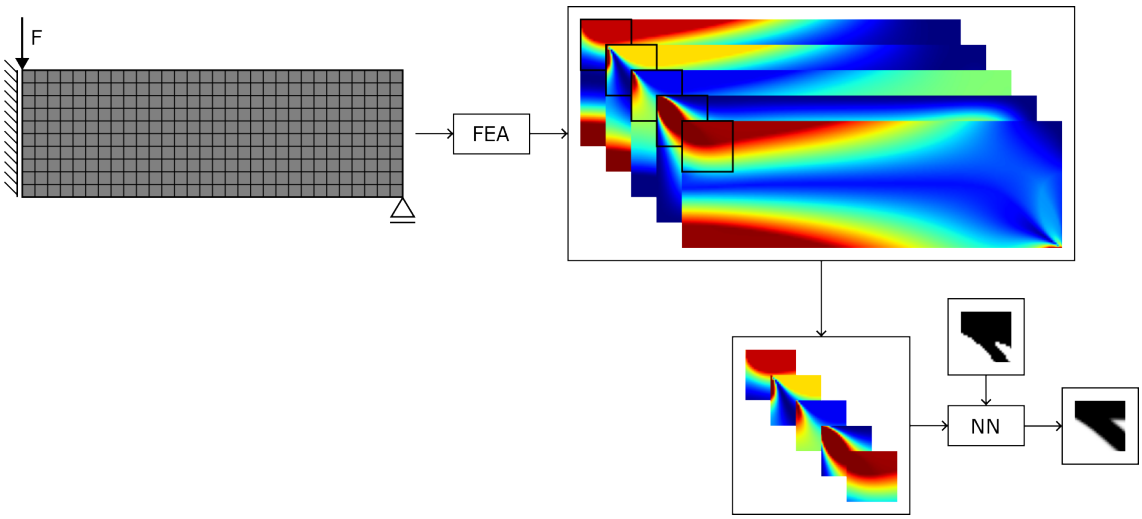



Figure 5.6: Workflow of the proposed method using local stress and strain fields. The input to the NN consists of $5 \times 32 \times 32$ blocks. The density distribution is predicted separately for each block.

Figure 5.7: Overlapping areas occur if the design space can not be decomposed exactly into subdomains of size $32 \times 32$.

overlapping region. Following the framework proposed by Joo et al. (2021), the higher density value is chosen between the two density values at the same position.

In the proposed method, the local stress and strain terms are obtained from the initial static analysis. It can be argued that these physical fields computed on the homogeneous structure do not hold enough information for the mapping to the final topology. This is especially evident in cases where fine features are found in the optimized design. In a modified approach, it is therefore proposed to calculate the stress and strain distributions after performing a small number of SIMP iterations. The intermediate designs can then be given to the NN as additional input. The modified input thus consists of six channels and a better approximation of the final design can be expected.

### 5.2.2 Data Generation

In order to cover a wide range of local stress and strain fields, 20 different mesh resolutions and five different boundary conditions are used for the training data generation. Since subdomains of size $32 \times 32$ are considered by the NN, all design spaces have dimensions dividable by $32$. The $20$ different mesh resolutions $N_x \times N_y$ are listed in table 5.1. The number of elements in $y$-direction is increased from $N_y = 32$ to $N_y = 160$ in steps of $32$. For each height, starting from a square design domain, the number of elements in

Table 5.1: The $20$ different mesh resolutions used for the data generation.

| | | | | |
|---|---|---|---|---|
| $32 \times 32$ | $64 \times 64$ | $96 \times 96$ | $128 \times 128$ | $160 \times 160$ |
| $96 \times 32$ | $128 \times 64$ | $160 \times 96$ | $192 \times 128$ | $224 \times 160$ |
| | $192 \times 64$ | $224 \times 96$ | $256 \times 128$ | $288 \times 160$ |
| | | $288 \times 96$ | $320 \times 128$ | $352 \times 160$ |
| | | | $384 \times 128$ | $416 \times 160$ |
| | | | | $480 \times 160$ |

$x$-direction is increased in steps of $64$ until a length-to-height ratio of $3 : 1$ is reached. For each mesh, five different support conditions are employed. In the first case, the left boundary is fixed as shown in figure 5.8a. The support condition shown in figure 5.8b is the same as for the MBB beam, where a symmetric condition is applied on the left side and the bottom right node is restricted to horizontal movements. In figure 5.8c and figure 5.8d, the same type of support conditions, but applied to the lower edge, are shown. Finally, the fifth support condition consists of two fixed nodes, which are chosen randomly each time. For each mesh resolution and each support condition, three different optimizations

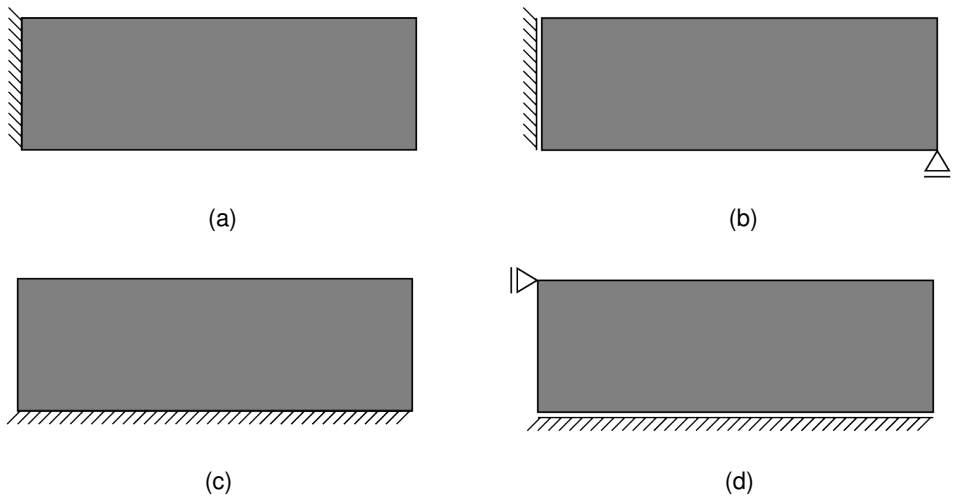
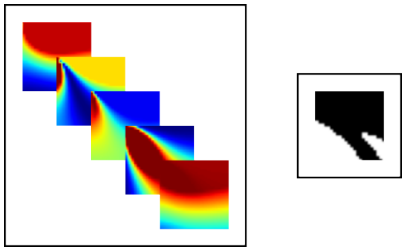



Figure 5.8: Four out of the five support conditions used for the training data generation.

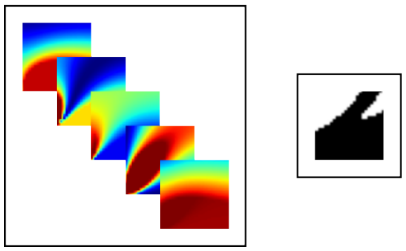
are performed, where one, two, or respectively three nodes are loaded. The nodes are chosen randomly from the free DOFs and the corresponding $x$- and $y$-components are set randomly between $-1$ and $1$. Using $20$ different design domains, each with five different support conditions and three different load cases, leads in total to $300$ optimized topologies. Decomposing these designs into subdomains of dimension $32 \times 32$ results in $8,400$ data samples. This process can be repeated any number of times to generate more training data.

Only using the five support conditions described above restricts the range of problem settings considered by the NN. Data augmentation is thus applied to the generated data set to further increase the sample size without performing additional optimizations. Important to note is that data augmentation should only be applied to the samples in the training set and not to those in the validation set. Each training data sample is first rotated

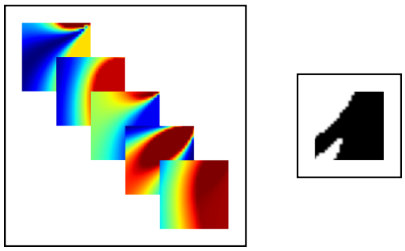
clockwise by $90°$, $180°$, and $270°$ and then flipped across the $x$-axis and again rotated by the aforementioned angles, leading to seven additional examples. The NN will consider these augmented samples as new information. Assuming that the $x$- and $y$-axis stay the same independent of the modification made to the data, the input channels describing the squared strain terms $\overline{\epsilon_x^2}$ and $\overline{\epsilon_y^2}$ have to switch position when a rotation about $90°$ or $270°$ is conducted, compare figure 5.9a and figure 5.9c. In figure 5.9c, the topology is rotated clockwise by $90°$ and the first channel, originally describing the normalized squared strain in $x$-direction, now represents the strain in $y$-direction. In all other cases the input is simply modified according to the label, compare figure 5.9a with figure 5.9b.



(a) Original sample



(b) Flipped across $x$-axis



(c) Rotated clockwise by $90°$

Figure 5.9: Data augmentation performed on the generated training data samples.

### 5.2.3 Network Architecture and Training

Following the paper of SEO and KAPANIA (2023), a UNet++ (Z. ZHOU et al., 2018) is employed for the density prediction. The architecture of the UNet++ is based on the U-Net architecture (RONNEBERGER et al., 2015), a state-of-the-art model for image segmentation tasks. The U-Net is a CNN composed of an encoder-decoder architecture with skip connections. The feature maps from the encoder sub-network are concatenated with the feature maps in the decoder sub-network before the convolutional operators are applied. A UNet++ extends the classical U-Net by adding deep supervision and dense convolution blocks on the skip pathways. Figure 5.10 shows the general UNet++ architecture for three down- and three upsampling layers. The blue components indicate the original U-Net. By adding the three additional nodes, the concatenated feature maps are assumed to be semantically more similar, thus making the optimization easier. Assuming that $x^{i,j}$ denotes

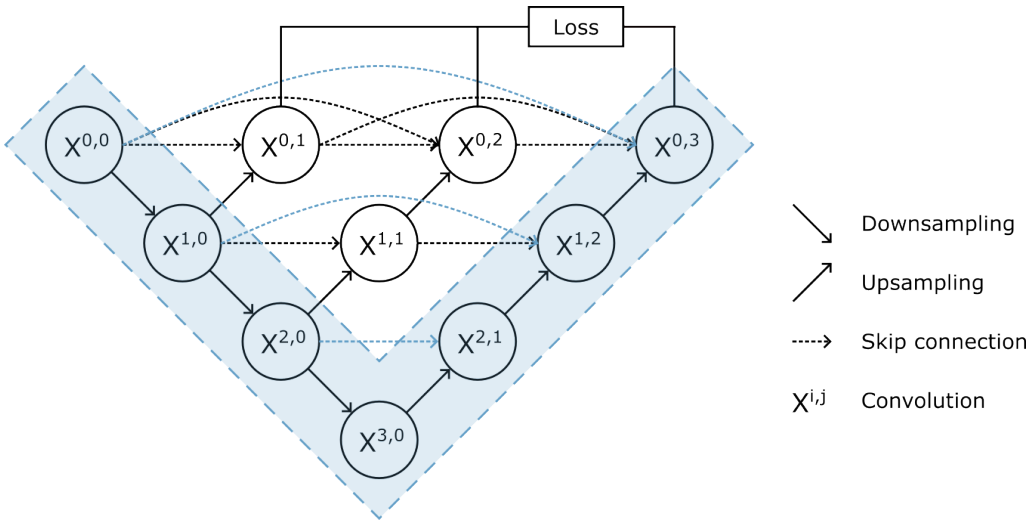



Figure 5.10: The UNet++ architecture for three down- and three upsampling layers.

