Structural and Multidisciplinary Optimization (2026) 69:162
https://doi.org/10.1007/s00158-026-04362-2

RESEARCH PAPER

Fail-safe topology optimization using artificial neural networks

Benedikt Hamann[1] · **Benedikt Kriegesmann**[1]

Received: 25 February 2026 / Revised: 12 May 2026 / Accepted: 15 May 2026

Abstract
Accounting for fail-safety in topology optimization requires the consideration of a large number of failure scenarios. For each scenario, the structural responses must be evaluated, rendering explicit fail-safe topology optimization computationally extremely demanding. In recent years, artificial neural networks (ANNs) have gained increasing attention in topology optimization as a means to overcome existing limitations and advance the state of the art. To explore the potential and limitations of ANN-based fail-safe topology optimization, this work proposes a novel framework for generating fail-safe designs. Within this framework, fail-safe designs are derived from designs optimized for minimal compliance using an ANN. In addition, a post-processing step is introduced to address common problems associated with topology-optimized designs predicted by ANNs. Compared to the traditional approach, the proposed framework reduces the computing time required to generate fail-safe designs by 84.98%. However, due to the high initial cost of data generation and model training, achieving a net computational benefit would require very large number of applications. Compared with deterministic designs, the generated fail-safe designs achieve a median reduction in damage compliance of 96.02%. However, compared with the traditional fail-safe approach, the damage compliance is 85.32% higher. Moreover, the results indicate limited versatility, with performance degrading as problem formulations deviate increasingly from the training distribution.

Keywords Fail-safe optimization · Artificial neural networks · Damage tolerance

1 Introduction

In safety-critical applications, such as aircraft design, structures must be resilient to local damage. The ability to sustain the intended load in the event of a damaged structural member is designated as "fail-safe." Sun et al. (1976) first introduced fail-safety as a constraint to truss structure optimization by evaluating every possible damage. In topology optimization, fail-safety was first explicitly considered by Jansen et al. (2013), who introduced a square damage patch, modeling the damage by assigning the minimum stiffness to all affected finite elements. In their formulation, every finite element is used as a damage patch origin, resulting in a large number of damage cases to be evaluated, making the approach computationally extremely demanding. To improve computational efficiency, some researchers focused on reducing the number of damage cases. Zhou and Fleury (2016) found that with no overlap of damage patches, similar fail-safe properties are achieved while the number of damage cases is reduced significantly. To only evaluate critical damage cases, Wang et al. (2020) employed the von Mises stress criterion, while Ambrozkiewicz and Kriegesmann (2018) introduced a material threshold that damage patches must exceed to be considered. Zhang et al. (2023) further reduced the number of evaluated scenarios through stochastic sampling within a two-level framework. Herrero-Pérez and Picó-Vicente (2024) combined domain decomposition and adaptive mesh refinement with a hierarchical parallelization scheme that exploits parallelism across damage scenarios and spatial subdomains to efficiently solve large-scale problems. Alternative formulations have also been proposed. Instead of relying on a classical min–max formulation, Peng and Sui (2021) introduced a smooth single-objective approach by employing reciprocal design variables and linearizing the structural response. Stolpe (2019) formulated fail-safe truss optimization as a worst-case compliance problem and proposed an active-set algorithm to manage active and inactive

Responsible Editor: Ole Sigmund.

✉ Benedikt Hamann
benedikt.hamann@tuhh.de

[1] Working Group Structural mechanics in lightweight design, Hamburg University of Technology, Eißendorfer Straße 40 (N), 21073 Hamburg, Hamburg, Germany

Published online: 13 June 2026

damage cases. Fairclough et al. (2023) replaced member-based damage with a patch-based model and introduced an adaptive strategy to include critical damage scenarios. Other studies have extended fail-safe topology optimization to different applications, such as fiber-reinforced composites (Cheng et al. 2025) and metamaterials (Zheng et al. 2025). For a comprehensive overview of methods aimed at improving computational efficiency, see the review by Whiteside et al. (2025). Although great progress has been made, fail-safe topology optimization remains computationally intensive, hindering progress in research and industrial application.

In recent years, a growing number of researchers have explored machine learning approaches in structural optimization for a range of objectives (Ramu et al. 2022; Woldseth et al. 2022; Shin et al. 2023). Extending these methods to fail-safe topology optimization offers a promising route for further gains in computational efficiency (Whiteside et al. 2025). Shin et al. (2023) divided the purpose of the application of machine learning in topology optimization into seven categories. With *non-iterative Optimization*, the optimal topology is predicted based on conditions, describing the optimization problem, without the need for expensive iterative optimization (Cang et al. 2019; Rade et al. 2020). *Acceleration of Iteration* aims to speed up conventional optimization by mapping the intermediate to the finished design (Banga et al. 2018; Kallioras et al. 2020). Through *Meta Modeling*, the computational cost is reduced by replacing, e.g., finite element analyses and/or sensitivity analysis using a machine learning algorithm (Lee et al. 2020; Chi et al. 2021). With *Dimensionality Reduction of Design Space*, machine learning models are used for reparameterization to reduce the number of design variables and thereby, computational cost (Li et al. 2019). With *Improvement of Optimizer*, either optimal parameters for a conventional optimizer are obtained via machine learning (Jiang et al. 2020) or replaced entirely by reparameterization and backpropagation (Chandrasekhar and Suresh 2021). In *Generative Design*, generative models are used to predict several solutions for the same optimization problem, offering the user a variety of designs to choose from (Yamasaki et al. 2021; Sim et al. 2021). With *Post-processing*, designs obtained by conventional optimizations are modified to ensure manufacturability (Napier et al. 2019; Xue et al. 2021).

The current paper proposes a new method to determine a fail-safe design by topology optimization, which is illustrated in Fig. 1. The idea is to first run a plain topology optimization for compliance minimization without fail-safety considerations. The result is then converted into a fail-safe design using an artificial neural network (ANN). Therefore, the approach falls in the category *Post-processing*, though the target is not to ensure manufacturability, but fail-safety. The primary objective of the proposed framework is to approximate fail-safe topology optimization solutions produced by the approach of Jansen et al. (2013) with comparable performance at reduced computational cost.

This task can be considered as an image-to-image translation problem, a class of tasks in which artificial neural networks typically perform well (Isola et al. 2017). However, as we show in the paper, pure image-to-image translation is not sufficient in order to come up with high-performing designs. Therefore, a post-processing step is required after the image-to-image translation. For training the ANN, a large dataset is generated by running plain topology optimization and topology optimization considering fail-safety for various combinations of boundary conditions, load cases, volume fractions, and mesh resolutions.

This paper is organized as follows. First, explicit and implicit considerations of fail-safety in topology optimization are recapitulated in Sect. 2. Section 3 provides the relevant foundations of deep learning. In Sect. 4, the proposed approach is elaborated, including training dataset creation, model and hyperparameter selection, and the developed post-processing steps. Sect. 5 presents the training results and discusses the potential and limitations of the proposed approach based on several examples. A final conclusion is given in Sect. 6.

2 Topology optimization for damage tolerance

Damage tolerance arises from redundant load paths that allow load redistribution after structural member failure. Fail-safe designs can be achieved either by explicitly including failure cases in the optimization or implicitly by enforcing structural redundancy. Publications already exist on both approaches, which are summarized in this section.

2.1 Wording

As already noted by Whiteside et al. (2025), the terminology for fail safety, damage tolerance or robustness is not consistent throughout the literature on this topic.

Sun et al. (1976) consider the exceedance of a stress limit, and hence a measure for failure, in the constraint of their optimization problem. Hence, structural integrity is ensured even in the case that one structural member failed. In the authors' opinion, this very much aligns with fail-safety. Jansen et al. (2013), Zhou and Fleury (2016), and Ambrozkiewicz and Kriegesmann (2020) consider the worst-case compliance as optimization objective. The resulting structures are hence as stiff as possible in the case of a removed structural member, but no statement is made about failure. Jansen et al. (2013) and Zhou and Fleury (2016) also speak of robustness of the structure. As Whiteside et al. (2025) points out, "the term

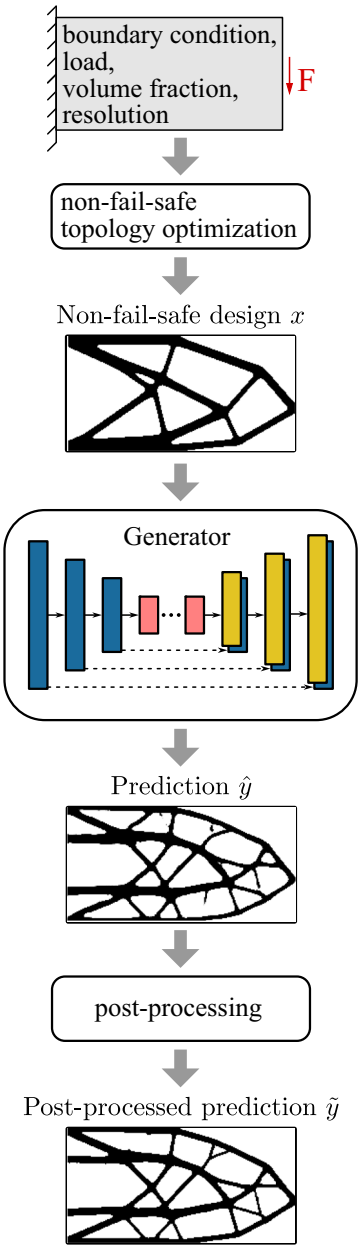

Fig. 1 Approach of fail-safe design by machine learning

'robustness' in the context of optimization is commonly used to describe a type of optimization problem where the design parameters possess some degree of uncertainty and ensuring the resulting structure is insensitive to such uncertainties." Whiteside et al. suggest the term 'structural robustness' in order to differentiate from robustness against uncertainty. In the authors view, the better option is the term 'damage tolerance', which was already used by Arora et al. (1980). Damage tolerance is a requirement for safety-critical structures that can be demonstrated by proving fail-safety, but also by other means. Hence, it appears to us as a more general term, which will therefore be used throughout this paper.

