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

Key words. topology optimization, deep learning, inverse problems

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

1. Introduction. The computational discipline of Topology Optimization (TO) generates mechanical structures. Increased computational power made TO an integral tool for engineers in fields ranging from heat transfer [14] and acoustics [17, 50] to fluid [7] and solid mechanics [19]. Still, TO remains computationally costly and often time-consuming [1, 25]. Recently, Deep Learning (DL) approaches tried to address this. While the approaches are promising, the subfield lacks foundations. This paper aims to establish these foundations, focusing on linear elasticity problems.

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

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

While much progress has been made, classical methods’ iterative nature and the in resolution superlinear computational PDE-cost makes classical methods highly computationally demanding – often to the point of practical impossibility [1]. Recent research tries to overcome these challenges using deep learning (DL), i.e., neural networks [5], to speed up and improve the optimization process.
While these DL methods can often solve TO problems in less than a second, they still lack a solid foundation regarding architecture, data preprocessing, evaluation criteria, benchmarks, and datasets. Overall, the subdiscipline of applying DL to TO is in its infancy, with most papers focusing on two-dimensional settings \cite{28, 32, 33, 37, 43, 46, 53, 54}. In particular, the lack of datasets drastically limits the evaluation and comparability of new approaches.

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

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

Our main contributions are:

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

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

2. Preliminaries and related work. We now briefly introduce classical TO, followed by a review of the current literature on DL for TO.

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

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

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

$$\begin{align*}
\text{(2.1a)} & \quad \min_{\rho} \quad F^T u(\rho) \\
\text{(2.1b)} & \quad \text{subject to} \quad K(\rho)u = F, \\
\text{(2.1c)} & \quad \|\rho\|_1 \leq V_{\text{max}}, \\
\text{(2.1d)} & \quad \sigma_{\text{M}}(u) \leq \sigma_{\text{ys}}.
\end{align*}$$

Here (2.1a) is the compliance objective function, $F$ represents the global load distribution, $u$ are the displacement and $K$ is the symmetric positive operator of linear elasticity. $K$ includes the characteristic properties of the used material described by
Young’s modulus and Poisson’s ratio, which we denote by \( E \in \mathbb{R} \) and \( \nu \in [0, 0.5] \), respectively. One constrains the amount of allowed material by \( V_{\text{max}} \) (2.1c), and one may include a stress constraint (2.1d). This is done to ensure that the maximal von Mises stress \( \sigma_{\text{vM}} \) is below the yield stress \( \sigma_{\text{ys}} \) of the material. The von Mises stresses are used to predict mechanical yielding and are derived non-linearly from the displacements \( u \).

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

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

2.2. Neural networks for topology optimization. One can broadly classify advances in TO using DL into three categories [54].

**Reduce or eliminate SIMP iterations:** The first serious attempts of TO via DL aimed to reduce the number of SIMP iterations [3, 43, 49]. In 2017, Sosnovik et al. [43] interpreted two-dimensional TO problems as image-to-image regression problems and were the first to apply convolutional neural networks (CNNs) to TO. Following well-known image processing approaches, they trained a UNet model [41] to map from intermediate SIMP iterations to the final structure. In 2018, Banga et al. [3] transferred these ideas to the three-dimensional case. Xue et al. [49] made each SIMP iteration cheaper by running it on a coarse resolution. They then applied a DL-based super-resolution method to increase the structure’s final granularity. In 2020, Abueidda et al. [2] were the first to use residual neural networks (ResNets) [55] and to consider two-dimensional nonlinear elasticity.

In 2019, Yu et al. [51] developed the first end-to-end learning routine that directly predicts the final density without performing any SIMP iterations. They also created a generative framework to increase the resolution of their predicted designs. This formed the basis for a series of publications [32, 33, 40, 42] on generative adversarial network (GAN)-based TO algorithms [21].

Compared to previous research, Nie et al. [32] and Zhang et al. [53] achieved a better generalization by not directly giving the network the boundary conditions as input. Instead, they passed displacements and von Mises stresses into the network. They argue that neural networks have difficulties extending to previously unseen boundary conditions if the input data is very sparse since the high sparsity of the input matrices leads to high variance of the mapping function.

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

**Neural reparameterization:** Using implicit neural representation for complex signals is an ongoing research topic in computer vision [10] and engineering [30]. Several TO publications [15, 23, 52, 54] feature neural networks to reparameterize the TO’s density field. While Hoyer et al. [23] mapped latent vectors to discrete grid densities, Chandrasekhar & Suresh [9] used multilayer perceptrons to learn a continuous mapping from spatial locations to density values. Since these models are mesh-independent, they can represent the density function at arbitrary resolutions. However, one usually still requires PDE evaluations for training, which is computationally demanding.

While most DL approaches to TO are entirely oblivious to the underlying physics, there is a small number of approaches that have started to include physics-inspired properties into the training process: Banga et al. [3] and Rade et al. [38] augmented the dataset by including rotations and mirrors of given loads and boundary conditions to encourage equivariance. Nie et al. [32] and Zhang et al. [55] involved physics by feeding the network strain and stress information as inputs. Cang et al. [8] and Zhang et al. [54] introduced physics via the loss function design, comparable to *physics-informed neural networks* (PINNs) [39].

To our knowledge, no literature incorporated the problem’s underlying physics by modifying the architecture of the DL model itself. We demonstrate that we can dramatically improve sample efficiency, i.e., the model’s performance when trained on a few training samples, and facilitate geometric reasoning by restricting the hypothesis space to group equivariant models. Cohen & Welling [13] introduced the first group equivariant CNNs in 2016. Nowadays, their application ranges from chemistry [47] and physics [6] to a wide range of tasks in geometric DL [45, 20]. For image processing tasks, Dumont et al. [18] showed that enforcing relevant equivariances can improve generalization performance and

### 3. Datasets.

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

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

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

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

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

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

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

Table 1: Overview of our datasets, called disc and sphere, with the names referring to the shape of their design spaces. Both datasets can be split into subsets with load cases of one or two points of attack. We call these subsets simple and complex, respectively. When we refer to the mixed datasets, i.e., the combination of simple and complex, we sometimes call these disc combined and sphere combined.
Fig. 1: Ground truth examples from the disc and sphere dataset [16]. The densities are defined on a voxel grid and are smoothed for visualization purposes using Taubin smoothing [44].