the output of node $X^{i,j}$, it is computed as

$$x^{i,j} = \begin{cases} \mathcal{H}\left(\mathcal{D}\left(x^{i-1,j}\right)\right), & \text{if} \quad j = 0 \text{ and } i > 0, \\ \mathcal{H}\left(\left[\left[x^{i,k}\right]_{k=0}^{j-1}, \mathcal{U}\left(x^{i+1,j-1}\right)\right]\right), & \text{if} \quad j > 0. \end{cases} \tag{5.18}$$

$\mathcal{H}$ represents a convolutional block consisting of multiple convolutional operators. $\mathcal{D}$ describes a downsampling layer, where a $2 \times 2$ max pooling is applied, and the operator $\mathcal{U}$ is the corresponding upsampling layer using a bilinear upsampling algorithm. The square brackets $[\cdot]$ denote concatenation layers, where different feature maps are stacked together. When deep supervision is activated, the loss from all segmentation branches is averaged. Z. ZHOU et al. (2018) have shown in their experiments that deep supervision improves the performance or, in the worst case, maintains a comparable performance compared to using only one segmentation layer. Since no significant improvement is observed with deep supervision in this context, only the last segmentation map is considered in the loss function.

Each convolutional block $\mathcal{H}$ consists of two convolutional layers, each followed by a batch normalization and a Leaky ReLU activation function. For the NN training, the training data is divided into batches with a batch size of $256$. Due to the choice of the Leaky ReLU activation function, the weights of the NN are initialized with Kaiming normal initialization. In order to reduce overfitting, a dropout layer is added at the end of the convolutional block. In a two-dimensional dropout layer, entire channels of the input are independently zeroed out with a chosen probability $p_d$. Hence, different channels are dropped in each forward pass, which can be interpreted as training a large number of NNs with different architectures in parallel.

In the final classification layer, one filter is applied to the output of the last node to reduce it to one single channel. A Sigmoid function is then used to scale the output to values between $0$ and $1$. The Sigmoid function is commonly used in binary classification problems

and is defined as

$$\text{Sigmoid}\,(x) = \frac{1}{1 + \exp\,(-x)}. \tag{5.19}$$

For the NN training, $80\%$ of the data is assigned to the training set and the remaining $20\%$ to the validation set. The parameter updates are then based on the data of the training set and the validation set is only used to monitor the generalization ability of the NN. The Adam optimizer (KINGMA & BA, 2014) with a learning rate of $10^{-4}$ is implemented for the parameter updates.

# Chapter 6

# Results and Discussion

## 6.1 Problem Definition

In this chapter, all methods will be applied to the MBB beam, which was already introduced in chapter 3. Due to the symmetry of the model, only half of the design domain is modeled as shown in figure 6.1. The length of the design domain is set to $L = 360$ and the

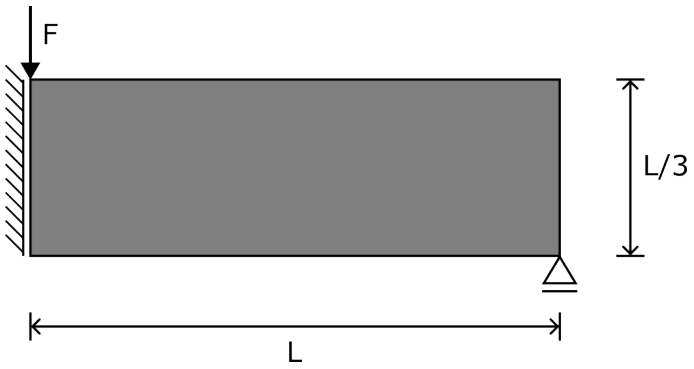



Figure 6.1: The MBB beam model used for TO.

height to $H = \frac{L}{3} = 120$. The design domain is discretized by squares of unit volume, so $N_x \times N_y = 360 \times 120$ density elements are considered for the optimization. This results in a relatively high resolution and a FE model with $87,362$ DOFs, assuming that the FE mesh and the density mesh share the same resolution. Unless stated otherwise, the optimization parameters for the following examples are set as follows:

- Applied force $F = 1$
- Young's modulus $E_s = 1$ and $E_{min} = 10^{-9}$
- Poisson's ratio $\nu = 0.3$
- Penalty factor $p = 3$
- Prescribed volume fraction $f = 0.5$
- Filter radius $r_{min} = 2$

## 6.2 Implementation Details

All numerical examples were implemented in Python using the 99-line Matlab code from SIGMUND (2001) as starting point.

The equilibrium equation is solved with the Python package PyPardiso[1], which provides a Python interface to the Intel oneAPI Math Kernel Library PARDISO solver[2]. This solver is designed for large sparse linear systems of equations and expects as input a square sparse matrix in Compressed Sparse Row (CSR) or Compressed Sparse Column format. In this thesis, the function

```
csr_matrix((data, (row_ind, col_ind)), [shape=(M, N)])
```

from the Python library SciPy[3] is used to construct the sparse input matrices in CSR format. It takes three arrays as input, where the first one holds the non-zero values and the others the corresponding row and column indices. This function is very convenient for assembling element stiffness matrices as duplicate entries are summed together. The computational cost related to the construction of the global stiffness matrix is minimized by taking all operations independent of the current design variables out of the iterative loop. These include, for example, the computation of the element stiffness matrix corresponding to a unit stiffness, see equation 3.9.

In order to avoid expensive for-loops, the sensitivities are computed with the Einstein summation convention[4]. A multidimensional convolution designed for image processing[5] is used to conduct the sensitivity filtering. The code for evaluating higher-order shape functions and their corresponding derivatives is taken from the repository introduced in the paper of KOPP et al. (2022)[6]. All NNs were implemented with PyTorch[7], an open-source ML framework and the implementation of the UNet++ was taken from the GitHub repository pytorch-nested-unet[8].

It is important to note that a computationally efficient implementation of the optimization process is not the main focus of this thesis. More importantly, the effect of different methods and parameterizations is analyzed. The computation time needed for each example is measured with the function

```
time.process_time()
```

from the Time module[9]. This function returns the sum of the system and user CPU time of the current process in seconds. In order to compare the effect of the different methods, the code is executed with a single thread only.

[1] https://github.com/haasad/PyPardisoProject
[2] https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-fortran/top/sparse-solver-routines/onemkl-pardiso-parallel-direct-sparse-solver-iface.html
[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html
[4] https://numpy.org/doc/stable/reference/generated/numpy.einsum.html
[5] https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.convolve.html
[6] https://gitlab.com/hpfem/publications/2021_nd-mlhp
[7] https://pytorch.org/
[8] https://github.com/4uiiurz1/pytorch-nested-unet
[9] https://docs.python.org/3/library/time.html

## 6.3 Comparison of Designs and Computational Cost

In the following section, the results obtained by the different methods will be compared with respect to the solution quality and the computation time. The final output from all optimizations is a density field with continuous density values between $0$ and $1$. A corresponding discrete $0-1$ density field can be obtained by a post-processing step, where the $f \cdot N_x \cdot N_y$ elements with the highest density values are chosen to be solid elements. In this way, the volume constraint is fulfilled exactly.

### Reference Solution

The reference solution is obtained with the conventional SIMP method introduced in chapter 3. As mentioned before, the optimization terminates as soon as the maximum change of the densities between two iterations is smaller than $\epsilon = 0.01$. Figure 6.2 shows the resulting continuous and discrete designs and their compliance values $c$ after $352$ iterations. The corresponding computation times in milliseconds are listed in table 6.1. The

(a) Continuous design, $c = 190.35$

(b) Discrete design, $c = 187.50$

Figure 6.2: The optimized MBB beam using the conventional SIMP method.

values describe the average time needed for each process in one iteration. Each iteration can be divided into the FE analysis, the sensitivity analysis, here denoted as SA, the sensitivity filtering, and the update of the design variables. The FE analysis is again split into two parts, in the first part the global stiffness matrix is built and in the second part the resulting system of equations is solved. In general, the measured computation time varies slightly every time the code is executed. In order to obtain numbers as representative as possible, the average time of three consecutive runs is taken. It is apparent that the FE

Table 6.1: Average time in milliseconds needed for each part of the optimization process for one iteration using the conventional SIMP method.

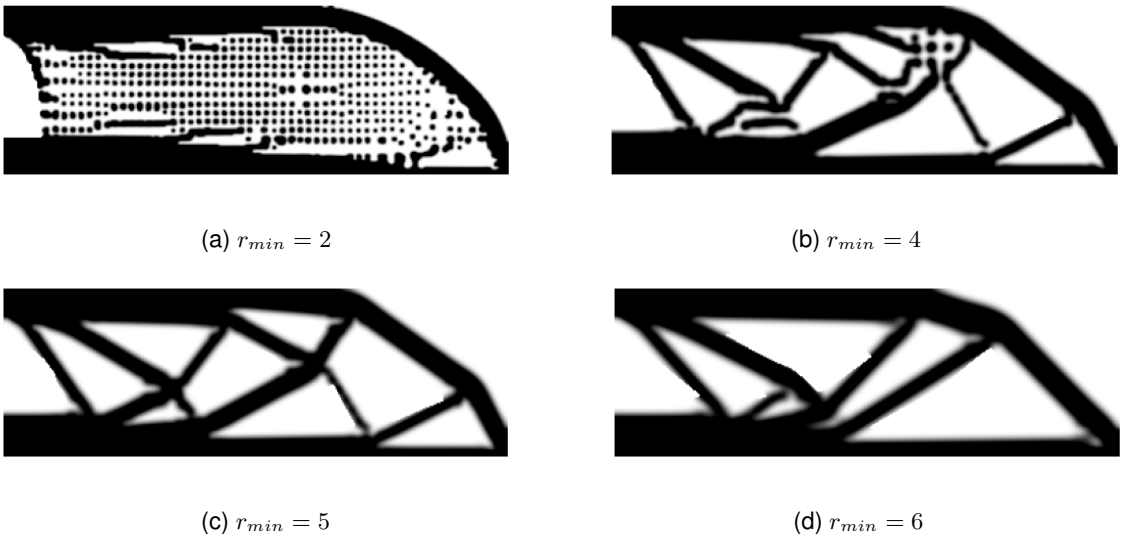
| FE analysis | | SA | Filtering | Update | Total |
|---|---|---|---|---|---|
| Build | Solve | | | | |
| 215.35 | 405.97 | 53.02 | 0.62 | 9.23 | 684.19 |

analysis, here with $87,362$ DOFs, constitutes the majority of the computation time with approximately $90\%$ of the total time. This becomes even worse for larger systems, for example in three dimensions. The sensitivity analysis is considerably less expensive due

to its simple computation, but a relatively large number of matrix multiplications is still needed. Compared to these two processes, the time spent in the sensitivity filtering and the design variable update is almost negligible. The majority of scientific contributions therefore focus on reducing the cost related to the FE analysis.