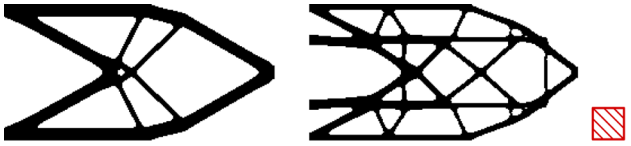

Fig. 2 Explicit fail-safe design (right) compared to non-fail-safe design (left). The applied damage patch is indicated by the red square

2.2 Explicit consideration of failure cases

In the approach proposed by Jansen et al. (2013), fail-safety is achieved by modeling partial local failure by removing patches of material in every possible location in each iteration. The resulting optimization problem reads

$$\begin{aligned} \min_{\boldsymbol{\rho}} \max_{j} \quad & c_j \\ s.t. \quad & \mathbf{K}_j \mathbf{u}_j = \mathbf{f} \quad \forall \quad j = 0, ..., N_{FC} \\ & 0 \leq \rho_e \leq 1 \quad \forall \quad e = 1, ..., N_E \\ & V(\boldsymbol{\rho}) = V_0 \end{aligned} \tag{1}$$

with c_j being the compliance of all N_{FC} failure cases. Using the SIMP approach, each finite element is assigned one pseudo density ρ_e, which is constrained to be in the interval [0, 1]. The design vector $\boldsymbol{\rho}$ hence contains the N_E pseudo densities ρ_e. The volume fraction of the design $V(\boldsymbol{\rho})$ is restricted to the prescribed volume fraction V_0. The equilibrium system $\mathbf{K}_j \mathbf{u}_j = \mathbf{f}$ is solved for each failure case in N_{FC}, resulting in extreme computational cost. Since the maximum operator is non-differentiable, the worst-case (i.e. maximal) compliance is commonly approximated using the p-norm. As shown in Fig. 2, the resulting design exhibits redundant load paths, thereby enhancing its damage tolerance. Here, we use the modified SIMP approach (Bendsøe 1989; Sigmund 2007) with a variable filter (Bruns and Tortorelli 2001).

2.3 Implicit enforcement of redundancy

Instead of explicitly considering failure cases, damage tolerance can be improved implicitly by enforcing structural redundancy. Wu et al. (2018) introduced a local volume constraint to optimize the infill for additive manufacturing. The optimization problem reads

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & c \\ s.t. \quad & \mathbf{K}\mathbf{u} = \mathbf{f} \\ & 0 \leq \rho_e \leq 1 \quad \forall \quad e = 1, ..., N_E \\ & \max_{e}(\bar{\rho}_e) \leq \alpha \end{aligned} \tag{2}$$

where $\bar{\rho}_e$ is the local volume in the neighborhood of the element e specified by radius r. The maximum local vol-

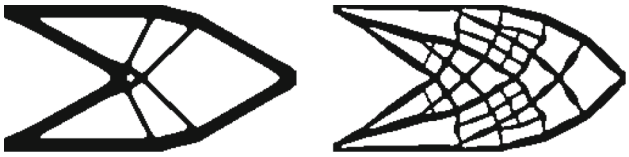

Fig. 3 Implicit fail-safe design (right) compared to non-fail-safe design (left)

ume is restricted to α, preventing the formation of thick structural members and enforcing designs with a larger number of thinner members, as illustrated in Fig. 3. Compared to plain compliance minimization, only one additional constraint needs to be evaluated per iteration; thus, this redundancy comes with little additional computational cost. Kranz et al. (2021) compared the explicit and implicit formulations for fail-safe stress optimization and found that the explicit approach consistently outperforms the implicit one in terms of damage tolerance. Moreover, the implicit consideration of failure requires substantial heuristic parameter tuning in order to achieve satisfying damage tolerance.

2.4 Evaluation of fail-safety

Jansen et al. (2013) use a square shape damage patch for the approach recapitulated in Sect. 2.2. This is an obvious choice for a topology optimization with a voxel mesh, but it is still an arbitrary choice. Kranz et al. (2021) compare different damage patch shapes in the context of fail-safe stress minimization. While the influence of the patch shape turned out to be surprisingly small, the size of the damage patch has a strong influence on the resulting topology. In order to allow for a fair comparison between different approaches, a fail-safety evaluation procedure was suggested by Kranz et al. The general idea is to identify load paths in the optimized topology using image processing techniques and to consider their removal as damage scenarios (Fig. 4). For this purpose, the density field is skeletonized, and structural nodes are identified as branch points. Removing these branch points yields individual structural struts. By clustering these struts, their midpoint, major and minor axes, and orientation can be determined. The damage size is determined by increasing it until a specified number of void elements are included. Damage scenarios located within a certain range of the load application points are excluded. Thereby, the measure for fail-safety is closer to what is considered as fail-safety in safety-critical applications.

Both the patch-based and the load path-based fail-safety evaluation are used for the evaluation of results in this paper.

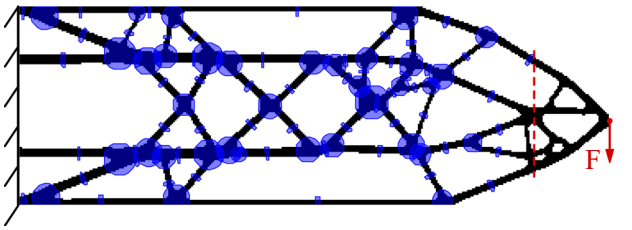

Fig. 4 Load-path based damage modeling, adapted from Kranz et al. (2021). Structural members to the right of the red dashed line are not considered

3 Deep learning foundations

The deep learning model used in this paper is a conditional generative adversarial network. This section elaborates on the model architecture and outlines the relevant deep learning foundations.

3.1 Conditional generative adversarial networks

The generative adversarial network (GAN), proposed by Goodfellow et al. (2014), is a framework to train generative models only using backpropagation. The framework consists of two models that are trained simultaneously. The generator is trained to produce samples from noise that match the data distribution, while the discriminator learns to estimate the probability that a sample originates from the data distribution rather than the generator. The training process can be viewed as a two-player game in which the generator attempts to fool the discriminator, while the discriminator seeks to expose the generator.

With the base GAN framework, there is no control over the output of the generator. Mirza and Osindero (2014) added conditional information to the noise input, allowing control over the output of the generator. This extension is referred to as the conditional generative adversarial network (cGAN).

In recent years, the cGAN framework has been frequently applied in the field of topology optimization, demonstrating reliable and strong performance (Behzadi and Ilie 2021; Nie et al. 2021; Huang et al. 2025; Parrott et al. 2023; Rawat and Shen 2019; Pereira and Driemeier 2024). Accordingly, a cGAN architecture is adopted as the model framework for the proposed approach.

3.2 The U-SE-ResNet architecture

The U-Net is a convolutional neural network introduced by Ronneberger et al. (2015) for biomedical image segmentation. Its architecture, illustrated in Fig. 5, consists of a contracting path for feature extraction and a symmetric expanding path that maps these features to the output domain. The connection between the two paths forms the bottleneck.

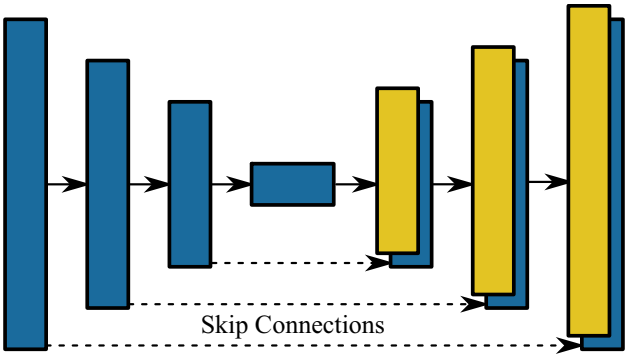

Fig. 5 U-Net architecture

Skip connections link corresponding levels of the contracting and expanding paths to preserve spatial information lost during downsampling and to recover high-resolution details. The U-Net architecture has since been widely adopted for topology optimization (Zheng et al. 2021; Qiu et al. 2021; Seo and Kapania 2023; Wang et al. 2023).

As neural networks deepen, training becomes more challenging due to vanishing gradients (Hochreiter 1991). To address this, He et al. (2016) proposed the residual learning framework, in which the model learns residual mappings rather than direct transformations.

Traditional convolutional networks treat all feature channels equally, ignoring potential inter-channel dependencies. To enhance representational capacity, Hu et al. (2018) introduced the squeeze-and-excitation (SE) block, which adaptively recalibrates channel-wise feature responses by modeling their interdependencies. A global average pooling operation first "squeezes" global spatial information into channel descriptors, which are then used to "excite" channels through learned weighting. Hu et al. incorporated the SE-block into a ResNet by adding it before the summation with the identity branch, as depicted in Fig. 6, and named it SE-ResNet.

For non-iterative optimization, Nie et al. (2021) proposed the U-SE-ResNet, a U-Net variant in which the bottleneck is replaced with SE-ResNet blocks, combining spatial precision with improved feature representation.

Given the promising performance of the U-SE-ResNet, it is adopted as the baseline generator architecture within the cGAN framework in the present work. Modifications to this architecture are introduced and discussed in Sect. 4.2.

4 New approach for fail-safe design

The implementation of the proposed approach, illustrated in Fig. 1, involves the following steps. A training dataset is generated, consisting of paired non-fail-safe (x) and fail-safe designs (y). For this, various combinations of boundary conditions, loads, volume fractions, and mesh resolutions are considered. Within a cGAN framework (Fig. 7), the generator G is trained to map each non-fail-safe design x, together with random noise z, to its corresponding fail-safe counterpart y. The generator output $\hat{y}$ is constrained to the interval $[0, 1]$ via a sigmoid activation function, representing a pseudo density field. The design predicted by the generator is then passed, along with the non-fail-safe design, to the discriminator D, which outputs the probability that the input sample corresponds to a real fail-safe design rather than a generated one. The generator loss is computed using the discriminator's response together with additional metrics, which are discussed in Sect. 4.3.

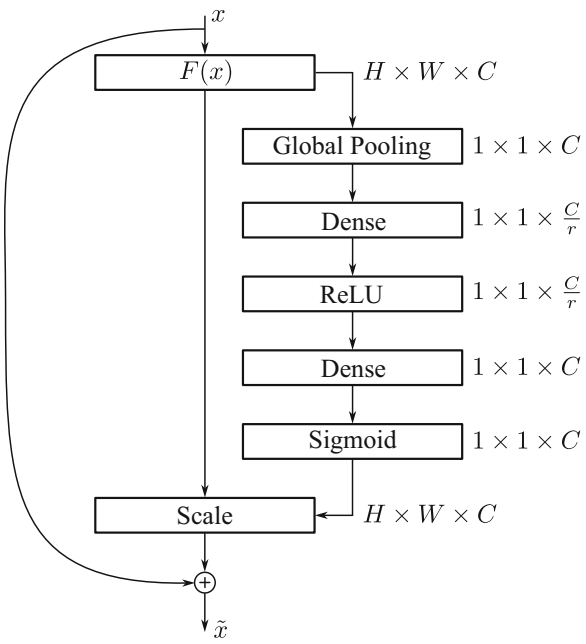

Fig. 6 SE-ResNet architecture of an artificial neural network suggested in Hu et al. (2018)

Given the limitations reported in literature (Woldseth et al. 2022), the predictions of the trained model are expected to exhibit weaknesses such as structural disconnections. To address these issues, a post-processing step is applied, which is presented in Sect. 4.4.