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

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

1. **Trivial preprocessing.** The input of the neural network is a 7-channel tensor which results from the channel-wise concatenation of Dirichlet boundary conditions $\omega_{\text{Dirichlet}}$, loads $F$, and design space information $\omega_{\text{design}}$. Additionally, we normalize each sample’s $F$ via the mean $\|F\|_\infty$ over all training samples. This is arguably the most straightforward and intuitive type of preprocessing for learned end-to-end TO.

2. **PDE preprocessing.** As proposed by Zhang et al. (2019) [53], we first define an initial density distribution $\rho_{\text{init}}$ that is 1 on $\Omega_1$ and $\Omega_{-1}$, and 0 on $\Omega_0$. For $\rho_{\text{init}}$, we then compute the initial von Mises stresses, which we obtain by solving the PDE for linear elasticity. We normalize the resulting tensor analogously to the normalization of $F$ above. These initial von Mises stresses are then used as a 1-channel input to the neural network. Analogously, it would also be possible to use the full initial stress tensor or the initial displacements as network input.

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

4.2. Architecture. We choose a UNet [41] as our neural network architecture, which is a convolutional encoder-decoder network. The encoder consists of repeated application of convolutions, each followed by a rectified linear unit (ReLU) activation function and a max pooling operation. During the encoding, the encoder of the UNet reduces the spatial information while it increases the feature information. The decoder then extends the feature and spatial information through convolution and upsampling steps and concatenations with high-resolution features from the encoder. See Appendix A for more details on the UNet architecture.
Fig. 2: Illustration of our model pipeline, using an end-to-end learning approach, where the function $f$ represents the neural network. Top: Trivial preprocessing. We concatenate $\omega_{\text{Dirichlet}}$, $\omega_{\text{design}}$, and forces $F = F_1 \cup F_2$ into a 7-channel tensor, which after normalization, we use as input for $f$. Bottom: PDE preprocessing. We compute an initial density distribution $\rho_{\text{init}}$ and corresponding initial displacements and stresses. The image displays the initial von Mises stress distribution, which is a 1-channel tensor.

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

$$f(T^X_g(x)) = T^Y_g(f(x)) \quad \forall g \in G, x \in X,$$

where $T^X_g$ and $T^Y_g$ denote linear group actions in the corresponding spaces $X$ and $Y$ [13]. That is, transforming an input $x$ by a transformation $g$ and then passing it through $f$ should give the same result as first mapping $x$ through $f$ and then transforming the output $f(x)$ (see left image of Figure 3). In many machine learning tasks, we possess prior knowledge about equivariances our predictor should have. Including such knowledge directly into the model can significantly facilitate learning by freeing up model capacity for other factors of variation [48]. Since $G$ is a group, it also contains the identity transformation and a unique inverse transformation $g^{-1}$ for each $g \in G$. Therefore, we can reformulate (4.1) as

$$f(x) = T^{Y^{-1}}_{g^{-1}} \left[f(T^X_g(x))\right],$$

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

$$F^f_G(x) := \frac{1}{|G|} \sum_{g \in G} T^{Y^{-1}}_{g^{-1}} \left[f(T^X_g(x))\right].$$
If \( f : X \to Y \) is \( G \)-equivariant, then for each \( g \in G \) applying \( f \circ T^X_g \) is equivalent to \( T^Y_g \circ f \). Here, \( g \) is represented by a 90° clockwise rotation. Right: Illustration of the equivariance wrapper \( F^f_G \). We transform the input \( x \) via group actions \( T^X_g \) and feed the transformed data into the model \( f \). We then reverse the initial transformation on the outputs via \( T^Y_g^{-1} \) and aggregate the results. This process is called group averaging. The choice of the (finite) transformation group \( G \) is entirely free.

\[
F^f_G \text{ is equivariant with respect to } G \text{ since for each } h \in G \text{ it holds that}
\]
\[
T^Y_{h^{-1}} \left[ F^f_G (T^X_h(x)) \right] = \frac{1}{|G|} \sum_{g \in G} T^Y_{gh^{-1}} \left[ f \left( T^X_g(x) \right) \right] = \frac{1}{|G|} \sum_{gh \in G} T^Y_{g^{-1}} \left[ f \left( T^X_g(x) \right) \right] = \frac{1}{|G|} \sum_{g \in G} T^Y_{g^{-1}} \left[ f \left( T^X_g(x) \right) \right] = F^f_G(x).
\]

For an illustration of \( F^f_G \) see the right image of Figure 3.

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

Natural choices of transformation groups for our applications are the dihedral symmetry group \( D_4 \) and the octahedral symmetry group \( O_h \), which describe all combinations of 90° rotations and reflections in \( \mathbb{R}^2 \) and \( \mathbb{R}^3 \), respectively. Thus the equivariance wrapper applies 8 transformations in the \( D_4 \) and 48 in the \( O_h \) case. Note that group actions on vector fields affect spacial locations and vector directions, unlike on scalar fields. The necessity to account for vector directions makes equivariance wrappers especially suited and flexible.
We want to clarify that the equivariance wrapper fundamentally differs from \textit{data augmentation}, i.e., augmenting the dataset with rotations and reflections. Augmentation averages over the objective, whereas the wrapper averages over the model. This difference endows the wrapper with several significant advantages, most notably: The model does not need to learn about equivariance as it is hard-coded. Consequently, the wrapper is not only approximately equivariant but exactly – importantly, this holds for any input, even for inputs drastically different from the training data.

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

5.1. Training. We train and compare different combinations of preprocessings described in Section 4, each with and without equivariance. Due to the reduced number of voxels in the \( z \)-direction and for simplicity, we chose the dihedral symmetry group, \( D_4 \), as the transformation group for all our experiments. All models are implemented in PyTorch \cite{paszke2019pytorch}. We determine the batch sizes individually, depending on the memory capacity and comparability (see Appendix A). We choose the weighted \textit{binary cross-entropy} (BCE) as our loss function, and calculate the weighting factor based on the training dataset. We use the Adam optimizer \cite{kingma2014adam} with a learning rate of \( 10^{-3} \) in all our experiments.