### Higher-Order Multi-Resolution Scheme

The multi-resolution scheme introduced in section 4.1 aims to reduce the FE analysis cost by employing a coarse FE mesh with less DOFs. As already shown in this section, QR-patterns will occur when the multi-resolution scheme is implemented with bilinear shape functions and too-small filter radii. The optimized topologies for different filter radii are visualized in figure 6.3. The underlying FE mesh consists of $60 \times 20$ elements, each containing $6 \times 6$ density elements. The resulting $360 \times 120$ density field thus shares the same resolution as the reference model, but the FE model with bilinear shape functions has only $2,562$ DOFs. For a filter radius of $r_{min} = 2$ and $r_{min} = 4$, QR-patterns appear



(a) $r_{min} = 2$ (b) $r_{min} = 4$

(c) $r_{min} = 5$ (d) $r_{min} = 6$

Figure 6.3: The optimized MBB beam using the multi-resolution scheme with bilinear shape functions and different radius sizes. QR-patterns occur for small filter radii.

in the final topology. A relatively high filter radius must be chosen to eliminate these QR-patterns. This in turn leads to a loss of fine features as the filter radius imposes a minimum feature size. At the same time, the number of gray elements is increased and the boundaries of the structure become less crisp. Although reasonable results can be achieved with higher filter radii, the fine discretization of the underlying density mesh is not fully exploited. A similar topology as shown in figure 6.3d could be obtained with the conventional SIMP method by choosing a smaller filter radius and correspondingly fewer density elements, see figure 6.4.

As suggested in section 4.2, the use of higher-order shape functions within the multi-resolution scheme can increase the analysis quality and thereby prevent the formation of QR-patterns. The same discretization as before is employed, with $60 \times 20$ finite elements

Figure 6.4: The optimized MBB beam using the conventional SIMP method with $90 \times 30$ density elements and a filter radius of $r_{min} = 1.5$.

and $6 \times 6$ density elements each. In figure 6.5, the continuous and discrete designs, obtained with $r_{min} = 2$ and shape functions of order $q = 4$, are shown. $415$ iterations were needed for the solution to converge. The resulting designs and compliance values

(a) Continuous design, $c = 190.38$

(b) Discrete design, $c = 187.53$

Figure 6.5: The optimized MBB beam using the multi-resolution scheme and higher-order shape functions of order $q = 4$.

are almost identical to the results obtained by the conventional SIMP method. Due to the higher-order shape functions, the number of DOFs increases from $2,562$ to $39,042$. Nevertheless, the number is still more than halved compared to the reference example with $87,362$ DOFs.

Table 6.2 compares the computation time of the reference method, here denoted as cSIMP, with the multi-resolution scheme. When bilinear shape functions of order $q = 1$ are used, the method is abbreviated with MTOP1. The expression MTOP4 respectively describes the use of higher-order shape functions of order $q = 4$. For the multi-resolution TO with bilinear shape functions, the computation time is measured for a filter radius of $r_{min} = 6$.

Table 6.2: Average time in milliseconds needed for each part of the optimization process for one iteration. The conventional SIMP method is compared with the multi-resolution scheme and different shape functions.

| Method | DOFs | FE analysis | | SA | Filtering | Update | Total |
|---|---|---|---|---|---|---|---|
| | | Build | Solve | | | | |
| cSIMP | $87,362$ | $215.35$ | $405.97$ | $53.02$ | $0.62$ | $9.23$ | $684.19$ |
| MTOP1 | $2,562$ | $7.68$ | $8.20$ | $7.03$ | $3.97$ | $8.67$ | $35.55$ |
| MTOP4 | $39,042$ | $246.41$ | $288.52$ | $91.55$ | $0.54$ | $10.81$ | $637.83$ |

The time needed for the update of the design variables is roughly the same for all three methods since the number of design variables does not change. For the multi-resolution

scheme with bilinear shape functions, fine features are lost, but the computational cost is greatly reduced due to the significantly smaller number of DOFs. Only the sensitivity filtering becomes more expensive as the filter size is increased. For higher-order shape functions of order $q = 4$, the numerical integration of the cell stiffness matrices becomes more involved. As expected, the time needed to build the equation system and the time spent on the sensitivity analysis are thus increased. Still, the total time for one iteration is slightly reduced compared to the conventional SIMP method because the reduced number of DOFs leads to a smaller system that can be solved faster. In the rest of this section, shape functions of order $q = 4$ are chosen when the higher-order multi-resolution scheme is employed.

The number of DOFs can be further reduced by eliminating the internal modes from the higher-order shape functions. The two reduction methods introduced in section 4.4, the standard Guyan reduction and the parametric MOR, are thus applied to the optimization of the MBB beam. In this static example, the condensation of the internal DOFs with the standard Guyan method leads to the same result as shown in figure 6.5. The corresponding number of DOFs is reduced from $39,042$ to $17,442$. With the proposed parametric MOR, the reduction is performed only once at the beginning of the optimization process, but the equilibrium equations are not solved exactly. The final continuous and discrete designs, obtained with the higher-order multi-resolution scheme and parametric MOR, are shown in figure 6.6. The corresponding computation times are summarized in table 6.3, where the letter G stands for Guyan reduction and the letter P for parametric MOR.

(a) Continuous design, $c = 199.06$

(b) Discrete design, $c = 212.49$

Figure 6.6: The optimized MBB beam using the higher-order multi-resolution scheme and parametric MOR.

Table 6.3: Average time in milliseconds needed for each part of the optimization process for one iteration. The standard Guyan reduction is compared with parametric MOR.

| Method | DOFs | FE analysis | | SA | Filtering | Update | Total |
|---|---|---|---|---|---|---|---|
| | | Build | Solve | | | | |
| MTOP4 | $39,042$ | $246.41$ | $288.52$ | $91.55$ | $0.54$ | $10.81$ | $637.83$ |
| MTOP4-G | $17,442$ | $301.39$ | $156.50$ | $88.72$ | $0.54$ | $9.24$ | $556.39$ |
| MTOP4-P | $17,442$ | $100.97$ | $145.06$ | $37.96$ | $0.58$ | $10.33$ | $294.90$ |

Evaluating the computation time, it is apparent that the time spent in the FE solver decreases for both condensation methods. This is as expected since the number of DOFs

is reduced in both examples to $17,442$. The main bottleneck with the standard Guyan condensation is now found in building the reduced global stiffness matrix. As explained in section 4.4, a matrix inversion is needed for each cell in each iteration, making the reduction process very expensive. In this example, approximately one-third of the building time is spent on the reduction of the cell stiffness matrices. When parametric MOR is employed, the reduction is performed just once at the beginning of the optimization process, and within the iterations, the reduced voxel matrices are simply scaled with the corresponding Young's moduli. This leads to a significant drop in the building time. Due to the simpler relation between the reduced cell stiffness matrices and the density values of each voxel, the sensitivity analysis also becomes faster.

From figure 6.6, it is visible that, although the overall topology obtained with parametric MOR is very similar to the reference solution, disconnections appear in the final design. The final compliance values are accordingly high. This problem is likely related to the inexact solution of the equilibrium equations and further measures are necessary to obtain viable results.

#### Density Parameterization using Neural Networks

It was proposed in section 5.1 to use NNs for the reparameterization of the density field in addition to the higher-order multi-resolution scheme. It will be shown that this new representation method can effectively eliminate the disconnections that occur in the final design when parametric MOR is employed. For all network architectures, continuation schemes are implemented for the penalty parameter $\alpha$ and the penalty factor $p$. The initial value for the penalty parameter $\alpha$ is set to $\alpha_0 = 0.5$ and the value is increased with a step size of $\Delta\alpha = 0.05$ until a maximum of $\alpha_{max} = 100$ is reached. For the penalty factor $p$, the initial value is set to $p_0 = 2$ and the upper limit to $p_{max} = 4$. The value increases by $\Delta p = 0.01$ in each iteration. The optimization terminates when the fraction of gray elements falls under the value of $\epsilon = 0.05$.

Firstly, a FCNN with Leaky ReLU activation functions, see equation 5.3, is employed for the density parameterization. The weights are correspondingly initialized with a Kaiming normal initialization. The FCNN consists of $13,186$ learnable parameters that represent the design variables of the optimization problem. The detailed network architecture can be found in appendix B.1.1. In the conventional TO approach, the number of design variables is defined by the density mesh. A density mesh with $360 \times 120$ elements thus corresponds to $43,200$ design variables. With the FCNN, the number of design variables is reduced to less than one-third of the original problem. Figure 6.7a shows the resulting optimized topology. Interesting to note is that no sensitivity filtering was necessary to eliminate checkerboard patterns and QR-patterns. Although the filtering can be implemented very efficiently, more crisp edges and designs with fewer gray elements are achieved when the filtering step is omitted. There are also no visible disconnections in the design, even though parametric MOR was applied to the system. However, it is apparent from the figure that a relatively simple structure without any fine features is obtained. A possible

(a) Learnable parameters: 13,186 (b) Learnable parameters: 154,562

Figure 6.7: The optimized MBB beam using FCNNs for the density parameterization and Leaky ReLU as activation function. The result for two different network sizes is shown.

explanation might be that the small network with only $13,682$ learnable parameters is not able to represent more complex topologies. For the design shown in figure 6.7b, the number of learnable parameters was thus increased to $154,562$. The definition of each layer can be found in appendix B.1.2. The overall topology shows no significant difference compared to the topology obtained with the smaller network. The optimization simply terminates in a different optimum when FCNNs are employed. It is important to note that the weights in NNs are, in general, initialized stochastically. The final design is thus not deterministic but varies every time the optimization is performed. Nonetheless, repeated optimizations have shown that the designs visualized in figure 6.7 are representative of their corresponding network architectures.

Since the larger network comes with a significantly increased computational cost but without any improvements in the design, the smaller network architecture is selected for the rest of this analysis. As mentioned in section 5.1.1, the Gaussian function, see equation 5.8, might provide better fitting capabilities than the Leaky ReLU function due to its nonlinear characteristics. The Leaky ReLU function is hence replaced with the Gaussian function and the weights of the FCNN are initialized with the Xavier normal initialization. Example results for different kernel sizes $\sigma$ are shown in figure 6.8. In general, a larger kernel width leads to simpler structures, while reducing $\sigma$ also reduces the minimum feature size. It can be seen that with smaller values of $\sigma$, more complex designs with finer features are obtained. The final topology obtained with a kernel size of $\sigma = 0.5$ has a compliance value of $c = 190.76$ and exhibits fine features comparable to those obtained with the conventional approach. Although reasonable results are possible with this kernel size, it was observed quite often how disconnections and artifacts appear in the final design. This effect is even more pronounced when the kernel width is decreased further. Figure 6.8d shows such an example, where the kernel width is set to $\sigma = 0.3$. The final design has a compliance value of $c = 743.79$ and is not usable anymore.