The motivation behind fail-safe design is that, if one load path fails, the applied load can be redistributed through the remaining load paths. In topology optimization, however, neither the final design nor its load paths are known in advance. Therefore, a natural choice for modeling damage is the use of predefined damage patches. Ambrozkiewicz and Kriegesmann (2018) proposed a damage modeling approach that more closely follows the original motivation of fail-safe design by identifying actual load paths and continuously deriving as well as updating damage scenarios based on them throughout the optimization process. This reduces the number of damage scenarios that need to be evaluated signif-

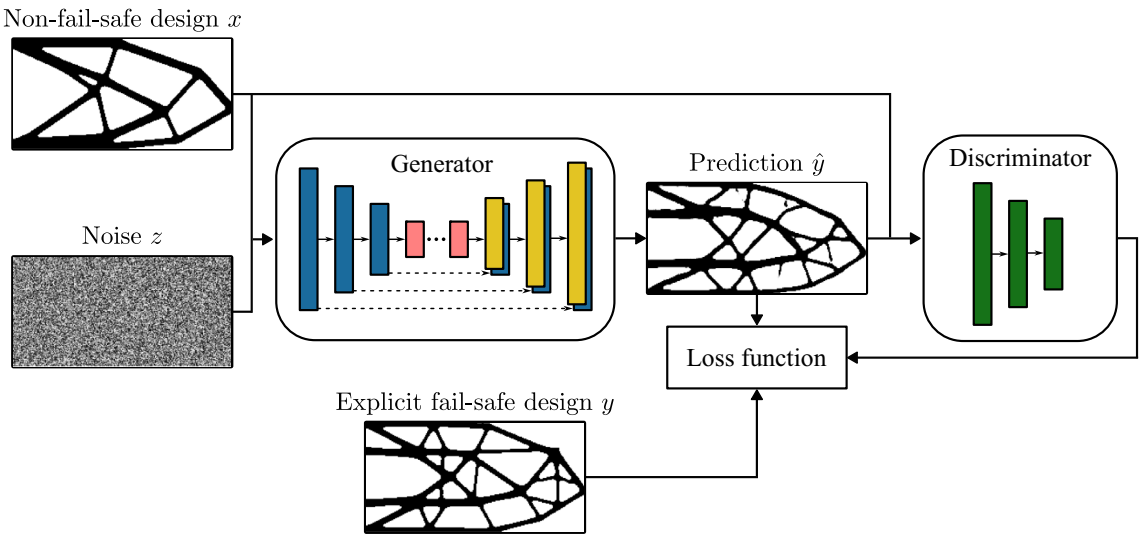

Fig. 7 Generator training scheme

icantly. However, because the optimization problem changes continuously, convergence is not likely, which limits the direct applicability of the method in topology optimization. Consequently, the approach was applied in combination with shape optimization (Ambrozkiewicz and Kriegesmann 2020).

The cGAN framework is trained to reproduce explicit fail-safe topology optimization solutions. Thus, its primary objective is to achieve comparable performance with respect to the patch-based criterion at reduced computational cost. The path-based criterion is additionally evaluated to provide a more comprehensive perspective and to better assess the damage tolerance of the designs produced by the proposed approach, both individually and in comparison with alternative approaches such as the implicit formulation.

The framework is evaluated on the test dataset. To evaluate the performance of the proposed approach, the predicted fail-safe designs are compared to the ground truth design in terms of volume fraction (VF), undamaged compliance C_{und}, damage-patch-based worst case compliance $C_{\text{wc}}^{\text{patch}}$, and load-path-based worst case compliance $C_{\text{wc}}^{\text{path}}$. Here, the term "worst case" refers to the maximum compliance value out of all considered damage scenarios. To determine $C_{\text{wc}}^{\text{patch}}$, the square damage patches, which were also used for the data generation via the approach of Jansen et al. (2013). For $C_{\text{wc}}^{\text{path}}$, the damage scenarios are determined via image processing, identifying individual load-paths and their intersections (Kranz et al. 2021). In each damage scenario, one branch or junction is removed.

4.1 Training data generation

For the supervised training of the cGAN framework, a large dataset is required. As illustrated in Fig. 8, each data sample consists of a non-fail-safe topology-optimized design and a fail-safe design. The non-fail-safe design is obtained using plain topology optimization for minimal compliance. Considering the same boundary conditions, the approach of Jansen et al. (2013) is used to obtain the fail-safe design.

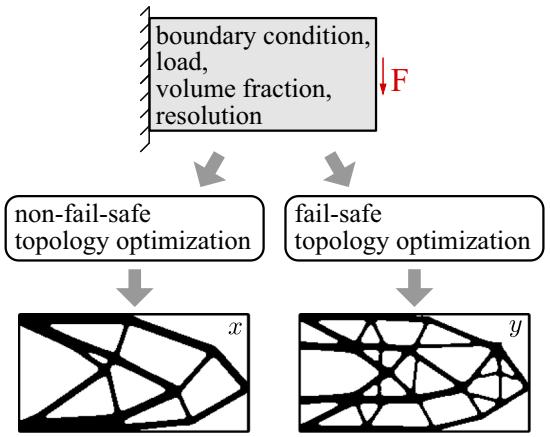

Fig. 8 Generation of training data

For both non-fail-safe and fail-safe topology optimization, the modified SIMP method (Bendsøe 1989; Sigmund 2007) is used with a penalty factor of 3. To suppress checkerboarding and reduce mesh dependence, a density filter (Bourdin 2001; Bruns and Tortorelli 2001) is applied with a radius of 6. In addition, a projection scheme (Xu et al. 2010; Wang et al. 2011) with continuation is used to improve convergence. The projection parameter β is initialized to 1 and doubled every 30 iterations for the non-fail-safe optimization and every 50 iterations for the fail-safe optimization, until a maximum value of 16 is reached.

The setup for damage modeling in the fail-safe topology optimization is illustrated in Fig. 9 for the cantilever example.

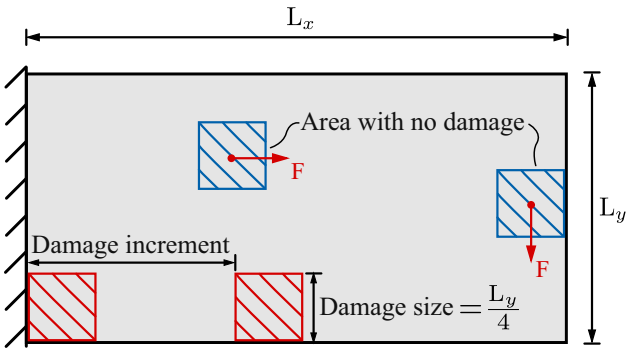

Fig. 9 Exemplary illustration of failure case size and increment, adapted from Kranz et al. (2021)

A damage patch size of $\frac{L_y}{4}$ is used for all samples. Damage patches located at or near load-application points are deactivated. To reduce the computational cost of dataset generation, the patch increment is chosen equal to the patch size, which limits the number of damage scenarios to be evaluated while still producing designs with strong fail-safe performance (Zhou and Fleury 2016). Moreover, the non-fail-safe and fail-safe optimizations are not run to full convergence; instead, they are terminated after 150 and 300 iterations, respectively. In rare cases, this early stopping yields designs of insufficient quality, which are subsequently removed from the dataset. The resulting grayscale designs are projected to strictly binary representations of void and material. Due to the early stopping procedure, both the non-fail-safe and, in particular, the fail-safe designs contain regions of intermediate density. Since the fail-safe designs generally exhibit a larger portion of such intermediate densities, slight differences in volume fraction between non-fail-safe and fail-safe designs can occur. It is assumed that these deviations do not significantly affect the training process or the results of the proposed framework.

The dataset is generated using four fixed boundary conditions: the cantilever, the MBB beam, the L-beam, and the diagonal support configuration. These setups are illustrated in Fig. 10.

To create deviating designs, 1–3 independent load cases are considered per sample. Across the dataset, $\frac{1}{6}$, $\frac{1}{3}$, and $\frac{1}{2}$ of the samples are generated with one, two, and three load cases, respectively. Each load case consists of a single point load, which is placed randomly in the design domain. All loads have the same magnitude and act either in horizontal or vertical direction, which is decided randomly. For the volume fraction constraint, upper limits of 0.20, 0.30, 0.40, and 0.50 are considered.

The design domain dimensions are sampled uniformly from discrete values, with $L_x \in \{120, 240, 360, 480, 600, 720\}$ and $L_y \in \{120, 240\}$. Woldseth et al. (2022) observed that most related studies employ comparatively low mesh resolutions, typically not exceeding 26,000 elements. With up to 92,160 elements (for a maximum aspect ratio of 1:6 of the design space), the mesh resolution used in this study is still low compared to state-of-the-art topology optimization. One reason is that determining the ground truth fail-safe design using the approach of Jansen et al. (2013) is extremely expensive compared to standard topology optimizations.

To enable training with designs of varying mesh resolutions, similar to Zheng et al. (2021), all designs are embedded into a zero-padded matrix large enough to contain the largest design in the dataset. The resulting unified representation has the dimensions 240×720 elements. To enhance generalization, each design is positioned at random coordinates within this matrix, as illustrated in Fig. 11.

Combining all variations in boundary conditions, volume fractions, mesh resolutions, and randomized load cases yields an initial training dataset of 12,417 designs. Applying 180° rotations as well as horizontal flips augments the dataset to a total of 49,668 samples. The dataset is divided into ten equally sized subsets. One subset is not used during model training but as an independent test set. The remaining nine subsets are used for 9-fold cross-validation. In each fold, eight subsets are used for training and one for validation. After training, the model is evaluated on the fixed test set.

4.2 Model and training

The cGAN framework used in the present work is implemented independently. The network architecture of the generator is based on the U-SE-ResNet proposed by Nie et al. (2021), while the discriminator's architecture is based on the PatchGAN architecture originally introduced by Isola et al. (2017) and adopted by Nie et al. (2021). Both networks are not used as-is but are modified to meet the requirements of the present approach. The generator is a U-Net with SE-ResNet blocks as the bottleneck. The contracting path contains three convolutional blocks, each consisting of a convolutional layer followed by batch normalization and activation function. The leaky-ReLU activation used by Nie et al. (2021) is replaced with the swish activation, which has shown improved performance in ResNet-type architectures (Ramachandran et al. 2017). The expanding path mirrors the contracting path. To mitigate checkerboarding artifacts, transposed convolutions are replaced by nearest-neighbor upsampling followed by convolution (Odena et al. 2016). The bottleneck comprises 32 SE-ResNet blocks (Hu et al. 2018).