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

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

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

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

- \textbf{IoU}: In contrast to our loss, the BCE, which is distribution-based, the \textit{Intersection over Union} (IoU) is region-based. It is defined as
  \[
  \text{IoU} = \frac{TP}{TP + FN + FP},
  \]
  where TP, FN and FP denote the number of true positive, false negative and false positive voxel predictions. We limit the evaluation of the IoU to the editable design space \( \Omega_{-1} \). Following Goodhart’s law \cite{goodhart2017anti} and established practices in the segmentation community, we use the IoU as our primary evaluation metric \cite{prat2001evaluation, wang2008automatic}.

- \textbf{Fail percentage}: We consider a prediction failed if the von Mises stress in any voxel exceeds the yield stress by more than 10% or if any voxel with a load case does not connect to one containing Dirichlet conditions. The fail
percentage is the fraction of failed parts. This criterion makes us the first in
the DL for TO community who verify their predictions’ mechanical integrity.
A typical evaluation criterion in the DL for TO literature is the (balanced) accuracy
metric. However, we found that in most cases, IoU yields comparable values but, in
general, reflects the quality of its input more appropriately when used across differ-
ent datasets. Given that we can interpret the binary densities of our structures as
segmentation masks, IoU’s applicability is not surprising as it is the most common
metric for semantic segmentation. Therefore we will use IoU as our main evaluation
criterion.

5.2.2. Sample efficiency. One of the main impediments to learned supervised
TO is the high cost associated with generating new datasets. The necessity to gen-
erate thousands of problems with corresponding ground truth solutions often hinders
practical applicability in real-world situations since the generation of large datasets
can take days, even weeks. Consequently, reducing the number of required training
samples is highly beneficial, e.g., by modifying the DL model design. Therefore, we
put a particular emphasis on the visualization and measurement of our models’ sample
efficiency, i.e., the model’s performance when trained on few training samples.
We visualize the sample efficiency of a model with what we call sample efficiency
curves (SE curves). For each SE curve, we train separate instances of a given model
setup on subsets of the original training dataset of varying sizes. We then evaluate
and compare the performance of these models on a fixed validation dataset, using IoU
and fail percentage as our evaluation criteria. For the disc dataset, we choose training
subsets of sizes 2, 4, 10, 50, 100, 150, 250, 500, 1000, and 1500. For the sphere dataset
we train on 2, 4, 10, 50, 100, and 150 samples. This way, we obtain an individual
SE curve for each evaluation criterion. In order to compare different SE curves of the
same criterion, we report two metrics:
1. The normalized area under the curve (AUC) of that criterion up to a training
sample size of 150, denoted by AUC_{150}. We use this metric to quantify sample
efficiency.
2. The final score, which is the value of that criterion achieved by the model
trained on the largest training subset.

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

5.3.1. Main results. As expected, model performance tends to improve with
increased training data. We also observe dramatic boosts in the UNet’s performance
when incorporating physics via equivariance and trivial+PDE preprocessing. These
improvements are especially visible for low numbers of training samples; see the SE
curves in Figure 4 and AUC_{150} scores in Table 2. From Figure 4, we observe that using
trivial+PDE preprocessing and equivariance in combination leads to a reduction in
the required training samples by two orders of magnitude while maintaining equivalent
IoU scores. Additionally, we notice a reduction in the fail percentage to almost 0%.
The improvements are also evident when visually comparing ground truths with
predictions; see Tables 4 to 7. Further, we observe that adding PDE preprocessing and
equivariance leads to predictions closer to the ground truth densities, which is espe-
Fig. 4: Sample efficiency curves, trained and evaluated on disc combined (left) and sphere combined (right). The x-axis shows the size of the training dataset for the different models on a logarithmic scale. On the y-axis we see performance of the criteria IoU and fail percentage. For each criterion we evaluate models using trivial preprocessing (red), trivial+PDE preprocessing (blue), equivariance (solid line) and no equivariance (dashed line).

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

For an unbiased impression of our model predictions and to avoid cherry-picking, we display 20 random samples and predictions from both datasets in Tables 9 and 10 in Appendix C. We show two predictions for each sample, one using trivial preprocessing only and one using trivial+PDE preprocessing and equivariance. We trained the UNets on 1500 and 150 samples for disc and sphere, respectively.

5.3.2. Generalization. As in all machine learning, generalization is also a problem in its application to TO. The problem led to some publications addressing the issue [32, 53]. We quantify our models’ generalizability by cross-evaluating them on different data subsets not used during training, i.e., we evaluate all models trained on either disc simple, disc complex, or disc combined on each of the others. We proceed analogously for the sphere models and datasets.

We compare the AUC$_{150}$ and final scores for each model based on the corresponding SE curve. For both IoU and fail percentage, we present the results for the AUC$_{150}$ score in Table 2 and the final score in Table 3. We observe that the addition of PDE preprocessing and equivariance improves the generalization capabilities of our models considerably for both IoU and fail percentage, especially on small training sets.

6. Conclusion. We aimed to provide a strong foundation for future research in DL for TO. On the one hand, we proposed and analyzed basic components and principles for designing DL pipelines for TO. On the other hand, we provide two TO datasets enabling the training and comparability of DL methods.
More specifically, method-wise, we focused on physical correctness and sample-efficiency. In practice, the existence of appropriate large-scale training data is costly or simply not realistic; hence it is our conviction that sample-efficiency is a crucial component of any future DL approach to TO.

To achieve this, we developed a physics-inspired approach. We conducted a large-scale ablation study to prove the effectiveness of the two critical components of our approach – on the one hand, PDE-based preprocessing and, on the other, mirror and rotation equivariant architectures. These two key elements drastically improve the overall predictions’ physical correctness and the sample efficiency, i.e., the model performance when trained on only a few samples.