Until now, no filtering was applied to the sensitivities when FCNNs were used for the density representation, and the TO was terminated when the fraction of gray elements reached the defined tolerance of $\epsilon = 0.05$. Adding a filter could reduce the number of disconnections and artifacts, but the number of gray elements would increase. The termination criterion would therefore have to be relaxed to achieve convergence within a reasonable number of iterations. In an alternative approach, the FCNN is replaced with a CNN. CNNs are suited

(a) Kernel width $\sigma = 2.0$, $c = 195.28$

(b) Kernel width $\sigma = 1.0$, $c = 193.17$

(c) Kernel width $\sigma = 0.5$, $c = 190.76$

(d) Kernel width $\sigma = 0.3$, $c = 743.79$

Figure 6.8: The optimized MBB beam using FCNNs for the density parameterization and the Gaussian function as activation function. The result for four different kernel widths is shown.

for spatially correlated data and might therefore be advantageous in the context of TO. Since the CNN does not treat each voxel individually, but the complete design space is considered, artifacts as in figure 6.8d might be eliminated. For the following topologies, a CNN with $13,818$ learnable parameters is employed. It implements the Gaussian activation function with a kernel width of $\sigma = 0.5$. In each convolutional layer, a convolution filter with a spatial dimension of $7 \times 7$ is applied. The network architecture is described in detail in appendix B.2. It is important to note that the network architecture of the CNN affects the resulting designs. After some parameter tuning, the architecture delivering the best results was selected. Around $30$ optimization runs were conducted with this architecture for the following discussion.

In figure 6.9, two example results are shown. Due to the stochastic initialization of the weights, different optima are found each time an optimization is performed. This can be very advantageous, as the designer can choose among multiple designs and consider, for example, other constraints and preferences. From the resulting compliance values,

(a) Iterations: $197$, $c = 187.70$

(b) Iterations: $192$, $c = 189.97$

Figure 6.9: The optimized MBB beam using CNNs for the density parameterization and the Gaussian function as activation function.

it is apparent that the quality of the results can compete with the quality obtained from the conventional SIMP method, where a compliance value of $c = 190.35$ was achieved

for the continuous design. Out of all conducted optimizations, the best and worst results, determined by their compliance value, are visualized in figure 6.10. The best continuous

(a) Continuous design, iterations: $182$, $c = 186.22$

(b) Discrete design, iterations: $182$, $c = 184.87$

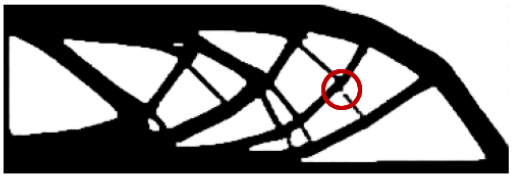
(c) Continuous design, iterations: $190$, $c = 190.93$

(d) Discrete design, iterations: $190$, $c = 189.11$

Figure 6.10: The best and worst designs that were obtained.

design has a compliance value of $c = 186.22$ and even the topology with the worst performance achieved a compliance value very close to the reference solution. The compliance values of all other designs lie in between these limits.

Although, in general, there are no artifacts in the final design, disconnections were observed in a total of two cases. These are shown in figure 6.11. The disconnections



(a) Iterations: $197$ , $c = 187.24$

(b) Iterations: $218$, $c = 189.38$

Figure 6.11: Disconnections occurring in the final topology.

appearing in these topologies are not of the same nature as the disconnections occurring in figure 6.8d. With CNNs, they only appear in isolated cases and locations, and the corresponding compliance values are still comparable to other examples without disconnections.

The goal of the thesis is to reduce the computational cost inherent with large-scale TOs. Thus, the average time needed for one iteration is documented in table 6.4. The elapsed computation time is again obtained by averaging three optimization runs and is given in milliseconds. In the second column, the respective number of design variables is compared. Within the underlying FE analysis, the higher-order multi-resolution scheme with parametric MOR is implemented for both the FCNN and the CNN. Compared to the previous methods, no filtering is implemented, but instead, a forward pass through the NN

Table 6.4: Average time in milliseconds needed for each part of the optimization process for one iteration. NNs are used to parameterize the density field.

| Method | DVs | Forward | FE analysis | | SA | Filtering | Update | Total |
|---|---|---|---|---|---|---|---|---|
| | | | Build | Solve | | | | |
| MTOP4-P | $43,200$ | - | $100.97$ | $145.06$ | $37.96$ | $0.58$ | $10.33$ | $294.90$ |
| FCNN | $13,186$ | $109.36$ | $101.02$ | $143.49$ | $223.97$ | - | $1.29$ | $579.13$ |
| CNN | $13,818$ | $34.88$ | $102.62$ | $140.79$ | $113.77$ | - | $0.84$ | $392.90$ |

is necessary. In order to obtain the density values used in the FE analysis, the input to the NN is fed through the layers to the output. This step constitutes a significant part of each iteration, especially for the FCNN. The time spent on the sensitivity analysis also increases, since not only the sensitivities of the compliance with respect to the density values have to be computed, but also the sensitivities of the density values with respect to the learnable parameters of the network. The time spent in the FE analysis stays approximately the same since the same algorithm is implemented for all three approaches. Finally, the update of the design variables is accelerated, since the OC method is not needed for the unconstrained optimization problem at hand.

Taking the sum of all sub-processes in the optimization, an average iteration with the CNN takes $392.90\,\mathrm{ms}$. Although this is longer than the time needed for an iteration in the parametric higher-order multi-resolution approach with $294.90\,\mathrm{ms}$, artifacts and defects as shown in figure 6.6 are effectively eliminated. Compared to the computation time of the fastest deterministic method without any defects, namely the higher-order multi-resolution approach with standard Guyan reduction and $556.39\,\mathrm{ms}$ per iteration, the use of CNNs is much more efficient. Table 6.5 gives a final comparison between the conventional SIMP method and the method using CNNs. The average time per iteration is reduced from $684.19\,\mathrm{ms}$ to $392.90\,\mathrm{ms}$. It is also important to note that the new parameterization

Table 6.5: A final comparison of the conventional SIMP method and the new parameterization method with CNNs. The average time in milliseconds needed for each part of the optimization process for one iteration is given.

| Method | Forward | FE analysis | | SA | Filtering | Update | Total |
|---|---|---|---|---|---|---|---|
| | | Build | Solve | | | | |
| cSIMP | - | $215.35$ | $405.97$ | $53.02$ | $0.62$ | $9.23$ | $684.19$ |
| CNN | $34.88$ | $102.62$ | $140.79$ | $113.77$ | - | $0.84$ | $392.90$ |

method with CNNs in combination with the new termination criterion leads to a faster convergence in fewer iterations. For all examples that were conducted, the maximum number of iterations is $269$, while most examples converged in less than $200$ iterations. This leads to a total computation time of approximately $100\,\mathrm{s}$ for the complete optimization of the MBB beam. Compared to that, the total computation time using the conventional SIMP method is approximately $250\,\mathrm{s}$ for $352$ iterations. The total computation time is thus reduced by a factor of roughly $2.5$.

### Acceleration with Local Stress and Strain Fields

The NNs used for the density parameterization do not save any information on the optimized topologies and the parameters of the NN are optimized anew for each TO. In another approach, a NN is trained with TO data to predict near-optimal designs in new problem settings. The goal is to accelerate the TO by achieving a faster convergence and thereby reducing the number of iterations after the prediction. The approach was introduced in section 5.2 and is based on the conventional SIMP method only. Hence, it serves as a proof-of-concept and none of the conventional acceleration methods introduced so far are integrated.

The first model is trained with local stress and strain fields computed on the initial homogeneous design. The corresponding data set was generated as described in section 5.2.2. The process described in this section was repeated five times, so $1500$ designs were generated in total. Dividing these designs into subdomains with $32 \times 32$ elements results in $42,000$ data samples. The samples are randomly split into a training set with $33,600$ entries and a validation set with the remaining $8,400$ samples. Data augmentation performed on the training samples leads to a final training data set of size $268,800$. Since from almost each TO, multiple training samples are generated, the necessary number of optimization runs is significantly smaller than the number used in comparable approaches. SEO and KAPANIA (2023), for example, conducted $12,500$ optimizations for their training and JOO et al. (2021) consider $5,000$ optimized topologies. The proposed Unet++ is trained for $300$ epochs, where in each epoch, the complete training set is passed through the network in batches of size $256$. The network has over $5.98$ million parameters and a dropout value of $p_d = 0.4$ is introduced to prevent overfitting. The detailed network architecture is given in appendix B.3. The training and validation loss for each epoch and the corresponding validation accuracy are plotted in figure 6.12. The accuracy is defined as the fraction of correctly classified elements when a threshold of $\hat{\rho}_t = 0.5$ is set for the predicted densities. From the loss history, it is visible that the training loss is still decreasing after $300$ epochs.

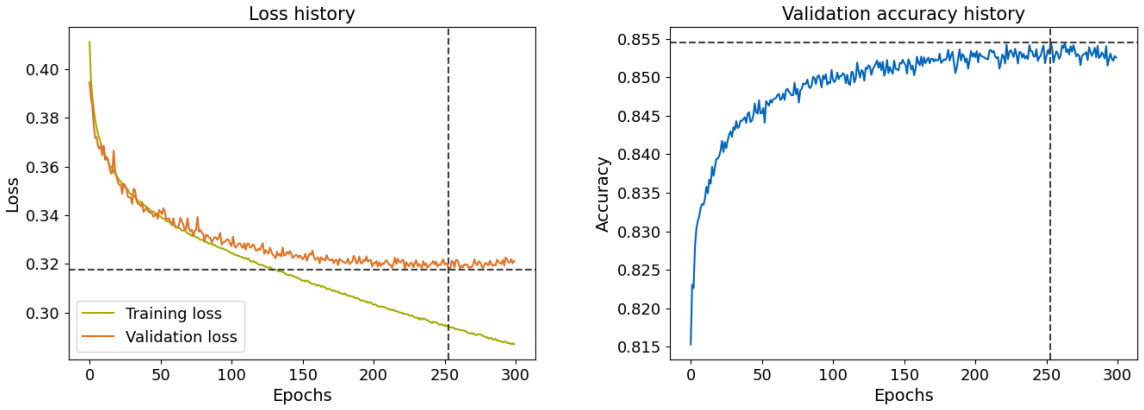



Figure 6.12: The training and validation loss history and the corresponding validation accuracy for each epoch. The input consists of the initial stress and strain terms.