While the conditional input used by Nie et al. (2021) consists of six channels, the present approach requires only one. Accordingly, the number of kernels of the first convolutional layer is reduced from 128 to 8; these are then doubled with each subsequent convolutional layer in the contracting path. Increasing the number of kernels was tested and resulted in improved training metrics. However, this improvement did

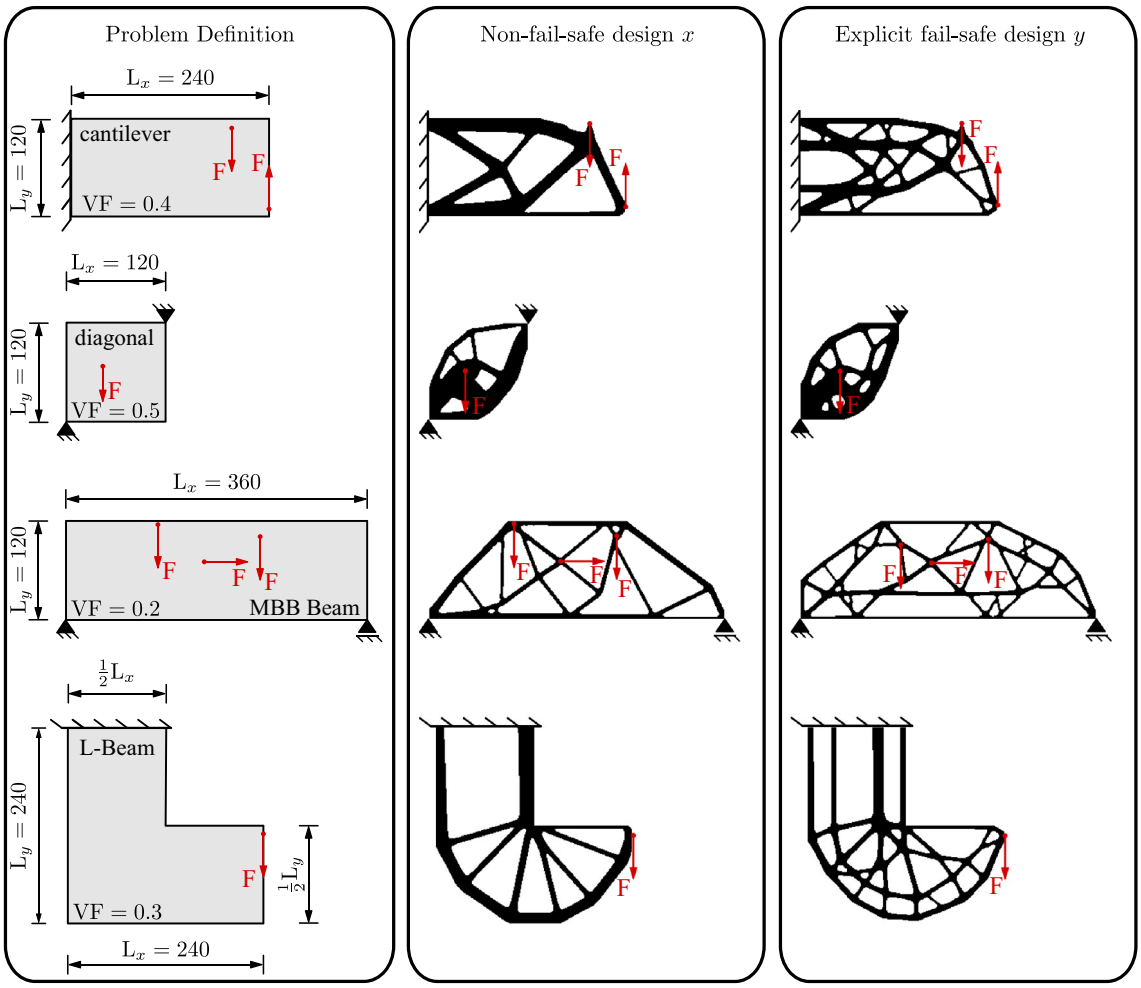

Fig. 10 Examples for training data

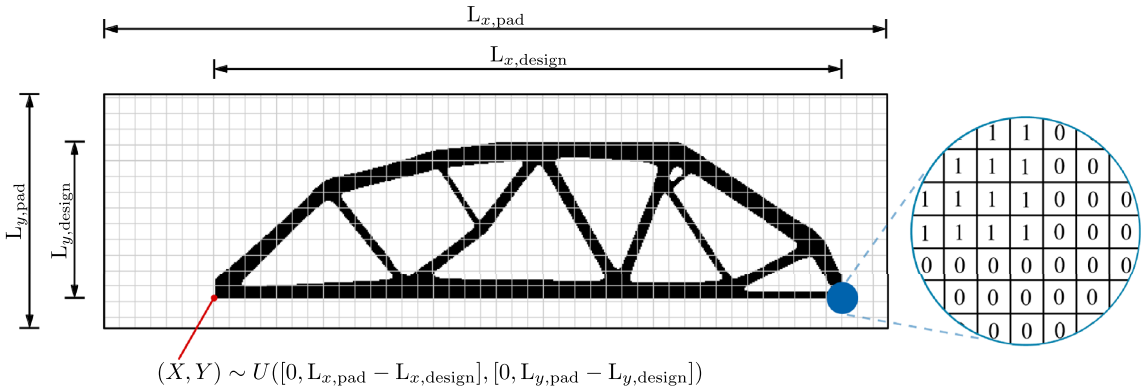

Fig. 11 Design placing in enclosing zero matrix

not translate into enhanced mechanical performance of the predicted designs, as discussed in Appendix A. Nie et al. (2021) considered designs of size 64 × 128, while the zero-padded inputs in this paper have a size of 240 × 720. All convolutional kernels are enlarged from 3×3 to 9×9, to better capture long-range spatial dependencies. The improvement produced by this modification is reported in Appendix A. The resulting generator features 23,013,921 trainable parameters.

The discriminator, based on the PatchGAN architecture of Isola et al. (2017), is modified analogously: the number of kernels in the first layer is reduced from 32 to 16, and the convolutional kernel size is increased from 3×3 to 9×9. The final

discriminator features 961,041 trainable parameters. The initial learning rates for the generator and discriminator are set to 0.001. Both are trained using the Adam optimizer (Kingma and Ba 2017) with a batch size of 64. All model and training hyperparameters were selected manually without the use of automated hyperparameter optimization.

4.3 Loss function

The loss function $\mathcal{L}_G$ of the generator G is an aggregation of four loss terms, given by

$$\mathcal{L}_G = \lambda_{\text{Adv}}\mathcal{L}_{\text{Adv}} + \lambda_{\text{wMSE}}\mathcal{L}_{\text{wMSE}} + \lambda_{\text{VF}}\mathcal{L}_{\text{VF}} + \lambda_{\text{ID}}\mathcal{L}_{\text{ID}} \quad (3)$$

The weight factors are manually chosen to $\lambda_{\text{Adv}} = 0.01$, $\lambda_{\text{wMSE}} = 1.0$, $\lambda_{\text{ID}} = 0.2$ and $\lambda_{\text{VF}} = 10$. Each term promotes a different aspect of the desired prediction, which is described below. Using the discriminator prediction D, the adversarial loss $\mathcal{L}_{\text{Adv}}$ forms the core objective,

$$\mathcal{L}_{\text{Adv}} = \mathbb{E}_{(x,y)\sim p_{\text{data}},\, z\sim p_z}\left[-\log D(x, G(x,z))\right] \quad (4)$$

where x denotes the conditional input, z the random noise, and $\mathbb{E}$ the expectation operator, which is approximated during training by the average over each mini-batch.

To encourage predictions that more closely match the ground truth and reduce compliance error, a weighted mean squared error term $\mathcal{L}_{\text{wMSE}}$ is added:

$$\mathcal{L}_{\text{wMSE}} = \mathbb{E}_{(x,y)\sim p_{\text{data}},\, z\sim p_z}\left[\frac{1}{N_e}\sum_{e=1}^{N_e}(y_e - \hat{y}_e)^2 \, (1 + y_e(\lambda_{\text{mat}} - 1))\right] \quad (5)$$

Here, y is the ground-truth fail-safe design and $\hat{y} = G(x,z)$ the generator prediction. The squared error is computed per element e and averaged over all $N_e = 172,800$ elements. The weighting factor λ_{mat} amplifies the contribution of elements that contain material in y, thereby reducing the penalty for falsely predicted material and encouraging the formation of structural members. This shifts emphasis from void regions, which are abundant due to the zero-padding approach, toward the material phase.

Because this weighting tends to increase the predicted volume fraction, an additional term is introduced to control volume:

$$\mathcal{L}_{\text{VF}} = \mathbb{E}_{x\sim p_{\text{data}}(x),\, z\sim p_z}\left[\left|\sum_{e=1}^{N_e}\hat{y}_e - \sum_{e=1}^{N_e}x_e\right|\right] \quad (6)$$

The volume error is computed relative to the conditional input x rather than to y to account for small volume discrepancies between paired samples. A sigmoid activation at the generator output bounds predictions between zero and one.

To further discourage intermediate densities and promote crisp black-and-white designs, the intermediate-density penalty $\mathcal{L}_{\text{ID}}$ is applied:

$$\mathcal{L}_{\text{ID}} = \mathbb{E}_{x\sim p_{\text{data}}(x),\, z\sim p_z}\left[\frac{1}{N_e}\sum_{e=1}^{N_e}4\hat{y}_e(1-\hat{y}_e)\right] \quad (7)$$

An additional loss term penalizing the difference in undamaged compliance between the predicted and ground truth design was explored. However, computing the undamaged compliance via finite element analysis for all samples within the training loop was found to be computationally prohibitive, resulting in a substantial increase in training time.

The discriminator is trained to distinguish real fail-safe designs from generator-produced predictions conditioned on x. Its loss is the standard binary cross-entropy,

$$\mathcal{L}_D = -\mathbb{E}_{(x,y)\sim p_{\text{data}}}\left[\log D(x,y)\right] - \mathbb{E}_{x\sim p_{\text{data}},\, z\sim p_z}\left[\log\left(1 - D(x, G(x,z))\right)\right]. \quad (8)$$

To stabilize training, one-sided label smoothing is applied Salimans et al. (2016). The real and fake labels of 1 and 0 are replaced with 0.9 and 0.1, reducing discriminator overconfidence and tempering its feedback to the generator.

4.4 Post-processing to ensure connectivity

Multiple references report that designs generated by ANNs exhibit structural flaws, such as disconnected structural members and unsupported loads (see Fig. 12, top). While there are several articles describing these limitations (Patel et al. 2022; Behzadi et al. 2024), there is yet no definitive solution. Woldseth et al. (2022) discussed these limitations and their probable cause.

For the predicted fail-safe designs, similar problems are encountered. Therefore, two possible postprocessing procedures are presented that improve the performance loss due to the missing connectivity of the topology:

1. A purely graphics-based method that extends disconnected areas until connectivity is reached.
2. A fast topology optimization using a modified local volume constraint approach.

Both approaches are presented in the following subsections.