On the data side, we publish two three-dimensional TO datasets with a total count of almost 10,000 problem-ground truth pairs. To our knowledge, these are the first publicly available datasets for TO.
Table 2: Tables containing AUC scores for different combinations of preprocessings and equivariances. We (cross-)evaluate performance of models trained on the disc and sphere datasets and highlight the row-wise best scores in bold.

| eval on          | train on       | trivial prepr. | trivial+PDE prepr. |         |         |
|------------------|----------------|----------------|--------------------|---------|---------|
|                  |                | no equiv.      | equiv.             | no equiv. | equiv. |
| disc simple      | simple         | 0.76           | 0.86               | 0.84     | 0.87    |
|                  | complex        | 0.35           | 0.59               | 0.57     | 0.63    |
|                  | combined       | 0.63           | 0.80               | 0.78     | 0.81    |
| disc complex     | simple         | 0.35           | 0.40               | 0.43     | 0.44    |
|                  | complex        | 0.28           | 0.44               | 0.48     | 0.58    |
|                  | combined       | 0.33           | 0.45               | 0.48     | 0.56    |
| disc combined    | simple         | 0.55           | 0.63               | 0.64     | 0.65    |
|                  | complex        | 0.31           | 0.52               | 0.52     | 0.61    |
|                  | combined       | 0.48           | 0.63               | 0.63     | 0.69    |
| sphere simple    | simple         | 0.28           | 0.41               | 0.50     | 0.61    |
|                  | complex        | 0.28           | 0.32               | 0.39     | 0.44    |
|                  | combined       | 0.30           | 0.37               | 0.43     | 0.56    |
| sphere complex   | simple         | 0.20           | 0.27               | 0.29     | 0.38    |
|                  | complex        | 0.45           | 0.47               | 0.56     | 0.59    |
|                  | combined       | 0.38           | 0.45               | 0.49     | 0.57    |
| sphere combined  | simple         | 0.24           | 0.34               | 0.40     | 0.49    |
|                  | complex        | 0.36           | 0.39               | 0.48     | 0.52    |
|                  | combined       | 0.34           | 0.41               | 0.46     | 0.56    |

(a) AUC scores for the IoU criterion. Higher values indicate better results.

| eval on          | train on       | trivial prepr. | trivial+PDE prepr. |         |         |
|------------------|----------------|----------------|--------------------|---------|---------|
|                  |                | no equiv.      | equiv.             | no equiv. | equiv. |
| disc simple      | simple         | 1.71           | 0.10               | 0.33     | 0.01    |
|                  | complex        | 27.21          | 1.35               | 1.11     | 1.10    |
|                  | combined       | 2.95           | 0.76               | 0.89     | 0.05    |
| disc complex     | simple         | 39.11          | **10.21**          | 22.30    | 14.50   |
|                  | complex        | 49.39          | 16.13              | 9.37     | **5.07**|
|                  | combined       | 37.58          | 17.07              | 14.95    | **8.47**|
| disc combined    | simple         | 20.41          | **5.16**           | 16.31    | 12.26   |
|                  | complex        | 38.30          | 8.74               | 5.24     | **2.69**|
|                  | combined       | 20.27          | 8.92               | 7.92     | **4.26**|
| sphere simple    | simple         | 48.03          | 12.16              | 9.55     | **4.11**|
|                  | complex        | 59.52          | 36.84              | 8.41     | **6.06**|
|                  | combined       | 56.51          | 24.02              | 27.93    | **7.73**|
| sphere complex   | simple         | 94.37          | 68.17              | 72.60    | **34.95**|
|                  | complex        | 63.53          | 37.44              | 14.62    | **3.55**|
|                  | combined       | 83.33          | 54.13              | 26.61    | **6.42**|
| sphere combined  | simple         | 71.20          | 40.17              | 41.08    | **19.53**|
|                  | complex        | 61.52          | 37.14              | 11.51    | **4.80**|
|                  | combined       | 69.92          | 39.08              | 27.27    | **7.08**|

(b) AUC scores for the fail% criterion.
Table 3: Tables containing the final scores for different combinations of preprocessings and equivariances. We (cross-)evaluate performance of models trained on the disc and sphere datasets and highlight the row-wise best scores in bold.

(a) Final scores of the IoU criterion. Higher values indicate better results.

| eval on     | train on | trivial prepr. | trivial+PDE prepr. |
|-------------|----------|----------------|--------------------|
|             |          | no equiv. | equiv. | no equiv. | equiv. |
| disc simple | simple   | 0.87     | 0.89   | 0.90     | 0.91   |
|             | complex  | 0.52     | 0.61   | 0.61     | 0.62   |
|             | combined | 0.77     | 0.84   | 0.83     | 0.85   |
| disc complex| simple   | 0.37     | 0.40   | 0.44     | 0.45   |
|             | complex  | 0.40     | 0.48   | 0.52     | 0.62   |
|             | combined | 0.40     | 0.48   | 0.52     | 0.59   |
| disc combined| simple  | 0.62    | 0.65   | 0.67     | 0.68   |
|              | complex  | 0.46    | 0.55   | 0.57     | 0.62   |
|              | combined | 0.59    | 0.66   | 0.67     | 0.72   |
| sphere simple| simple  | 0.26    | 0.46   | 0.58     | 0.67   |
|              | complex  | 0.29    | 0.35   | 0.39     | 0.48   |
|              | combined | 0.34    | 0.44   | 0.52     | 0.66   |
| sphere complex| simple  | 0.18    | 0.23   | 0.35     | 0.40   |
|              | complex  | 0.46    | 0.55   | 0.64     | 0.67   |
|              | combined | 0.43    | 0.50   | 0.58     | 0.63   |
| sphere combined| simple  | 0.22    | 0.34   | 0.46     | 0.53   |
|              | complex  | 0.37    | 0.45   | 0.51     | 0.57   |
|              | combined | 0.38    | 0.47   | 0.55     | 0.65   |

(b) Final scores of the fail% criterion. Lower values indicate better results.