This indicates that the proposed UNet++ architecture could have enough capacity to model

the mapping between the input and the corresponding label of the given training set. The validation loss, on the other hand, reaches its minimum value after epoch $254$ with a loss value of $L_{min} = 0.318$, and the highest accuracy is achieved in the same epoch with $A_{max} = 0.855$. After that, the network overfits the training samples and the prediction quality on the unseen validation set decreases. The model parameters in epoch $254$ are thus saved for the density prediction in new examples. This trained model is now used to predict the density distribution of the MBB beam. The corresponding input computed on the initial structure is visualized in figure C.1 in appendix C. The density prediction of the MBB beam is shown in figure 6.13. The discrete final design, representing the label, is given on the right for comparison. Since the $N_x \times N_y = 360 \times 120$ design space

(a) Predicted density values

(b) Reference solution

Figure 6.13: Prediction of the density field based on the initial stress and strain terms. The individual subdomains are joined after the NN prediction.

cannot be divided exactly into subdomains of size $32 \times 32$, the next larger number is chosen. The design space is thus divided into $12$ subdomains in $x$-direction and four in $y$-direction. Each subdomain, except for the last two of each row and column, overlaps the previous subdomain by two elements. For the last two subdomains, the overlapping region is increased to three, such that the complete design domain is fitted exactly. The average prediction accuracy is $A = 0.778$ and the binary cross entropy loss is $L = 0.411$. It is apparent that the NN can predict the rough structure of the optimized design but in regions where finer features occur, it is not able to learn the complex relationship between the input and the label. In those regions, the NN simply predicts density values close to the average of $0.5$. Considering the input fields visualized in figure C.1, this result is not surprising. The information on these fine features is just not present in the stress and strain terms computed on the initial homogeneous beam. Using the predicted density distribution from figure 6.13a as starting point for a conventional SIMP based TO, the final design is obtained after $349$ iterations. The continuous design achieves a compliance value of $c = 187.91$ and the discrete design a value of $c = 184.25$. The resulting topologies are visualized in figure 6.14. Due to the different density initialization, the optimization converges to a different minimum. In this case, the final topology obtained with the NN prediction achieves better compliance values than the reference solution. However, the number of iterations is not significantly reduced. The total computation time is thus approximately the same, if not slightly increased, considering that the prediction by the UNet++ takes additional time. For the example at hand, the total time needed for the initial FE analysis, the generation of the input to the NN, the loading of the pre-trained model, the

(a) Continuous design, $c = 187.91$ (b) Discrete design, $c = 184.25$

Figure 6.14: The optimized MBB beam using the density prediction from the UNet++ as starting point. The input to the UNet++ is obtained from the initial structure.

forward pass, and the combination of the predicted subdomains is about $4.56\,\mathrm{s}$. This value was obtained by taking the average time of three consecutive runs.

As discussed in section 5.2.1, the stress and strain fields obtained from the initial structure might not hold enough information to predict density fields with fine features. Alternatively, the stress and strain fields can be computed after some iterations when the basic structure has already evolved. In this thesis, the FE analysis for the input data is performed after $10$ iterations. In addition to the five input channels, consisting of the quadratic strain terms and the principal stress term, the intermediate density field after $10$ iterations is also given to the NN. The input thus consists of six channels. The remaining architecture of the UNet++ stays the same as before, only the dropout parameter is increased to $p_d = 0.5$. Figure 6.15 shows the loss and validation accuracy history of the model for $200$ epochs. With the new

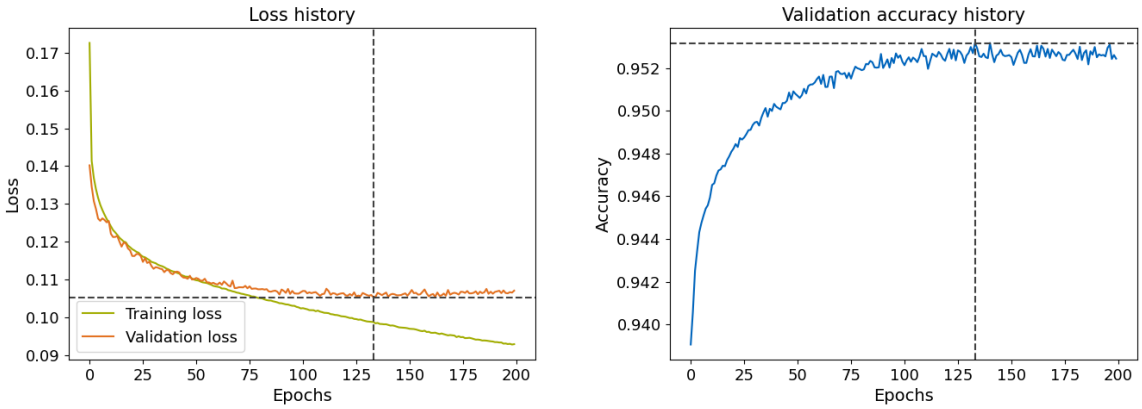



Figure 6.15: The training and validation loss history and the corresponding validation accuracy for each epoch. The input consists of the stress and strain terms computed after $10$ SIMP iterations.

input data, the NN can predict density values with higher accuracy. The smallest validation loss is obtained after $134$ iterations with a value of $L_{min} = 0.105$. The corresponding classification accuracy is increased to $A_{max} = 0.953$. The model parameters after this epoch are saved for the density prediction.

For the optimization of the MBB beam, the conventional SIMP method is used to perform $10$ iterations. The resulting density distribution is shown in figure 6.16. The corresponding stress and strain fields are computed and visualized in figure C.2 in appendix C. Together

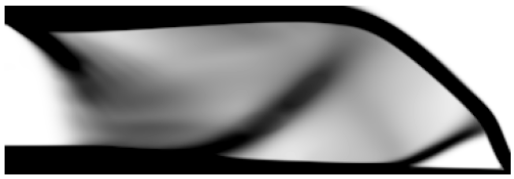


Figure 6.16: The MBB beam using the conventional SIMP method after $10$ iterations.

with the density distribution shown in figure 6.16, these fields are given to the trained model. Figure 6.17 shows the output of the NN and the reference solution for comparison. The average prediction accuracy for this example is $A = 0.907$ and the binary cross

(a) Predicted density values

(b) Reference solution

Figure 6.17: Prediction of the density field based on the stress and strain terms computed after $10$ SIMP iterations. The individual subdomains are joined after the NN prediction.

entropy loss is $L = 0.205$. From the result, it is apparent that the input computed after a few iterations contains much more information on the optimized design. The NN can predict more detailed features and fewer gray regions are present. The predicted density field is already quite close to the final optimized topology. Continuing the optimization after the prediction leads to a converged design in altogether $423$ iterations. Figure 6.18 shows the final continuous and discrete designs, as well as the corresponding compliance values. Again, a different minimum is reached and slightly better compliance values are

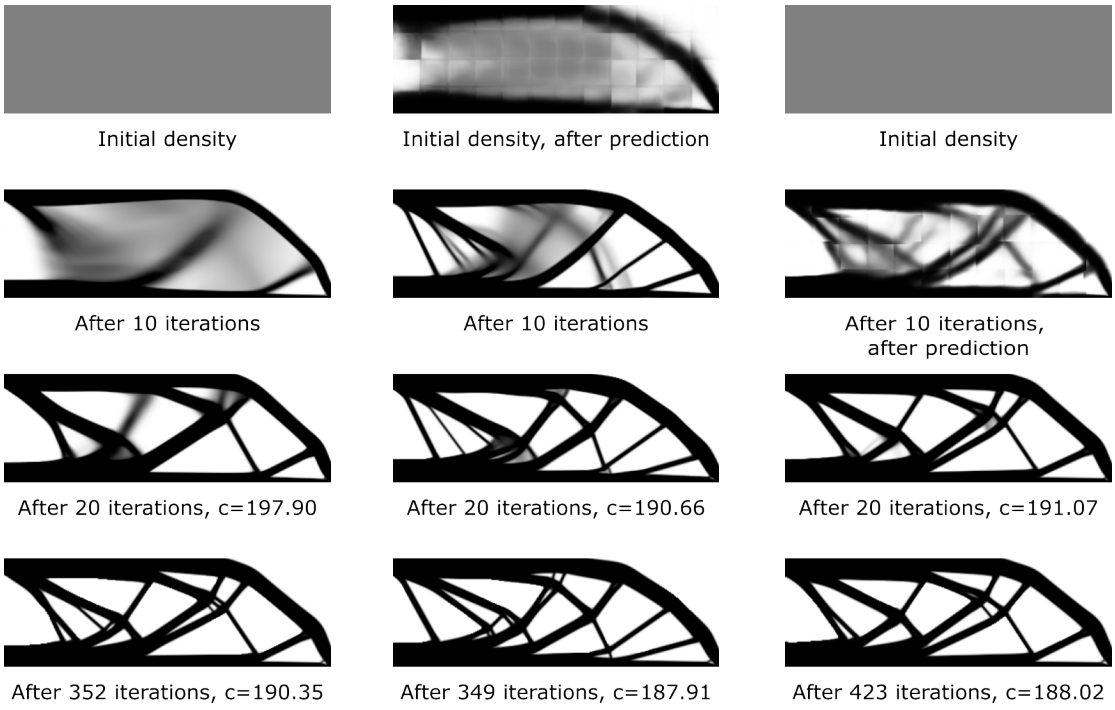
(a) Continuous design, $c = 188.02$

(b) Discrete design, $c = 184.51$

Figure 6.18: The optimized MBB beam with a density prediction from the UNet++ after $10$ iterations.

achieved. Although the prediction quality of the NN is significantly improved compared to the previous model, the number of iterations until the termination criterion is satisfied increases. If the design development in each iteration is analyzed, it becomes clear that significant changes in the topology and compliance value are found only in the first few dozen iterations. Therefore, it may be interesting to evaluate the design development in the early stages of the TO in more detail. Figure 6.19 shows the development of the topology

for the different approaches. For all methods, the structure after $20$ iterations is already




(a) Conventional SIMP method

(b) Density prediction at the beginning

(c) Density prediction after $10$ iterations

Figure 6.19: The design at different stages of the optimization without density prediction, with a density prediction at the beginning, and with a density prediction after $10$ iterations.

close to the final optimized design, even more for the two cases where the UNet++ was used to predict the density. Interesting to note is that after $20$ iterations, both the designs shown in figure 6.19b and figure 6.19c already achieve a compliance value close to the final compliance value of the reference design. The evolution of the compliance value for all three optimizations is plotted in figure 6.20. On the left figure, the compliance history is