4.4.1 Post-processing by headland extension

This image-based post-processing approach, illustrated in Fig. 12, starts with up-scaling and smoothing the predicted

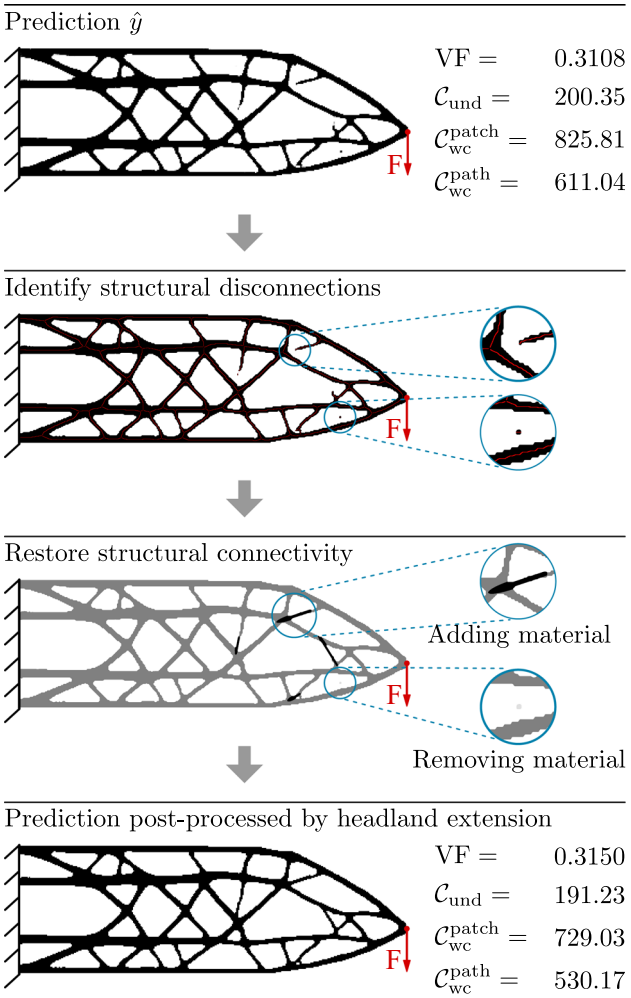

Fig. 12 Post-processing via headland extension

design. Applying skeletonization, disconnected members are identified, and their areas, minor and major axis lengths, orientations, and endpoints are determined. Disconnected structural members below a specified area threshold are removed. Members exceeding this threshold are reconnected by extending them in their determined orientation until material is reached. The transition between added and original material is smoothed, and the design is downscaled to the original resolution. As shown in Fig. 12, the image-based post-processing not only improves the design visually but also its mechanical performance.

In some cases, however, load introduction areas and/or boundary conditions are disconnected from the structure in the ANN prediction. For these cases, this image-based post-processing is not successful.

4.4.2 Post-processing by local volume constraints

Given the computationally extremely expensive nature of fail-safe topology optimizations, running a plain topology optimization as a post-processing does not compromise the computational gain of the approach. However, allowing a topology optimization to redistribute all material without considering fail-safety explicitly will provide a non-fail-safe design, even if the prediction of the ANN is used as a start vector. The idea is to mostly maintain the topology of the prediction and achieve improved compliance by redistributing as little material as possible. To reach this objective, the local volume constraint introduced by Wu et al. (2018) is modified such that the current local volume ρ_r does not fall below its initial value $\rho_{r,\text{init}}$. Enforcing a local volume constraint for every element, as proposed by Wu et al. (2018), would prevent any design changes once the global volume fraction limit is reached, since any material redistribution would immediately violate a constraint. Instead, the local volume is constrained only within a limited number of non-overlapping regions. This allows the optimizer to rearrange material within these regions without violating the imposed constraints. In general, these regions may be defined with arbitrary shape and size. For the post-optimization of the predicted fail-safe designs, square regions with dimensions equal to the damage patch size used during data generation were found to provide consistently good results. The constraint is given by:

$$\bar{\rho}_r \geq \bar{\rho}_{r,\text{init}} \quad \forall \quad r = 1, ..., N_r \tag{9}$$

with $\bar{\rho}_r$ being the local volume in the neighborhood of all N_r regions.

Instead of introducing a large number of constraints, the maximum constraint violation is considered.

$$\max \left(\frac{\bar{\rho}_{r,\text{init}}}{\bar{\rho}_r} \right) \leq 1 \tag{10}$$

To use gradient-based optimization, the maximum operator is approximated using the general "p-like" aggregation function A_g proposed by Ambrozkiewicz (2022). The choice of aggregation function and the corresponding parameters are described in Appendix B. Using the modified local volume constraint, the optimization problem becomes:

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & c \\ s.t. \quad & \mathbf{K}\mathbf{u} = \mathbf{f} \\ & 0 \leq \rho_e \leq 1 \quad \forall \quad e = 1, ..., N_E \\ & \max_r \left(\frac{\bar{\rho}_{r,\text{init}}}{\bar{\rho}_r} \right) \leq 1 \quad \forall \quad r = 1, ..., N_r \end{aligned} \tag{11}$$

The example shown in Fig. 13 demonstrates how the proposed post-processing approach substantially improves the overall performance of the predicted design by establishing connectivity to the supports and load application points,

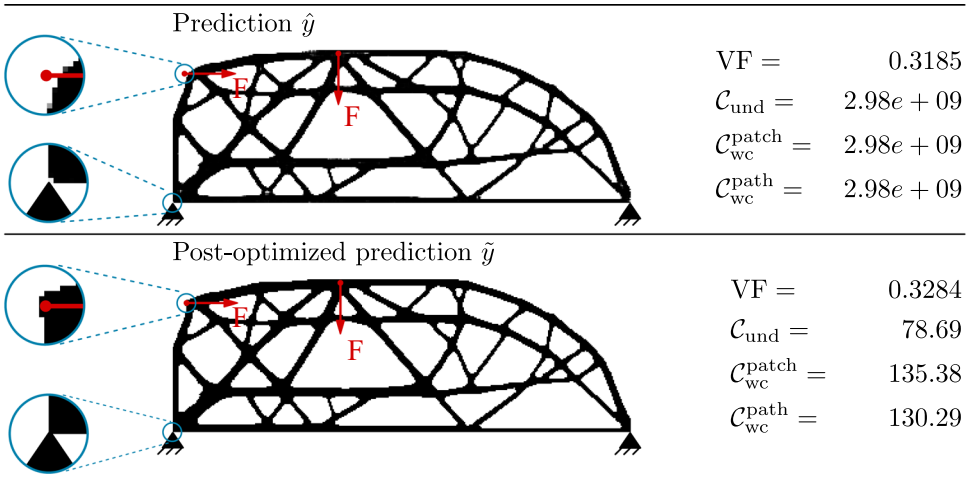

Fig. 13 Post-processing via topology optimization using a modified local volume constraint

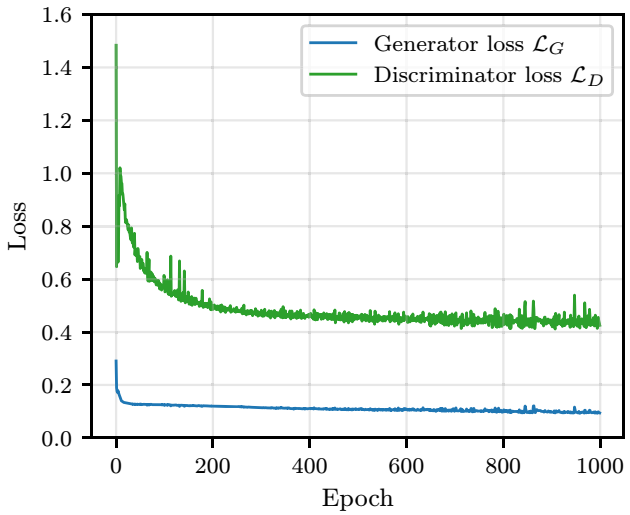

Fig. 14 Training loss of generator and discriminator throughout training

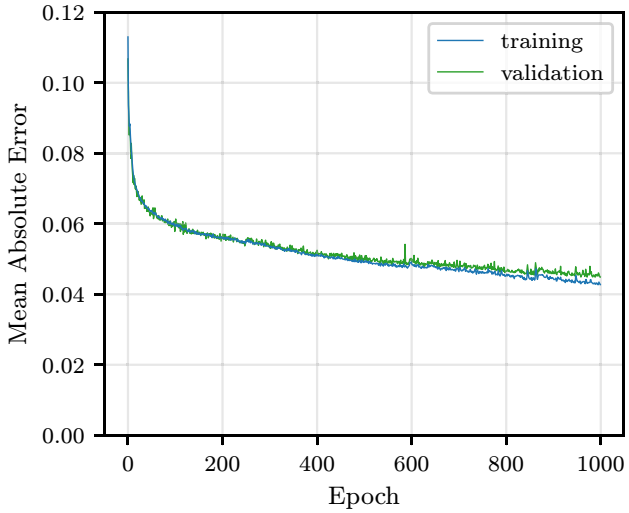

Fig. 15 Mean absolute error throughout training

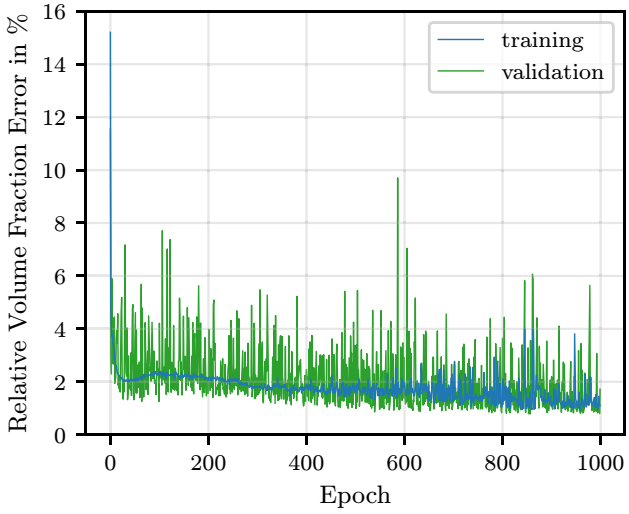

Fig. 16 Relative volume fraction error throughout training

while preserving structural redundancy. The computational cost of running this post-process optimization is similar to plain compliance minimization.

Since this approach allows for fixing both the disconnected beams as well as disconnected load introduction areas, it is used in the subsequent numerical examples.

5 Numerical results

The training of the cGAN was conducted on an RTX 5090 and took about 52 h across all cross-validation folds. The generator loss $\mathcal{L}_G$ and discriminator loss $\mathcal{L}_D$ of cross-validation fold 1 are illustrated in Fig. 14. Both the generator and the discriminator improve rapidly during the early stages of training, until both models converge with some oscillation in the later stages. Further training with a reduced learning rate did not result in significant additional improvements.

In addition, two metrics were tracked throughout training for both the training and validation datasets. The mean absolute error (MAE) between the predicted and ground-truth fail-safe design, and the relative volume fraction error (VFE) between the predicted and conditional non-fail-safe design. These are shown in Figs. 15 and 16, respectively. Similar to the loss terms, the MAE decreases rapidly in the early training stages, after which the rate of improvement diminishes. No meaningful difference between training and validation MAE is observed. Throughout training, the VFE exhibits pronounced oscillations, particularly on the validation dataset. As before, continued training does not lead to significant improvement. The final values of the loss terms and evaluation metrics for all cross-validation folds are reported in Appendix C.