| eval on     | train on | trivial prepr. | trivial+PDE prepr. |
|-------------|----------|----------------|--------------------|
|             |          | no equiv. | equiv. | no equiv. | equiv. |
| disc simple | simple   | 0.00     | 0.00   | 0.00     | 0.00   |
|             | complex  | 0.00     | 0.00   | 0.00     | 0.00   |
|             | combined | 0.00     | 0.00   | 0.00     | 0.00   |
| disc complex| simple   | 32.50    | 15.50  | 24.50    | 13.50  |
|             | complex  | 12.50    | 11.00  | 5.00     | 3.50   |
|             | combined | 25.50    | 15.00  | 7.00     | 5.00   |
| disc combined| simple   | 16.25   | 7.75   | 12.25   | 6.75   |
|              | complex  | 6.25     | 5.50   | 2.50     | 1.75   |
|              | combined | 12.75    | 7.50   | 3.50     | 2.50   |
| sphere simple| simple  | 52.78   | 8.33   | 0.00     | 0.00   |
|              | complex  | 69.44   | 13.89  | 0.00     | 0.00   |
|              | combined | 33.33   | 8.33   | 8.33     | 0.00   |
| sphere complex| simple | 99.97   | 61.11  | 38.89    | 13.89  |
|              | complex  | 66.67   | 13.89  | 0.00     | 0.00   |
|              | combined | 69.44   | 38.89  | 8.33     | 0.00   |
| sphere combined| simple | 76.39   | 34.72  | 19.44    | 6.94   |
|              | complex  | 68.06   | 13.89  | 0.00     | 0.00   |
|              | combined | 51.39   | 23.61  | 8.33     | 0.00   |
Table 4: Model predictions of two different problems from the disc simple validation dataset, using the UNet with different preprocessings and equivariances. We train the models on subsets of the dataset and vary the training size along the columns of the table. At the boxes below the tables we show the corresponding ground truth density for each problem.

| prepr. | equiv. | 10   | 50   | 100  | 500  | 1500 |
|--------|--------|------|------|------|------|------|
| trivial|        | ![Image](trivial_10.png) | ![Image](trivial_50.png) | ![Image](trivial_100.png) | ![Image](trivial_500.png) | ![Image](trivial_1500.png) |
|        | ✓      | ![Image](trivial+PDE_10.png) | ![Image](trivial+PDE_50.png) | ![Image](trivial+PDE_100.png) | ![Image](trivial+PDE_500.png) | ![Image](trivial+PDE_1500.png) |

| prepr. | equiv. | 10   | 50   | 100  | 500  | 1500 |
|--------|--------|------|------|------|------|------|
| trivial|        | ![Image](trivial_10.png) | ![Image](trivial_50.png) | ![Image](trivial_100.png) | ![Image](trivial_500.png) | ![Image](trivial_1500.png) |
|        | ✓      | ![Image](trivial+PDE_10.png) | ![Image](trivial+PDE_50.png) | ![Image](trivial+PDE_100.png) | ![Image](trivial+PDE_500.png) | ![Image](trivial+PDE_1500.png) |
Table 5: Model predictions of two different problems from the **disc complex** validation dataset, using the UNet with different preprocessings and equivariances. We train the models on subsets of the dataset and vary the training size along the columns of the table. At the boxes below the tables we show the corresponding ground truth density for each problem.

| prepr. | equiv. | 10     | 50     | 100    | 500    | 1500   |
|--------|--------|--------|--------|--------|--------|--------|
| trivial|        | ![image](image1.png) | ![image](image2.png) | ![image](image3.png) | ![image](image4.png) | ![image](image5.png) |
| ✓      |        | ![image](image6.png) | ![image](image7.png) | ![image](image8.png) | ![image](image9.png) | ![image](image10.png) |
| trivial+PDE |        | ![image](image11.png) | ![image](image12.png) | ![image](image13.png) | ![image](image14.png) | ![image](image15.png) |
| ✓      |        | ![image](image16.png) | ![image](image17.png) | ![image](image18.png) | ![image](image19.png) | ![image](image20.png) |

| prepr. | equiv. | 10     | 50     | 100    | 500    | 1500   |
|--------|--------|--------|--------|--------|--------|--------|
| trivial|        | ![image](image21.png) | ![image](image22.png) | ![image](image23.png) | ![image](image24.png) | ![image](image25.png) |
| ✓      |        | ![image](image26.png) | ![image](image27.png) | ![image](image28.png) | ![image](image29.png) | ![image](image30.png) |
| trivial+PDE |        | ![image](image31.png) | ![image](image32.png) | ![image](image33.png) | ![image](image34.png) | ![image](image35.png) |
| ✓      |        | ![image](image36.png) | ![image](image37.png) | ![image](image38.png) | ![image](image39.png) | ![image](image40.png) |
Table 6: Model predictions of two different problems from the sphere simple validation dataset, using the UNet with different preprocessings and equivariances. We train the models on subsets of the dataset and vary the training size along the columns of the table. At the boxes below the tables we show the corresponding ground truth density for each problem.

| prepr. | equiv. | 10  | 50  | 100 | 500 | 1500 |
|--------|--------|-----|-----|-----|-----|------|
| trivial |        | ![Image](image1.png) | ![Image](image2.png) | ![Image](image3.png) | ![Image](image4.png) | ![Image](image5.png) |
|        | ✓      | ![Image](image6.png) | ![Image](image7.png) | ![Image](image8.png) | ![Image](image9.png) | ![Image](image10.png) |
| trivial+PDE |        | ![Image](image11.png) | ![Image](image12.png) | ![Image](image13.png) | ![Image](image14.png) | ![Image](image15.png) |
|        | ✓      | ![Image](image16.png) | ![Image](image17.png) | ![Image](image18.png) | ![Image](image19.png) | ![Image](image20.png) |

| prepr. | equiv. | 10  | 50  | 100 | 500 | 1500 |
|--------|--------|-----|-----|-----|-----|------|
| trivial |        | ![Image](image21.png) | ![Image](image22.png) | ![Image](image23.png) | ![Image](image24.png) | ![Image](image25.png) |
|        | ✓      | ![Image](image26.png) | ![Image](image27.png) | ![Image](image28.png) | ![Image](image29.png) | ![Image](image30.png) |
| trivial+PDE |        | ![Image](image31.png) | ![Image](image32.png) | ![Image](image33.png) | ![Image](image34.png) | ![Image](image35.png) |
|        | ✓      | ![Image](image36.png) | ![Image](image37.png) | ![Image](image38.png) | ![Image](image39.png) | ![Image](image40.png) |
Table 7: Model predictions of two different problems from the sphere complex validation dataset, using the U-Net with different preprocessings and equivariances. We train the models on subsets of the dataset and vary the training size along the columns of the table. At the boxes below the tables we show the corresponding ground truth density for each problem.