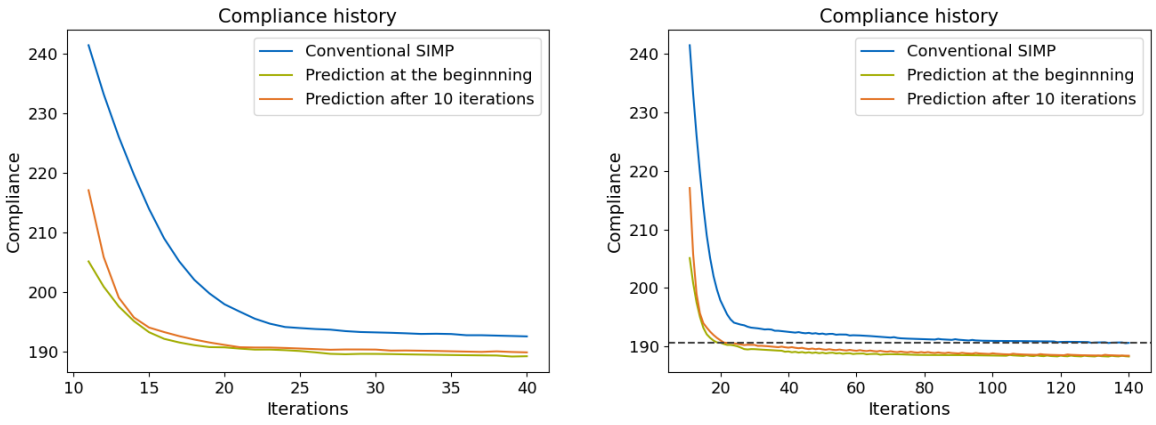



Figure 6.20: The compliance history for all three optimization methods.

plotted from the eleventh iteration to the $40$-th iteration. The values decrease rapidly in the first iterations and then begin to stagnate. The relative change in the compliance value for

an iteration $i$ is defined as

$$r_i = \frac{|c_{i-1} - c_i|}{c_{i-1}}. \tag{6.1}$$

For the conventional SIMP method, three consecutive iterations with a relative change smaller than $r = 0.003$ are achieved after $25$ iterations. For both the NN based optimizations, this condition is fulfilled after $19$ iterations. A faster convergence is thus achieved by the NN prediction. In the right plot, the compliance history is extended to the $140$-th iteration. As determined before, using a Unet++ to predict density fields leads to compliance values close to the final reference compliance already after a few iterations. The dashed horizontal line is placed at $c = 190.66$, which corresponds to the compliance value after $20$ iterations when the density is predicted at the beginning of the TO. The conventional SIMP method needs approximately $130$ iterations to achieve a design with similar compliance. The use of NNs thus leads to improved designs in the early stage of the optimization.

Another interesting aspect is the development of the number of gray elements. In the previous section, where NNs were used to reparameterize the design field, the fraction of gray elements was used as the termination criterion. In the same manner as before, an element is categorized as gray, if the density is between $0.05$ and $0.95$. The evolution of the number of gray elements until the $40$-th iteration is plotted in figure 6.21. Comparable

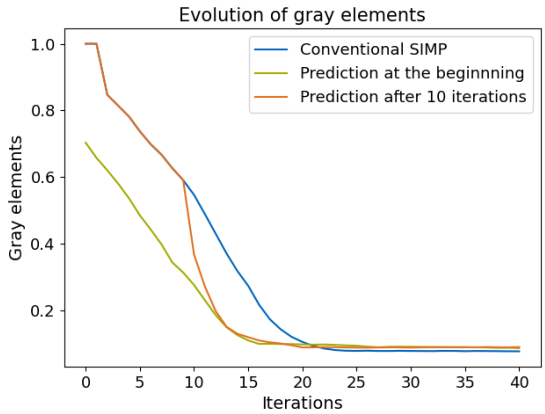



Figure 6.21: The evolution of gray elements for all three optimization methods.

to the compliance value, the number of gray elements decreases rapidly in the first few iterations and reaches a plateau afterward. When a NN is used, the point where this plateau starts to emerge is shifted forwards by a few iterations, similar to the behavior of the compliance history. From this plot and figure 6.19, it is apparent that, compared to the conventional SIMP method, clearer structures with fewer gray elements can be achieved earlier in the optimization.

Although the use of NNs for the density prediction cannot reduce the total number of iterations, better designs with lower compliance values and faster convergence properties are achieved. The use of UNet++ for the density prediction can be interpreted as an accelerator that skips some of the iterations. Thus, an even faster convergence might be achieved if this prediction is performed more than once. For this purpose, a new model

needs to be trained, for example, based on the designs obtained after an initial prediction and $10$ additional iterations. Also, the definition of the convergence criterion should be questioned. As seen from the previous figures, the design and compliance values do not change significantly after the first few dozen iterations. Another, more appropriate definition of the termination condition, based, for example, on the compliance evolution or the number of gray elements, could take full advantage of the faster convergence of the proposed method and terminate the optimization earlier.

# Chapter 7

# Conclusion

The goal of this thesis was to develop a more efficient TO algorithm by combining conventional acceleration methods with NNs. In the first approach, these conventional methods were combined with NNs for the reparameterization of the density field. The final results obtained with CNNs achieved, in most cases, lower compliance values, and the corresponding computation time was reduced by more than half. Starting with the conventional SIMP method, a multi-resolution scheme was introduced. By separating the FE analysis mesh from the density mesh, the DOFs of the resulting system are greatly reduced and a much faster FE analysis was achieved. When bilinear quadrilaterals are used for the FE mesh, a relatively high filter radius must be chosen to avoid the formation of QR-patterns. This leads to a loss of fine features and, at the same time, to an increase in gray elements. In order to achieve designs similar to the reference design, higher-order shape functions were added to the multi-resolution scheme. The use of higher-order shape functions improves the analysis quality, and filter radii much smaller than one FE cell can be chosen. The drawback of higher-order shape functions is the increased number of DOFs, which inevitably leads to higher computation times. The higher the order, the higher the number of DOFs. Although the overall computation time for one iteration is slightly reduced compared to the reference method, the gain is not sufficient. Higher-order shape functions come with internal DOFs that are local to one cell. These internal DOFs can be condensed out of cell stiffness matrices to reduce the computation time further. Using the standard Guyan reduction, an exact solution can be achieved, but a costly matrix inversion is needed for each cell in each iteration. It was therefore proposed to use a parametric MOR technique, which retains the parameter dependency. The costly reduction is then performed only once at the beginning of the TO and within the iterations, the reduced matrices can be used directly. The resulting FE analysis is about $2.5$ times faster than in the conventional SIMP method. The drawback of parametric MOR is that the reduced system of equations is not exact anymore. This leads to artifacts and disconnections in the final structure and thus to unusable designs. To prevent this issue, NNs were employed to reparameterize the design variables of the optimization problem. Indeed, the use of NNs can effectively eliminate these artifacts and promising results were achieved, especially with CNNs. In the second approach, NNs were used to locally predict near-optimal density values from stress and strain properties. These properties were either computed on the initial structure or the design obtained after $10$ SIMP iterations. In contrast to the first approach, no other acceleration method was combined with the prediction from the NN, and the optimization was continued with conventional SIMP iterations. The NNs in this case do not only serve as a representation tool but are trained with data generated from standard optimizations. The UNet++ network architecture was implemented for this

purpose. Although the total number of iterations could not be reduced, better designs with lower compliance values were achieved and a faster convergence was observed in the early stage of the optimization.

The performance of the introduced methods was compared on the two-dimensional MBB beam. From the results it is apparent that some of the different acceleration methods benefit from each other: A simple reparameterization with CNNs without any additional measures would still require a costly FE analysis in each iteration, and applying parametric MOR without the use of NNs would lead to infeasible results. The use of NNs in this context comes with two further advantages. The first one is the wide range of different designs that can be obtained. Due to the random initialization of the weights in the NN, the optimization procedure terminates in a different local minimum each time an optimization is conducted. Most of the designs obtained by the proposed algorithm achieved a compliance value better than the reference value of $c = 190.35$. Even the compliance of the worst design with $c = 190.93$ was only slightly higher. Although the overall performance is very promising, some disconnections were observed in rare cases. These disconnections occur only in isolated locations and do not significantly influence the overall performance. In the scope of this thesis, no further investigation was conducted on why these disconnections occur or how they can be avoided. Nevertheless, it would be desirable to eliminate these disconnections to ensure robust and high-quality solutions. The second advantage is the reduced number of iterations. The average number of iterations for the developed algorithm was less than $200$, while the reference problem solved with the conventional SIMP method terminated after $352$ iterations. The total computation time is thus reduced from approximately $250\,\mathrm{s}$ to $100\,\mathrm{s}$.

As demonstrated in the last chapter, the network architecture of the NN for the density parameterization has a significant influence on the resulting designs. Although CNNs seem to perform better than FCNNs, the outcome is affected by different factors such as the choice of the activation function, the choice of the network hyperparameters, but also the choice of the optimizer, and the way the constraints are taken into account. Thus, the potential of FCNNs should not be completely disregarded. In the case of CNNs, it is likely that even better results can be obtained by tuning the network architecture. Further research should be conducted to determine the most efficient and robust architecture. Related to this, the generalization ability of the chosen network should also be analyzed. The CNN introduced in this thesis may work well for similar mesh resolutions, but might not have enough representation power for much larger examples. The efficiency gain is lost, if too much time and effort is spent on defining the network architecture every time the problem scale changes. Ideally, some general rules should be defined that relate the network architecture to the problem's complexity.

The second approach proposed in this thesis, where a UNet++ is used to accelerate the TO, can be seen as a first proof of concept. The results have shown that a faster convergence was achieved and in combination with a different convergence criterion, a significant acceleration of the TO can be expected. Learning a useful mapping in the context of TO is still in its infancy and an active field of research. Many existing attempts

struggle with the huge amount of required data and poor generalization abilities. In the current work, the trained models should, to some degree, be able to generalize to unseen problem settings as local features were given to the NN during the training. However, many configurations were not considered in the training data. The prescribed volume fraction was, for example, fixed to $f = 0.5$, the filter radius to $r_{min} = 2$, and no distributed load conditions were applied. The limits of the proposed model thus need to be analyzed in more detail. In the ideal case, the trained model should continuously collect new data and update itself as soon as a new optimization is conducted. Additionally, some of the conventional acceleration methods introduced in the first chapters could be integrated into the subsequent optimization iterations.

The motivation behind this work was to make large-scale TO more accessible. The proposed algorithms led to promising designs with better compliance values and to reduced computation times or an accelerated convergence. The analysis of static, minimum compliance problems in two dimensions should only pose the starting point. The generalization ability of the proposed methods can be investigated by integrating different objectives and constraints and the efficiency gain is expected to be even more significant when the algorithm is extended to large-scale problems in three dimensions.