The performance of the trained generator is evaluated using the test dataset, which is the 10-th fold held out in each cross-validation fold. The samples included in the test dataset are created using the same boundary conditions and volume fractions as for the training dataset. The test samples are unseen for the trained Generator in terms of load configuration. To enable a fair comparison (Woldseth et al. 2022), it is ensured that all designs consist only of pure material or void elements.

The designs predicted on the test dataset are further optimized using the modified local volume constraint presented in Sect. 4.4.2. On average, the relative error in volume fraction between the post-optimized prediction ($\tilde{y}$) and the ground truth explicit fail-safe design (y) is $+1.17\%$.

5.1 Comparison with ground truth

In order to evaluate the structural performance of designs originating from ANN and post-optimization $\tilde{y}$, their worst-case compliance C_{wc} is compared to the worst-case compliance of the corresponding design obtained by explicit consideration of fail-safety (ground truth y). For that, we consider the ratio of the worst-case compliance for both the patch-shaped failure cases (i.e. $\frac{C_{\text{wc}}^{\text{patch}}(y)}{C_{\text{wc}}^{\text{patch}}(\tilde{y})}$) and the removal of load paths as failure cases (i.e. $\frac{C_{\text{wc}}^{\text{path}}(y)}{C_{\text{wc}}^{\text{path}}(\tilde{y})}$). If this ratio is equal to one, then the fail-safety of the design suggested by the ANN is as good as the ground truth. Only in very rare cases, the predictions perform better than the ground truth. (This would lead to a ratio greater than one and is set to one for the graphs shown.)

Figure 17 (left) shows that, for most test cases, the ratio is significantly smaller than one. The median compliance increase of 85.32% indicates that the proposed method performs substantially worse than the ground truth fail-safe designs under the damage model used for training and reference generation. This represents an important limitation of the current framework. Such behavior is to be expected, as the ground truth designs are determined by explicit consideration of failure patches, and even a small deviation from the ideal topology can lead to a large increase in compliance. When considering the path-based criterion, however, many more designs reach the same fail-safety as the ground truth (see Fig. 17 (right)). With respect to this criterion, the median compliance increase is 8.91%.

5.2 Comparison with non-fail-safe design

To quantify the improvement in damage tolerance achieved by the post-optimized prediction $\tilde{y}$ relative to the conditional non-fail-safe design x, the ratios of worst-case compliance with respect to both criteria are evaluated (i.e. $\frac{C_{\text{wc}}^{\text{patch}}(\tilde{y})}{C_{\text{wc}}^{\text{patch}}(x)}$ and $\frac{C_{\text{wc}}^{\text{path}}(\tilde{y})}{C_{\text{wc}}^{\text{path}}(x)}$). A ratio close to zero indicates significant improvement in damage tolerance, whereas ratios greater than one indicate a degradation. In Fig. 18, the latter cases are highlighted by the red bins. Figure 18 shows that, for both criteria and in the majority of cases, these ratios are close to zero, demonstrating a significant improvement in damage tolerance. The median compliance reduction is 96.02% and 96.05% with respect to the patch- and path-based criteria, respectively.

Nevertheless, there remain samples for which the proposed framework fails, and the post-optimized predictions perform worse than the non-fail-safe designs. With respect to the patch-based criterion, 175 (3.55 %) samples are affected. In most of these cases, the post-optimized designs lack structural redundancy in critical regions, rendering them vulnerable to damage. Regarding the load-path-based criterion, 179 post-optimized predictions (3.65 %) exhibit inferior performance compared to the non-fail-safe designs. In addition, 247 ground truth designs (5.01 %) also perform worse than the non-fail-safe designs with respect to this criterion. In these instances, the non-fail-safe design displays unusually high damage tolerance, while the fail-safe design performs comparatively poorly. This behavior can be attributed either to damage being applied closer to the load application point than accounted for in the explicit optimization, or to the complete removal of structural members located between adjacent damage patches when damage is applied along load paths (see Fig. 19). Among the post-optimized predictions and ground truth designs that perform worse than the non-fail-safe design with respect to the load-path-based criterion, there is a substantial overlap of 98 samples. In these overlapping cases, both designs often share the same worst-case damage scenario, which is consistent with the model being trained to approximate the ground truth designs. In the remaining cases without overlap, the post-optimized predictions typically lack sufficient structural redundancy in critical

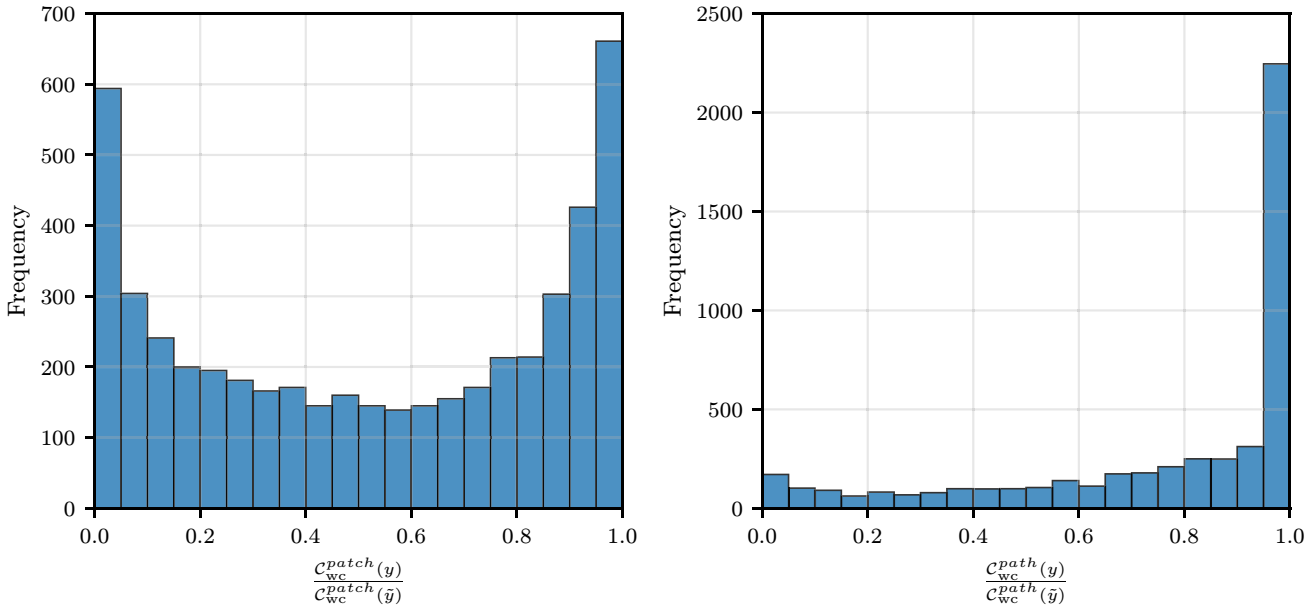

Fig. 17 Ratio of damage-patch-based (left) and load-path-based (right) worst case compliance between ground truth y and post-optimized prediction $\tilde{y}$ on the test dataset

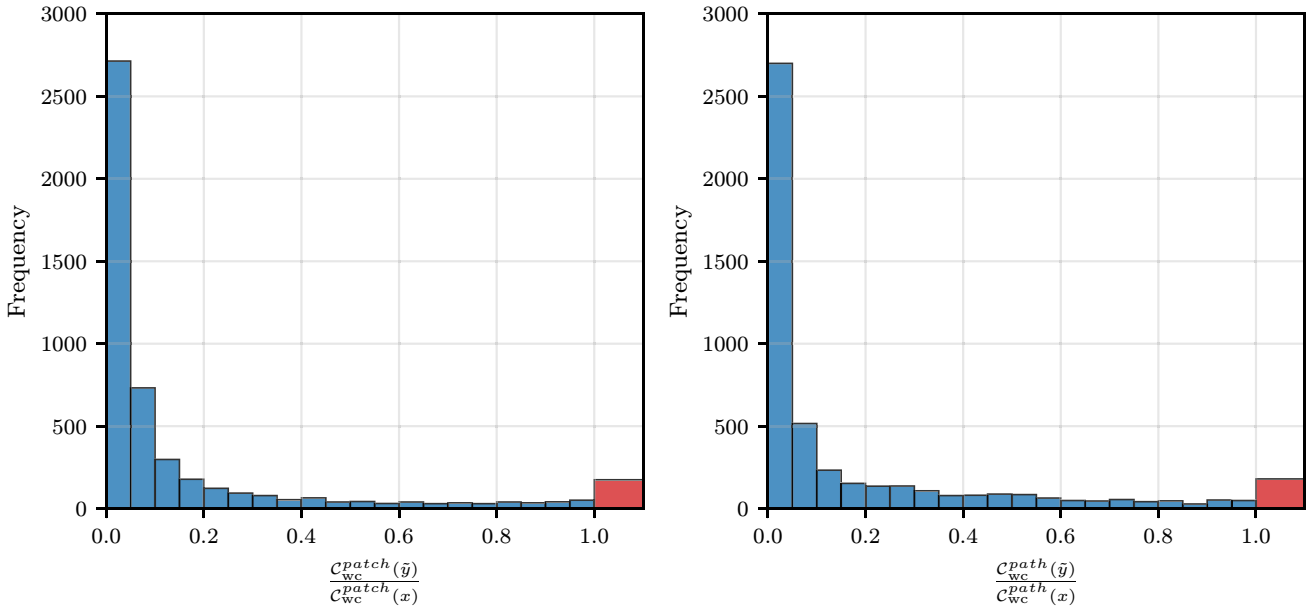

Fig. 18 Ratio of damage-patch-based (left) and load-path-based (right) worst case compliance between post-optimized prediction $\tilde{y}$ and non-fail-safe design x on the test dataset

regions and exhibit inferior undamaged performance compared to the non-fail-safe design.

5.3 Comparison with implicit enforcement of redundancy

In order to fully assess the performance of the designs obtained via the proposed approach, the post-optimized predictions are additionally compared with designs created via the computationally more affordable implicit enforcement of redundancy discussed in Sect. 2.3. As noted above, the implicit formulation requires substantial tuning of the heuristic parameters to achieve satisfactory damage tolerance. Despite extensive efforts, no single parameter set could be identified that performs well across all samples in the test dataset. Consequently, implicit fail-safe designs ($\bar{y}$) are created for only 44 test samples spanning boundary conditions, volume fraction, resolutions, and load configurations.