| prepr. | equiv. | training samples |
|--------|--------|------------------|
| trivial | ✓      | ![image1](image1) |
|        | ✓      | ![image2](image2) |
| trivial+PDE | ✓      | ![image3](image3) |
|        | ✓      | ![image4](image4) |

| prepr. | equiv. | training samples |
|--------|--------|------------------|
| trivial | ✓      | ![image5](image5) |
|        | ✓      | ![image6](image6) |
| trivial+PDE | ✓      | ![image7](image7) |
|        | ✓      | ![image8](image8) |

sphere complex
Acknowledgments. We thank Marco Gosch, Rielson Falck, Christian Knorr (ArianeGroup) and Daniel Siegel (Synera) for providing the datasets. We also thank Denis Regenbrecht and Susanne Heckrodt (DLR, German Aerospace Center) and Janek Gödeke (University of Bremen) for their support.

REFERENCES

[1] N. Aage, E. Andreassen, and B. S. Lazarov, Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework, Structural and Multidisciplinary Optimization, 51 (2015), pp. 565–572.

[2] D. W. Abueidda, S. Koric, and N. A. Sobh, Topology optimization of 2d structures with nonlinearities using deep learning, Computers & Structures, 237 (2020), p. 106283.

[3] S. Banga, H. Gehani, S. Bhilare, S. Patel, and L. Kára, 3d topology optimization using convolutional neural networks, arXiv preprint arXiv:1808.07440, (2018).

[4] M. P. Bendsoe and O. Sigmund, Topology optimization: theory, methods, and applications, Springer Science & Business Media, 2003.

[5] Y. Bengio, I. Goodfellow, and A. Courville, Deep learning, vol. 1, MIT press Cambridge, MA, USA, 2017.

[6] A. Bogatskiy, B. Anderson, J. Offermann, M. Roussi, D. Miller, and R. Kondor, Lorentz group equivariant neural network for particle physics, in International Conference on Machine Learning, PMLR, 2020, pp. 992–1002.

[7] T. Borrvall and J. Petersson, Topology optimization of fluids in stokes flow, International journal for numerical methods in fluids, 41 (2003), pp. 77–107.

[8] R. Cang, H. Yao, and Y. Ren, One-shot generation of near-optimal topology through theory-driven machine learning, Computer-Aided Design, 109 (2019), pp. 12–21.

[9] A. Chandrasekhar and K. Suresh, Tounn: topology optimization using neural networks, Structural and Multidisciplinary Optimization, 63 (2021), pp. 1135–1149.

[10] H. Chen, B. He, H. Wang, Y. Ren, S. N. Lim, and A. Shrivastava, Nerv: Neural representations for videos, Advances in Neural Information Processing Systems, 34 (2021), pp. 21557–21568.

[11] H. Chi, Y. Zhang, T. L. E. Tang, L. Mirabella, L. Dalloro, L. Song, and G. H. Paulino, Universal machine learning for topology optimization, Computer Methods in Applied Mechanics and Engineering, 375 (2021), p. 112739.

[12] Ō. Čiček, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, 3d u-net: learning dense volumetric segmentation from sparse annotation, in International conference on medical image computing and computer-assisted intervention, Springer, 2016, pp. 424–432.

[13] T. Cohen and M. Welling, Group equivariant convolutional networks, in International conference on machine learning, PMLR, 2016, pp. 2990–2999.

[14] E. M. Deine, Multiphysics topology optimization of heat transfer and fluid flow systems, 715 (2009).

[15] H. Deng and A. C. To, Topology optimization based on deep representation learning (drl) for compliance and stress-constrained design, Computational Mechanics, 66 (2020), pp. 449–469.

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

[17] M. B. Dühring, J. S. Jensen, and O. Sigmund, Acoustic design by topology optimization, Journal of sound and vibration, 317 (2008), pp. 557–575.

[18] B. Dumont, S. Maggio, and P. Montalvo, Robustness of rotation-equivariant networks to adversarial perturbations, arXiv preprint arXiv:1802.06627, (2018).

[19] H. A. Eschenauer and N. Olhoff, Topology optimization of continuum structures: a review, Appl. Mech. Rev., 54 (2001), pp. 331–390.

[20] J. E. Gerken, J. Aronsson, O. Carlsson, H. Linander, F. Ohlsson, C. Petersson, and D. Persson, Geometric deep learning and equivariant neural networks, arXiv preprint arXiv:2105.13926, (2021).

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, Advances in neural information processing systems, 27 (2014).

[22] C. Goodhart, Problems of monetary management: the uk experience in papers in monetary economics, Monetary Economics, 1 (1975).

[23] S. Hoyer, J. Sohl-Dickstein, and S. Greydanus, Neural reparameterization improves structural optimization, arXiv preprint arXiv:1909.04240, (2019).
SÖREN DITTMER, DAVID ERZMANN, HENRIK HARMS, PETER MAASS

[24] S. IOFFE and C. SZEGEDY, 
Batch normalization: Accelerating deep network training by reducing internal covariate shift, in International conference on machine learning, PMLR, 2015, pp. 448–456.

[25] P. D. L. JENSEN, F. WANG, I. DIMINO, and O. SIGMUND, 
Topology optimization of large-scale 3d morphing wing structures, in Actuators, vol. 10, MDPI, 2021, p. 217.

[26] K. KAWAGUCHI, L. P. KAELBLEING, and Y. BENGIO, 
Generalization in deep learning, arXiv preprint arXiv:1710.05468, (2017).

[27] D. P. KINGMA and J. BA, 
Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, (2014).

[28] S. LEE, H. KIM, Q. X. LIEU, and J. LEE, 
Cnn-based image recognition for topology optimization, Knowledge-Based Systems, 198 (2020), p. 105887.

[29] J. MA, 
Segmentation loss odyssey, arXiv preprint arXiv:2005.13449, (2020).

[30] D. MROWCA, C. ZHUANG, E. WANG, N. HABER, L. F. FEI-FEI, J. TENENBAUM, and D. L. YAMINS, 
Flexible neural representation for physics prediction, Advances in neural information processing systems, 31 (2018).