# Appendix A

# Legendre Polynomials

The first eight Legendre polynomials are defined as:

$$
\begin{aligned}
P_0(t) &= 1, \\
P_1(t) &= t, \\
P_2(t) &= \frac{1}{2}\left(3t^2 - 1\right), \\
P_3(t) &= \frac{1}{2}\left(5t^3 - 3t\right), \\
P_4(t) &= \frac{1}{8}\left(35t^4 - 30t^2 + 3\right), \\
P_5(t) &= \frac{1}{8}\left(63t^5 - 70t^3 + 15t\right), \\
P_6(t) &= \frac{1}{16}\left(231t^6 - 315t^4 + 105t^2 - 5\right), \\
P_7(t) &= \frac{1}{16}\left(429t^7 - 693t^5 + 315t^3 - 35t\right).
\end{aligned}
\tag{A.1}
$$

# Appendix B

# Neural Network Architectures

## B.1 Fully Connected Neural Network

### B.1.1 Small Network

Table B.1: The network architecture of the FCNN with $13,186$ learnable parameters.

| Layer | Type | Output Dimension | Learnable Parameters |
|---|---|---|---|
| 1 | Input | 2 | 0 |
| 2 | Dense | 32 | 96 |
| | Batch normalization | 32 | 64 |
| | Activation function | 32 | 0 |
| 3 | Dense | 64 | 2,112 |
| | Batch normalization | 64 | 128 |
| | Activation function | 64 | 0 |
| 4 | Dense | 64 | 4,160 |
| | Batch normalization | 64 | 128 |
| | Activation function | 64 | 0 |
| 5 | Dense | 64 | 4,160 |
| | Batch normalization | 64 | 128 |
| | Activation function | 64 | 0 |
| 6 | Dense | 32 | 2,080 |
| | Batch normalization | 32 | 64 |
| | Activation function | 32 | 0 |
| 7 | Dense | 2 | 66 |
| | Classification | 2 | 0 |

### B.1.2 Large Network

Table B.2: The network architecture of the FCNN with $154,562$ learnable parameters.

| Layer | Type | Output Dimension | Learnable Parameters |
|---|---|---|---|
| 1 | Input | 2 | 0 |
| 2 | Dense | 32 | 96 |
| | Batch normalization | 32 | 64 |
| | Activation function | 32 | 0 |
| 3 | Dense | 64 | 2,112 |
| | Batch normalization | 64 | 128 |
| | Activation function | 64 | 0 |
| 4 | Dense | 128 | 8,320 |
| | Batch normalization | 128 | 256 |
| | Activation function | 128 | 0 |
| 5 | Dense | 256 | 33,024 |
| | Batch normalization | 256 | 512 |
| | Activation function | 256 | 0 |
| 6 | Dense | 256 | 65,792 |
| | Batch normalization | 256 | 512 |
| | Activation function | 256 | 0 |
| 7 | Dense | 128 | 32,896 |
| | Batch normalization | 128 | 256 |
| | Activation function | 128 | 0 |
| 8 | Dense | 64 | 8,256 |
| | Batch normalization | 64 | 128 |
| | Activation function | 64 | 0 |
| 9 | Dense | 32 | 2,080 |
| | Batch normalization | 32 | 64 |
| | Activation function | 32 | 0 |
| 10 | Dense | 2 | 66 |
| | Classification | 2 | 0 |

## B.2 Convolutional Neural Network

Table B.3: The network architecture of the CNN with $13,818$ learnable parameters.

| Layer | Type | Output Dimension | Learnable Parameters |
|---|---|---|---|
| 1 | Input | $1 \times 360 \times 120$ | $0$ |
| 2 | Convolution, $7 \times 7$ | $8 \times 360 \times 120$ | $400$ |
| | Batch normalization | $8 \times 360 \times 120$ | $16$ |
| | Activation function | $8 \times 360 \times 120$ | $0$ |
| 3 | Convolution, $7 \times 7$ | $16 \times 360 \times 120$ | $6,288$ |
| | Batch normalization | $16 \times 360 \times 120$ | $32$ |
| | Activation function | $16 \times 360 \times 120$ | $0$ |
| 4 | Convolution, $7 \times 7$ | $8 \times 360 \times 120$ | $6,280$ |
| | Batch normalization | $8 \times 360 \times 120$ | $16$ |
| | Activation function | $8 \times 360 \times 120$ | $0$ |
| 5 | Convolution, $7 \times 7$ | $2 \times 360 \times 120$ | $786$ |
| | Classification | $2 \times 360 \times 120$ | $0$ |

## B.3 UNet++

Table B.4: The network architecture of one convolutional block.

| Layer | Type | Output Dimension | Learnable Parameters |
|---|---|---|---|
| 1 | Input | $C_{in} \times W \times H$ | $0$ |
| 2 | Convolution, $3 \times 3$ | $C_{out} \times W \times H$ | $C_{out} \cdot (C_{in} \cdot 3 \cdot 3 + 1)$ |
| | Batch normalization | $C_{out} \times W \times H$ | $2 \cdot C_{out}$ |
| | Activation function | $C_{out} \times W \times H$ | $0$ |
| 3 | Convolution, $3 \times 3$ | $C_{out} \times W \times H$ | $C_{out} \cdot (C_{out} \cdot 3 \cdot 3 + 1)$ |
| | Batch normalization | $C_{out} \times W \times H$ | $2 \cdot C_{out}$ |
| | Activation function | $C_{out} \times W \times H$ | $0$ |
| 4 | Dropout | $C_{out} \times W \times H$ | $0$ |

Table B.5: The network architecture of the UNet++ with $5.98M$ learnable parameters.

| Layer | Type | Output Dimension | Learnable Parameters |
|---|---|---|---|
| | Input | $5 \times 32 \times 32$ | $0$ |
| 0,0 | ConvBlock($C_{in} = 5$, $C_{out} = 64$, $W = 32$, $H = 32$) | $64 \times 32 \times 32$ | $40,128$ |
| 1,0 | ConvBlock($C_{in} = 64$, $C_{out} = 128$, $W = 16$, $H = 16$) | $128 \times 16 \times 16$ | $221,952$ |
| 2,0 | ConvBlock($C_{in} = 128$, $C_{out} = 256$, $W = 8$, $H = 8$) | $256 \times 8 \times 8$ | $886,272$ |
| 3,0 | ConvBlock($C_{in} = 256$, $C_{out} = 256$, $W = 4$, $H = 4$) | $256 \times 4 \times 4$ | $1,181,184$ |
| 0,1 | ConvBlock($C_{in} = 192$, $C_{out} = 64$, $W = 32$, $H = 32$) | $64 \times 32 \times 32$ | $147,840$ |
| 1,1 | ConvBlock($C_{in} = 384$, $C_{out} = 128$, $W = 16$, $H = 16$) | $128 \times 16 \times 16$ | $590,592$ |
| 2,1 | ConvBlock($C_{in} = 512$, $C_{out} = 256$, $W = 8$, $H = 8$) | $256 \times 8 \times 8$ | $1,771,008$ |
| 0,2 | ConvBlock($C_{in} = 256$, $C_{out} = 64$, $W = 32$, $H = 32$) | $64 \times 32 \times 32$ | $184,704$ |
| 1,2 | ConvBlock($C_{in} = 512$, $C_{out} = 128$, $W = 16$, $H = 16$) | $128 \times 16 \times 16$ | $738,048$ |
| 0,3 | ConvBlock($C_{in} = 320$, $C_{out} = 64$, $W = 32$, $H = 32$) | $64 \times 32 \times 32$ | $221,568$ |
| | Convolution, $3 \times 3$ | $1 \times 32 \times 32$ | $577$ |
| | Classification | $1 \times 32 \times 32$ | $0$ |

# Appendix C

# Computed Stress and Strain Fields

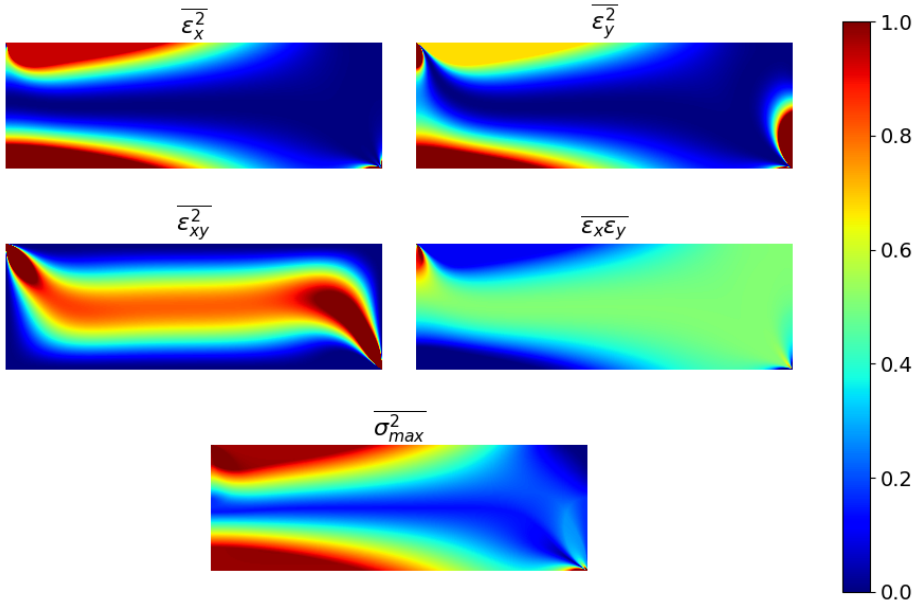



Figure C.1: The normalized elemental stress and strain terms computed on the initial structure.

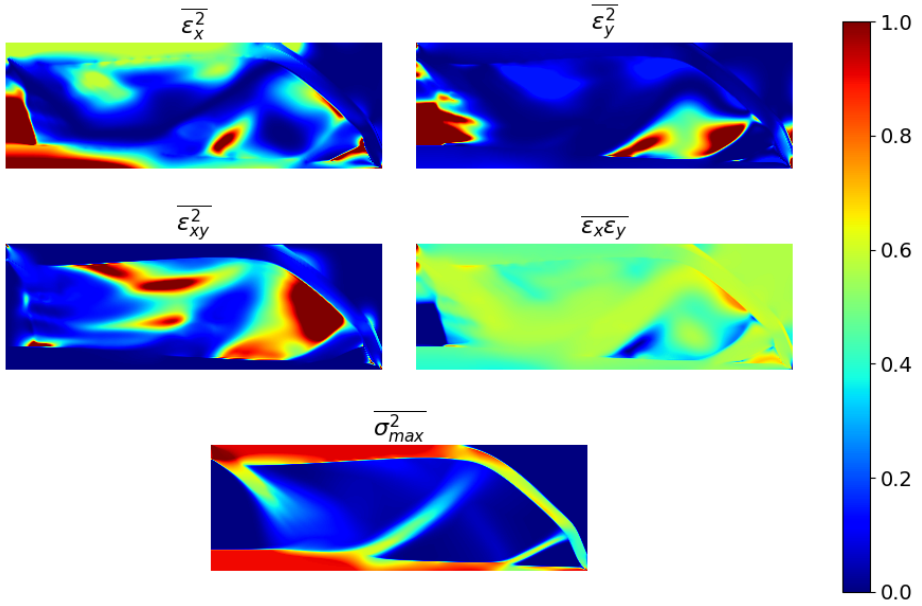



Figure C.2: The normalized elemental stress and strain terms computed on the structure obtained after $10$ iterations.