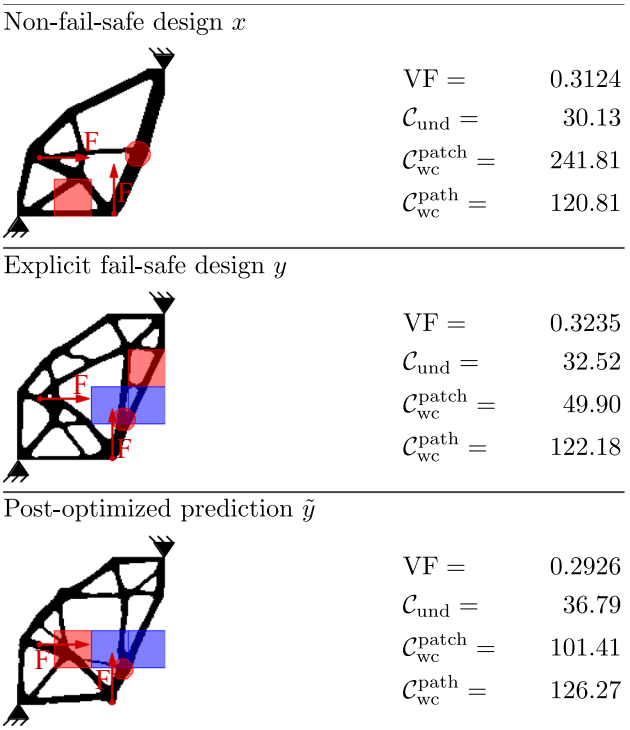

Fig. 19 Performance comparison of non-fail-safe design x, explicit fail-safe design y, post-optimized prediction $\tilde{y}$, where both y and $\tilde{y}$ perform worse than x with respect to the path-based criterion. Worst-case damage-patch and load-path damage are highlighted by a red square and circle, respectively. Adjacent damage patches that do not completely eliminate the affected load path are highlighted in blue

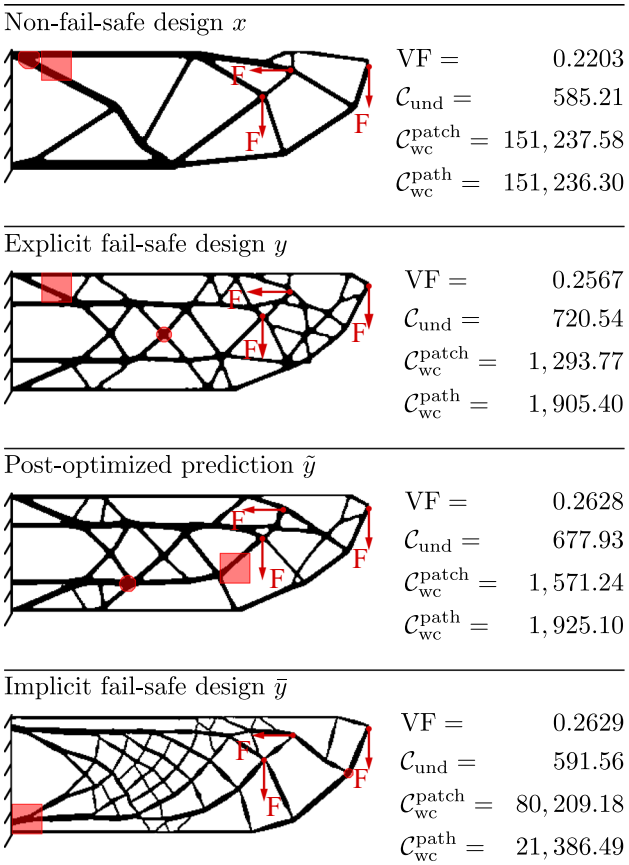

Fig. 20 Performance comparison of non-fail-safe design x, explicit fail-safe design y, post-optimized prediction $\tilde{y}$, and implicit fail-safe design $\bar{y}$ for test sample 1. Worst-case damage-patch and load-path damage are highlighted by a red square and circle, respectively

Adjusting the radius and volume threshold of the local volume constraint, both the number of structural members and the spacing between them can be controlled. These parameters are tuned to achieve the best possible damage tolerance with respect to both the patch-based and load-path-based criteria.

Based on the limited number of samples, the damage tolerance achieved through the implicit formulation appears to be highly sensitive to the problem setup, including boundary conditions, design space, volume fraction, and loading configuration. With Figs. 20, 21 and 22, three examples are displayed. Compared are the non-fail-safe design x, the explicit fail-safe design y (ground truth), the post-optimized prediction $\tilde{y}$ generated from x, and the implicit fail-safe design $\bar{y}$. Figure 20 shows that the implicit formulation exhibits lower damage tolerance for low volume fractions and design spaces with high aspect ratios. For design spaces with small aspect ratios, exemplified by Fig. 21, the implicit formulation can yield designs with strong damage tolerance. Although the implicit design generally performs worse with respect to the patch-based criterion, it surpasses both $\tilde{y}$ and y in several cases when evaluated using the load-path-based criterion. Exemplified by Fig. 22, the results indicate that the damage tolerance of the implicit design improves with increasing volume fraction. In the example shown, the implicit design again outperforms both $\tilde{y}$ and y with respect to the path-based criterion.

Across the 44 samples considered, the implicit formulation generally outperforms the proposed approach with respect to the path-based criterion, with a median 18.30% lower compliance. In contrast, the implicit formulation does not achieve comparable performance with respect to the patch-based criterion, for which the proposed approach achieves a median compliance that is 81.22% lower.

5.4 Evaluation of computational cost

The computational cost of the proposed approach comprises 12,417 non-fail-safe and explicit fail-safe topology optimizations with 150 and 300 iterations, respectively. Training the cGAN framework required about 52 h, excluding the additional runs performed for hyperparameter tuning.

To assess the computational benefit of the proposed approach compared with the explicit formulation, the average component cost is determined with respect to the dataset.

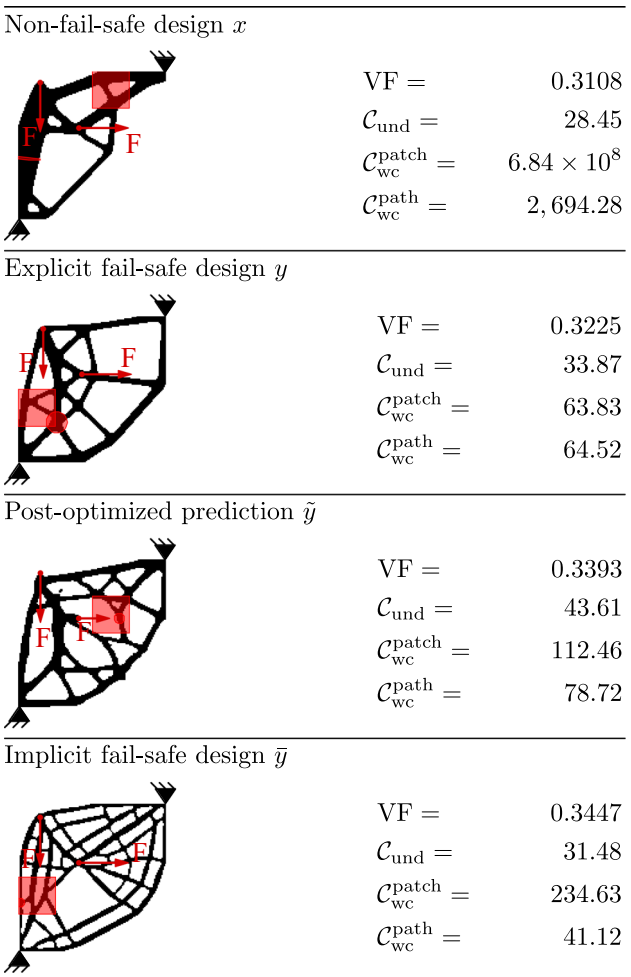

Fig. 21 Performance comparison of non-fail-safe design, explicit fail-safe design, post-optimized prediction, and implicit fail-safe design for test sample 2. Worst-case damage-patch and load-path damage are highlighted by a red square and circle, respectively

Table 1 Estimated average computing times on standardized hardware

	Average computing time in hours
T_x	0.3293
T_y	5.1259 (23.0293 CPU-h)
$T_{\hat{y}}$	0.00167
$T_{\tilde{y}}$	0.4391

Since the dataset was generated on different nodes of a high-performance computing cluster with varying hardware resources, the original computations involved inconsistent levels of parallelization (12-32 CPU cores for the evaluation of damage scenarios). For a consistent relative comparison, the computing times are re-estimated on a single hardware configuration (workstation running Ubuntu 22.04.5 LTS, equipped with an AMD Ryzen 9 7950X processor with 16 physical cores, 32 threads, and 128 GB RAM), using 16 CPU cores for the parallel evaluation of damage scenarios.

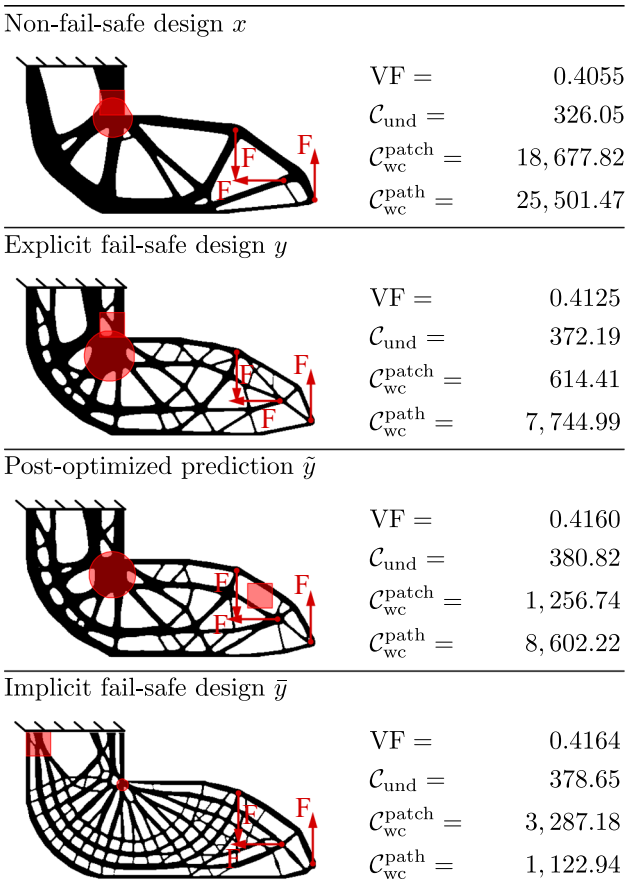

Fig. 22 Performance comparison of non-fail-safe design, explicit fail-safe design, post-optimized prediction, and implicit fail-safe design for test sample 3. Worst-case damage-patch and load-path damage are highlighted by a red square and circle, respectively

Since all topology optimizations were carried out using fixed iteration numbers, the computational cost is primarily governed by the problem size and the number of considered load cases rather than convergence behavior. Accordingly, for every combination of design domain size and number of load cases included in the dataset, one representative run is performed on the selected hardware. Using the frequency of each configuration in the dataset, average computing times are then computed. Table 1 summarizes the average computing times required to generate the non-fail-safe design T_x, the explicit fail-safe design T_y, the predicted fail-safe design $T_{\hat{y}}$, and obtain $T_{\tilde{y}}$ via post-processing.