[31] R. L. MURPHY, B. SRINIVASAN, Y. RAO, and B. RIBEIRO, 
Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs, arXiv preprint arXiv:1811.01900, (2018).

[32] Z. NIE, T. LIN, H. JIANG, and L. B. KARA, 
Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain, Journal of Mechanical Design, 143 (2021).

[33] S. OH, Y. JUNG, S. KIM, I. LEE, and N. KANG, 
Deep generative design: Integration of topology optimization and generative models, Journal of Mechanical Design, 141 (2019).

[34] A. PASKRE, S. GROSS, S. CHINTALA, G. CHANAN, E. YANG, Z. DeVITO, Z. LIN, A. DESMAISON, L. ANTIGA, and A. LEBER, 
Automatic differentiation in pytorch, (2017).

[35] O. PUNY, M. ATZMON, H. BEN-HAMU, E. J. SMITH, I. MISRA, A. GROVER, and Y. LIPMAN, 
Frame averaging for invariant and equivariant network design, arXiv preprint arXiv:2110.03336, (2021).

[36] F. PÉREZ-GARCÍA, 
Pytorch implementation of 2d and 3d u-net (v0.7.5). https://github.com/fepegar/unet, 2020.

[37] C. QIAN and W. YE, 
Accelerating gradient-based topology optimization design with dual-model artificial neural networks, Structural and Multidisciplinary Optimization, 63 (2021), pp. 1687–1707.

[38] J. RADE, A. BALU, E. HERRON, J. PATRAK, R. RANADE, S. SARKAR, and A. KRISHNAMURTHY, 
Physics-consistent deep learning for structural topology optimization, arXiv preprint arXiv:2012.05359, (2020).

[39] M. RAISI, P. PERDIKARIS, and G. E. KARMIADAKIS, 
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational physics, 378 (2019), pp. 686–707.

[40] S. RAWAT and M.-H. H. SHEN, 
A novel topology optimization approach using conditional deep learning, arXiv preprint arXiv:1901.04859, (2019).

[41] O. RONNEBERGER, P. FISCHER, and T. BROX, 
U-net: Convolutional networks for biomedical image segmentation, in International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[42] M.-H. H. SHEN and L. CHEN, 
A new cgan technique for constrained topology design optimization, arXiv preprint arXiv:1901.07675, (2019).

[43] I. SOSNOVIK and I. OSELEDETS, 
Neural networks for topology optimization, Russian Journal of Numerical Analysis and Mathematical Modelling, 34 (2019), pp. 215–223.

[44] G. TAUBIN, 
Curve and surface smoothing without shrinkage, in Proceedings of IEEE international conference on computer vision, IEEE, 1995, pp. 852–857.

[45] N. THOMAS, T. SMIDT, S. KEARNES, L. YANG, L. LI, K. KOHLHOFF, and P. RILEY, 
Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds, arXiv preprint arXiv:1802.08219, (2018).

[46] C. WANG, S. YAO, Z. WANG, and J. HU, 
Deep super-resolution neural network for structural topology optimization, Engineering Optimization, 53 (2021), pp. 2108–2121.

[47] M. WEILER, M. GEIGER, M. WELLING, W. BOOMSMA, and T. S. COHEN, 
3d steerable cnns: Learning rotationally equivariant features in volumetric data, Advances in Neural Information Processing Systems, 31 (2018).

[48] M. WEILER, F. A. HAMPRECHT, and M. STORATH, 
Learning steerable filters for rotation equivariant cnns. (2018), pp. 849–858.

[49] L. XUE, J. LIU, G. WEN, and H. WANG, 
Efficient, high-resolution topology optimization method based on convolutional neural networks, Frontiers of Mechanical Engineering, 16 (2021), pp. 448–456.
[50] G. H. Yoon, J. S. Jensen, and O. Sigmund, Topology optimization of acoustic-structure interaction problems using a mixed finite element formulation, International Journal for Numerical Methods in Engineering, 70 (2007), pp. 1049–1075.

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

[52] J. Zehnder, Y. Li, S. Coros, and B. Thomaszewski, Ntopo: Mesh-free topology optimization using implicit neural representations, Advances in Neural Information Processing Systems, 34 (2021), pp. 10368–10381.

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

[54] Z. Zhang, Y. Li, W. Zhou, X. Chen, W. Yao, and Y. Zhao, Tonr: An exploration for a novel way combining neural network with topology optimization, Computer Methods in Applied Mechanics and Engineering, 386 (2021), p. 114083.

[55] Z. Zhang, Q. Liu, and Y. Wang, Road extraction by deep residual u-net, IEEE Geoscience and Remote Sensing Letters, 15 (2018), pp. 749–753.
Appendix A. Network architecture and training details.

Our UNet architecture is inspired by Çiçek et al.[12] and based on the implementation from Pérez-García[36]. Like in Çiçek et al.[12], each layer of our encoder contains two $3 \times 3 \times 3$ convolutions, each followed by batch normalization [24] and a rectified linear unit (ReLU) activation function, and then a max pooling operation. For the sphere dataset, we choose a pooling kernel size of $2 \times 2 \times 2$; for the disc dataset, we choose $2 \times 2 \times 1$ to account for the lower number of voxels in the $z$-direction. In the decoding path, each layer consists of a nearest neighbor upsampling step, followed by two $3 \times 3 \times 3$ convolutions with batch normalization and ReLU activation functions. In the last layer, a $1 \times 1 \times 1$ convolution with a sigmoid activation function reduces the number of output channels to 1 and acts as a binary voxel-wise classifier.

We use skip connections to pass information from layers of equal resolution in the encoding path to the decoder and apply a dropout rate of $0.3$ to reduce overfitting. We use five encoding and four decoding blocks. Each encoding block doubles the number of channels, starting with 10 output channels for the first block. We choose the padding, so the image resolution stays unaffected by the convolutions. Based on memory availability, we set the batch size to 128 when training on the disc dataset and 28 when training on the sphere dataset. All models have been implemented in PyTorch [34] with a total parameter count of approximately $1.6M$. For training we used an Nvidia GeForce GTX 1080 Ti GPU with 11GB VRAM. For the number of training epochs and the approximate training times for different models trained on the largest training subset, see Table 8.