# Bibliography


ALLAIRE, G., JOUVE, F., & TOADER, A.-M. (2004). Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, *194*(1), 363–393.

AMIR, O., AAGE, N., & LAZAROV, B. S. (2014). On multigrid-cg for efficient topology optimization. *Structural and Multidisciplinary Optimization*, *49*(5), 815–829.

AMIR, O., BENDSØE, M. P., & SIGMUND, O. (2009). Approximate reanalysis in topology optimization. *International Journal for Numerical Methods in Engineering*, *78*(12), 1474–1491.

ANDREASSEN, E., CLAUSEN, A., SCHEVENELS, M., LAZAROV, B. S., & SIGMUND, O. (2011). Efficient topology optimization in matlab using 88 lines of code. *Structural and Multidisciplinary Optimization*, *43*, 1–16.

BABUSKA, I., SZABO, B. A., & KATZ, I. N. (1981). The p-version of the finite element method. *SIAM journal on numerical analysis*, *18*(3), 515–545.

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

BENDSØE, M. P. (1995). *Optimization of structural topology, shape, and material* (Vol. 414). Springer.

BENDSØE, M. P., & SIGMUND, O. (1999). Material interpolation schemes in topology optimization. *Archive of applied mechanics*, *69*(9), 635–654.

BENDSØE, M. P., & KIKUCHI, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer methods in applied mechanics and engineering*, *71*(2), 197–224.

BENDSOE, M. P., & SIGMUND, O. (2003). *Topology optimization: Theory, methods, and applications*. Springer Science & Business Media.

BENNER, P., GUGERCIN, S., & WILLCOX, K. (2015). A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, *57*(4), 483–531.

BORRVALL, T., & PETERSSON, J. (2001). Large-scale topology optimization in 3d using parallel computing. *Computer methods in applied mechanics and engineering*, *190*(46-47), 6201–6229.

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

BRUNS, T. E., & TORTORELLI, D. A. (2001). Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer methods in applied mechanics and engineering*, *190*(26-27), 3443–3459.

CHANDRASEKHAR, A., & SURESH, K. (2021). Tounn: Topology optimization using neural networks. *Structural and Multidisciplinary Optimization*, *63*(3), 1135–1149.

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.

CHOI, Y., OXBERRY, G., WHITE, D., & KIRCHDOERFER, T. (2019). Accelerating design optimization using reduced order models. *arXiv preprint arXiv:1909.11320*.

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.

DIAZ, A., & SIGMUND, O. (1995). Checkerboard patterns in layout optimization. *Structural optimization*, *10*, 40–45.

GLOROT, X., & BENGIO, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

GOODFELLOW, I., BENGIO, Y., & COURVILLE, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

GROEN, J. P., LANGELAAR, M., SIGMUND, O., & RUESS, M. (2017). Higher-order multi-resolution topology optimization using the finite cell method. *International Journal for Numerical Methods in Engineering*, *110*(10), 903–920.

GUO, T., LOHAN, D. J., CANG, R., REN, M. Y., & ALLISON, J. T. (2018). An indirect design representation for topology optimization using variational autoencoder and style transfer. *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 0804.

GUPTA, D. K., LANGELAAR, M., & van KEULEN, F. (2018). Qr-patterns: Artefacts in multiresolution topology optimization. *Structural and Multidisciplinary Optimization*, *58*(4), 1335–1350.

GUYAN, R. J. (1965). Reduction of stiffness and mass matrices. *AIAA journal*, *3*(2), 380–380.

HABER, R. B., JOG, C. S., & BENDSØE, M. P. (1996). A new approach to variable-topology shape design using a constraint on perimeter. *Structural optimization*, *11*, 1–12.

HE, K., ZHANG, X., REN, S., & SUN, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

HERRERO-PÉREZ, D., & CASTEJÓN, P. J. M. (2021). Multi-gpu acceleration of large-scale density-based topology optimization. *Advances in Engineering Software*, *157*, 103006.

HOYER, S., SOHL-DICKSTEIN, J., & GREYDANUS, S. (2019). Neural reparameterization improves structural optimization. *arXiv preprint arXiv:1909.04240*.

IOFFE, S., & SZEGEDY, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 448–456.

JIHONG, Z., HAN, Z., CHUANG, W., LU, Z., SHANGQIN, Y., & ZHANG, W. (2021). A review of topology optimization for additive manufacturing: Status and challenges. *Chinese Journal of Aeronautics*, *34*(1), 91–110.

JOO, Y., YU, Y., & JANG, I. G. (2021). Unit module-based convergence acceleration for topology optimization using the spatiotemporal deep neural network. *IEEE Access*, *9*, 149766–149779.

KALLIORAS, N. A., KAZAKIS, G., & LAGAROS, N. D. (2020). Accelerated topology optimization by means of deep learning. *Structural and Multidisciplinary Optimization*, *62*(3), 1185–1212.

KIM, N. H., DONG, T., WEINBERG, D., & DALIDD, J. (2021). Generalized optimality criteria method for topology optimization. *Applied Sciences*, *11*(7), 3175.

KIM, Y. Y., & YOON, G. H. (2000). Multi-resolution multi-scale topology optimization—a new paradigm. *International Journal of Solids and Structures*, *37*(39), 5529–5559.

KINGMA, D., & BA, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

KIRSCH, U., & PAPALAMBROS, P. Y. (2001). Structural reanalysis for topological modifications–a unified approach. *Structural and multidisciplinary Optimization*, *21*, 333–344.

KOPP, P., RANK, E., CALO, V. M., & KOLLMANNSBERGER, S. (2022). Efficient multi-level hp-finite elements in arbitrary dimensions. *Computer Methods in Applied Mechanics and Engineering*, *401*, 115575.

LIU, J., & MA, Y. (2016). A survey of manufacturing oriented topology optimization methods. *Advances in Engineering Software*, *100*, 161–175.

MARTÍNEZ-FRUTOS, J., MARTÍNEZ-CASTEJÓN, P. J., & HERRERO-PÉREZ, D. (2017). Efficient topology optimization using gpu computing with multilevel granularity. *Advances in Engineering Software*, *106*, 47–62.

MUKHERJEE, S., LU, D., RAGHAVAN, B., BREITKOPF, P., DUTTA, S., XIAO, M., & ZHANG, W. (2021). Accelerating large-scale topology optimization: State-of-the-art and challenges. *Archives of Computational Methods in Engineering*, *28*(7), 4549–4571.

NGUYEN, T. H., LE, C. H., & HAJJAR, J. F. (2013). High-order finite elements for topology optimization. *10th World congress on structural and multidisciplinary optimization. Orlando*.

NGUYEN, T. H., PAULINO, G. H., SONG, J., & LE, C. H. (2010). A computational paradigm for multiresolution topology optimization (mtop). *Structural and Multidisciplinary Optimization*, *41*(4), 525–539.

NOCEDAL, J., & WRIGHT, S. J. (2006). *Numerical optimization*. Springer.

PARVIZIAN, J., DÜSTER, A., & RANK, E. (2012). Topology optimization using the finite cell method. *Optimization and Engineering*, *13*(1), 57–78.

POULSEN, T. A. (2003). A new scheme for imposing a minimum length scale in topology optimization. *International Journal for Numerical Methods in Engineering*, *57*(6), 741–760.

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.

RONNEBERGER, O., FISCHER, P., & BROX, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-*

*Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, 234–241.

SEO, J., & KAPANIA, R. K. (2023). Topology optimization with advanced cnn using mapped physics-based data. *Structural and Multidisciplinary Optimization*, *66*(1), 1–20.

SIGMUND, O. (1994). *Design of material structures using topology optimization* (Doctoral dissertation). Technical University of Denmark Lyngby.

SIGMUND, O. (1997). On the design of compliant mechanisms using topology optimization. *Journal of Structural Mechanics*, *25*(4), 493–524.

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

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

SIGMUND, O., & MAUTE, K. (2013). Topology optimization approaches. *Structural and Multidisciplinary Optimization*, *48*(6), 1031–1055.

SOSNOVIK, I., & OSELEDETS, I. (2019). Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling*, *34*(4), 215–223.

STAINKO, R. (2006). An adaptive multilevel approach to the minimal compliance problem in topology optimization. *Communications in numerical methods in engineering*, *22*(2), 109–118.

SVANBERG, K. (1987). The method of moving asymptotes—a new method for structural optimization. *International journal for numerical methods in engineering*, *24*(2), 359–373.

SZABÓ, B., & BABUŠKA, I. (2021). Finite element analysis: Method, verification and validation.

VEMAGANTI, K., & LAWRENCE, W. E. (2005). Parallel methods for optimality criteria-based topology optimization. *Computer Methods in Applied Mechanics and Engineering*, *194*(34-35), 3637–3667.

WADBRO, E., & BERGGREN, M. (2009). Megapixel topology optimization on a graphics processing unit. *SIAM review*, *51*(4), 707–721.

WALLIN, M., RISTINMAA, M., & ASKFELT, H. (2012). Optimal topologies derived from a phase-field method. *Structural and Multidisciplinary Optimization*, *45*, 171–183.

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.

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.

WOLDSETH, R. V., AAGE, N., BÆRENTZEN, J. A., & SIGMUND, O. (2022). On the use of artificial neural networks in topology optimisation. *Structural and Multidisciplinary Optimization*, *65*(10), 1–36.

XIAO, M., LU, D., BREITKOPF, P., RAGHAVAN, B., DUTTA, S., & ZHANG, W. (2020). On-the-fly model reduction for large-scale structural topology optimization using principal components analysis. *Structural and Multidisciplinary Optimization*, *62*, 209–230.

Xie, Y. M., & Steven, G. P. (1993). A simple evolutionary procedure for structural optimization. *Computers & structures*, *49*(5), 885–896.

Yan, J., Zhang, Q., Xu, Q., Fan, Z., Li, H., Sun, W., & Wang, G. (2022). Deep learning driven real time topology optimisation based on initial stress learning. *Advanced Engineering Informatics*, *51*, 101472.

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.

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

Zhou, M., & Rozvany, G. (1991). The coc algorithm, part ii: Topological, geometrical and generalized shape optimization. *Computer methods in applied mechanics and engineering*, *89*(1-3), 309–336.

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, 3–11.

# Declaration

I hereby affirm that I have independently written the thesis submitted by me and have not used any sources or aids other than those indicated.

Munich, 29.04.2023 Viola Li

Location, Date, Signature