Table 8: The number of training epochs and the approximate training time for UNets with different preprocessings and equivariances, trained on the largest training subsets of the disc and sphere combined datasets.

| preprocessing | equivariance | training epochs | training time (in minutes) |
|---------------|--------------|----------------|---------------------------|
| trivial       |              | 210            | 12                        |
| trivial       | ✓            | 120            | 34                        |
| trivial+PDE   | ✓            | 230            | 14                        |
| trivial+PDE   |              | 110            | 32                        |

(a) disc combined

| preprocessing | equivariance | training epochs | training time (in minutes) |
|---------------|--------------|----------------|---------------------------|
| trivial       |              | 100            | 5                         |
| trivial       | ✓            | 80             | 17                        |
| trivial+PDE   | ✓            | 170            | 9                         |
| trivial+PDE   |              | 140            | 31                        |

(b) sphere combined

Appendix B. Ablations and negative results. This section discusses several model alterations that did not lead to noticeable improvements in our numerical experiments. We think these results are still valuable to the community.

Above, we exclusively used the dihedral symmetry group, $D_4$, in the equivariance wrapper. For the sphere dataset, we also examined the performance of the octahedral symmetry group $O_h$. While still constituting a significant benefit over not utilizing
equivariance, this method was inferior to the use of the $D_4$ symmetry group. We suspect this is due to the fixed location of the sphere dataset’s Dirichlet condition, negating the benefit of generalizing to more diverse locations of boundary conditions.

In addition to the preprocessings we examined in Subsection 5.3.1, we experimented with various other preprocessing combinations. For instance, for an alternative preprocessing strategy we construct a binary mask to extract all voxels with either Dirichlet boundary or load cases. The convex hull of this mask is then given as the set of voxels included in the smallest convex polygon that surrounds all these voxels. In case of one load point and one Dirichlet voxel this results in the simplest possible density distribution that leads to a connected mechanical structure. This convex hull density distribution is then used as a 1-channel input to the neural network. We found that convex hull preprocessing did not lead to any measurable improvements on the validation dataset, neither on its own nor in combination with other preprocessings. We speculate that this is due to the convex hull being invariant to the direction of the applied forces. We also found that applying PDE preprocessing on its own produced similar but somewhat inferior results compared to the trivial+PDE preprocessing. Also, adding the full initial stress tensor and initial displacements to the PDE preprocessing did not improve over the presented PDE preprocessing.

We also experimented with different network architectures. In addition to the UNet, we considered a ResNet [55] of depth 10 with 5-layer CNN blocks. However, despite the UNet requiring less VRAM and computation time, it reliably outperformed the ResNet.
Appendix C. Examples of model predictions.

Table 9: Random samples from the disc dataset. The first columns displays prediction from the UNet with trivial preprocessing and without equivariance. The second columns shows predictions from the UNet with trivial+PDE preprocessing and equivariance, which is our best model pipeline. All models have been trained on 1500 samples. In the third columns we show the ground truths densities corresponding to each sample.

| UNet       | UNet +physics | ground truth |
|------------|---------------|--------------|
| ![Sample](image1) | ![Sample](image2) | ![Sample](image3) |
| ![Sample](image4) | ![Sample](image5) | ![Sample](image6) |
| ![Sample](image7) | ![Sample](image8) | ![Sample](image9) |
| ![Sample](image10) | ![Sample](image11) | ![Sample](image12) |
| ![Sample](image13) | ![Sample](image14) | ![Sample](image15) |
| ![Sample](image16) | ![Sample](image17) | ![Sample](image18) |
| ![Sample](image19) | ![Sample](image20) | ![Sample](image21) |
| ![Sample](image22) | ![Sample](image23) | ![Sample](image24) |

(a) disc simple

(b) disc complex
Table 10: Random samples from the sphere dataset. The first columns displays prediction from the UNet with trivial preprocessing and without equivariance. The second columns shows predictions from the UNet with trivial+PDE preprocessing and equivariance, which is our best model pipeline. All models have been trained on 150 samples. In the third columns we show the ground truths densities corresponding to each sample.

| UNet | UNet +physics | ground truth |
|------|---------------|--------------|
| ![Sphere Simple Prediction](image1.png) | ![Sphere Simple Prediction](image2.png) | ![Sphere Simple Ground Truth](image3.png) |
| ![Sphere Simple Prediction](image4.png) | ![Sphere Simple Prediction](image5.png) | ![Sphere Simple Ground Truth](image6.png) |
| ![Sphere Simple Prediction](image7.png) | ![Sphere Simple Prediction](image8.png) | ![Sphere Simple Ground Truth](image9.png) |
| ![Sphere Simple Prediction](image10.png) | ![Sphere Simple Prediction](image11.png) | ![Sphere Simple Ground Truth](image12.png) |
| ![Sphere Simple Prediction](image13.png) | ![Sphere Simple Prediction](image14.png) | ![Sphere Simple Ground Truth](image15.png) |
| ![Sphere Simple Prediction](image16.png) | ![Sphere Simple Prediction](image17.png) | ![Sphere Simple Ground Truth](image18.png) |
| ![Sphere Complex Prediction](image19.png) | ![Sphere Complex Prediction](image20.png) | ![Sphere Complex Ground Truth](image21.png) |
| ![Sphere Complex Prediction](image22.png) | ![Sphere Complex Prediction](image23.png) | ![Sphere Complex Ground Truth](image24.png) |
| ![Sphere Complex Prediction](image25.png) | ![Sphere Complex Prediction](image26.png) | ![Sphere Complex Ground Truth](image27.png) |
| ![Sphere Complex Prediction](image28.png) | ![Sphere Complex Prediction](image29.png) | ![Sphere Complex Ground Truth](image30.png) |
| ![Sphere Complex Prediction](image31.png) | ![Sphere Complex Prediction](image32.png) | ![Sphere Complex Ground Truth](image33.png) |
| ![Sphere Complex Prediction](image34.png) | ![Sphere Complex Prediction](image35.png) | ![Sphere Complex Ground Truth](image36.png) |

(a) sphere simple  
(b) sphere complex