Once the generator is trained, a single application of the proposed approach consists of three components: a plain minimum-compliance topology optimization, evaluation of the trained model, and a post-optimization step using the modified local volume constraint. Thus, the average cost for one evaluation of the framework, in hours, is given by:

$$T_{\text{eval}} = T_x + T_{\hat{y}} + T_{\tilde{y}} = 0.7701\,\text{h} \tag{12}$$

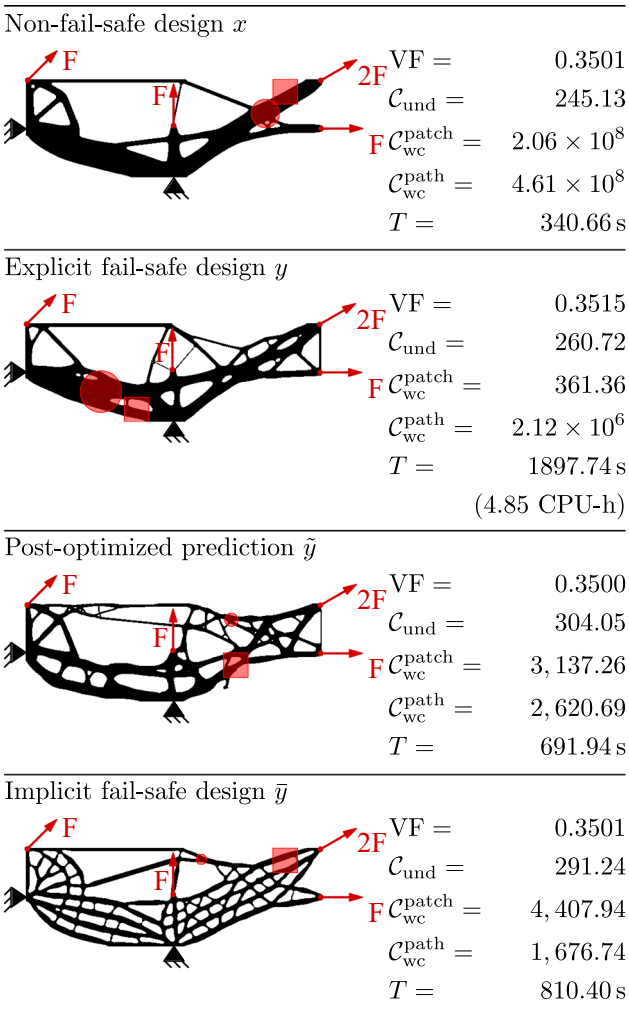

Fig. 23 Performance comparison of non-fail-safe design, explicit fail-safe design, post-optimized prediction, and implicit fail-safe design for a test sample with boundary conditions and loads that were not represented in the training set. Worst-case damage-patch and load-path damage are highlighted by a red square and circle, respectively

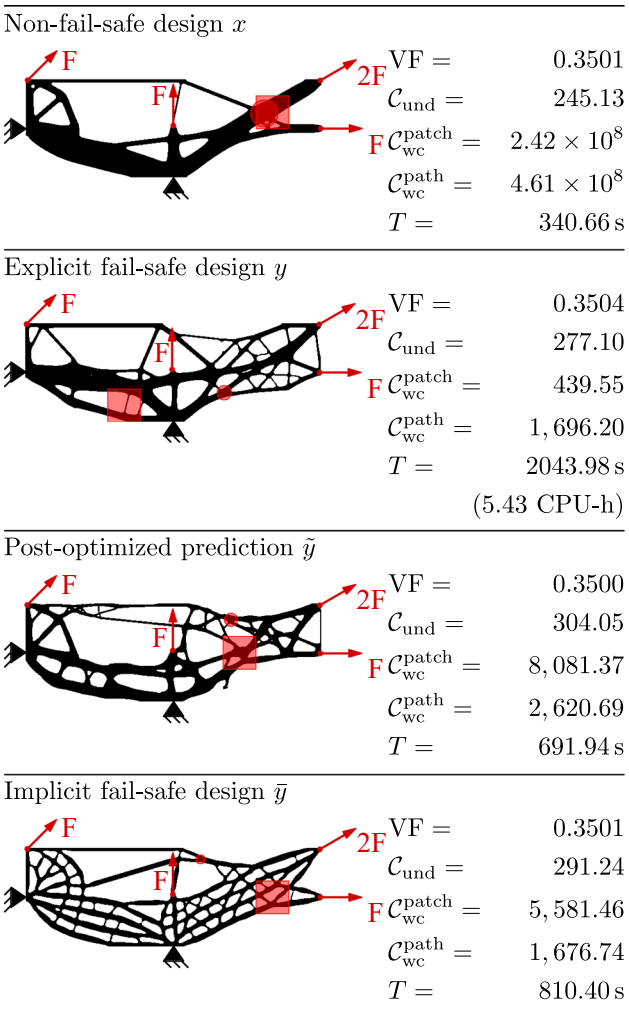

Fig. 24 Performance comparison of non-fail-safe design, explicit fail-safe design, post-optimized prediction, and implicit fail-safe design for a test sample, for a test sample with boundary conditions, loads, a damage size, and a patch increment that were not represented in the training set. Worst-case damage-patch and load-path damage are highlighted by a red square and circle, respectively

Based on the difference between T_y and T_{eval}, the average computational saving per evaluation is $T_{\text{saving}} = 4.3558\,\text{h}$, corresponding to a reduction of 84.98% or a 6.66× speedup. Obviously, the evaluations of the damage scenarios of the explicit formulation are straightforward to parallelize. Consequently, the computational gain per application of the proposed approach strongly depends on the available hardware. Moreover, a net computational benefit is only achieved when the framework is applied sufficiently often for the accumulated savings to offset the initial computational cost of data generation and model training. This initial cost, T_{init}, is given by:

$$T_{\text{init}} = N_{\text{samples}}(T_x + T_y) + T_{\text{train}} = 67,789.22\,\text{h} \tag{13}$$

where the total number of samples is $N_{\text{samples}} = 12,417$ and the training time of the cGAN framework is $T_{\text{train}} = 52\,\text{h}$.

Based on the average saving per application, the break-even point N_{BE}, at which a net computational benefit is achieved, is reached after

$$N_{BE} = \frac{T_{\text{init}}}{T_{\text{saving}}} = 15,562.98 \approx 15,563 \tag{14}$$

applications. If the efficiency of the explicit formulation increases, e.g., through extensive parallelization (Herrero-Pérez and Picó-Vicente 2024), while the efficiency of the proposed approach remains unchanged, the break-even point is further delayed until the per-application savings approach zero.

Although the implicit formulation is computationally inexpensive, with a per-iteration cost comparable to plain compliance minimization, it generally requires more itera-

tions (1000 iterations were used to create the 44 test samples) and therefore requires 64.92% more computing time on average. While the implicit formulation performs well with respect to the path-based criterion, its damage tolerance is highly dependent on the problem setup and an optimal parameter configuration. Moreover, it generally does not match the performance of the post-optimized predictions with respect to the patch-based criterion.

The above considerations show that if a designer has the task to derive a fail-safe structure, it is not beneficial to first generate a dataset, then train a neural network, if it is only used for a few structures. Now that our dataset is available, using it with the presented framework may actually be an alternative approach. Other researchers may come up with a better architecture for the neural network or enhance our data basis, which we strongly encourage.

5.5 Evaluation of versatility

To explore the versatility of the proposed approach, its performance is additionally evaluated on a test sample that deviates more strongly from the training dataset. The problem formulation includes an unseen design domain size ($[L_x, L_y] = [180, 540]$ (Fig. 9)), unseen support conditions, and four load cases of different magnitudes. The results shown in Fig. 23 indicate that the proposed framework indeed provides a redundant design; however, its performance measures are significantly worse than those of the majority of test cases from the training set. While $\tilde{y}$ surpasses x in terms of damage tolerance, it does not nearly reach the performance of y with respect to the patch-based criterion.

To further investigate the versatility of the proposed framework, a different patch-based damage modeling is considered for the evaluation of damage tolerance and the creation of the explicit fail-safe design. In this case, a damage size of $\frac{L_x}{3}$ is used, and the damage increment is set to half the damage size. Figure 24 shows that while the obtained design still looks redundant, its performance further deteriorates when considering unseen damage-patch sizes and increments. This behavior is expected, as the trained model was not exposed to varying damage sizes and does not have the ability to incorporate information about the desired damage size as an input.

As a result of the limited versatility of the proposed approach, the range of problems to which it can be effectively applied is reduced, making the break-even point more difficult to reach in practice. To improve its versatility, the dataset could be extended, for example, by incorporating additional boundary conditions or damage patch sizes.

6 Conclusion

This paper introduces a novel framework for generating fail-safe designs using Artificial Neural Networks, with the aim of exploring the potential and limitations of ANN-based fail-safe topology optimization. Within this framework, a design obtained from plain minimum-compliance topology optimization is converted into a fail-safe design using an ANN. Given the current limitations of ANN-predicted topology-optimized designs, a post-optimization step is proposed to ensure structural connectivity without compromising damage tolerance.

The numerical results allow the following conclusions

- The proposed approach consistently outperforms non-fail-safe designs in terms of damage tolerance, exhibiting median compliance reductions of 96.02% and 96.05% with respect to the patch- and path-based criteria, respectively.
- While reducing computational time by 84.98% on average compared with designs obtained using the explicit formulation, the proposed approach performs significantly worse with respect to the patch-based criterion, showing a median compliance increase of 85.32%. For the path-based criterion, more comparable performance is achieved, with a median increase of 8.91%. These results indicate that the framework is not suitable for safety-critical applications without careful validation.
- Compared to designs obtained using a local volume constraint, the new approach provides better results for patch-shaped damages, exhibiting a median damage compliance that is 81.22% lower. Given a trained ANN, the computational cost of the new approach also remains lower than that of the local volume constraint approach, with an average computational saving of 64.92% for the 44 generated reference designs.

Depending on the available hardware, a single application of the proposed framework is, on average 84.98% computationally less expensive than an explicit fail-safe topology optimization. However, to reach a net benefit, the framework needs to be applied an unrealistically large number of times to offset the substantial cost of data generation and model training, a limitation further aggravated by its limited versatility. Moreover, this break-even point is delayed as the efficiency of the explicit formulation increases, assuming the efficiency of the proposed framework remains unchanged.

Extending the proposed method to three-dimensional problems is generally possible, but not straightforward. This, however, also applies to the other discussed approaches to obtain redundant design. The high computational cost of an explicit consideration of failure patches becomes even higher in 3D. This is not only a challenge for the application of this approach itself, but also the training required for the approach proposed in this paper. The local volume constraint can easily be transferred to 3D. However, it does not necessarily lead to a redundant design. Moreover, topology optimization in 3D

does not provide the truss-like structures as it typically does in 2D, which makes the definition of a load path less clear. These aspects should be investigated in future work.

Lastly, we want to encourage researchers to extend and use the dataset published along with this paper to implement and test their own frameworks.

Appendix A Influence of convolutional kernel parameters

As discussed in Sect. 4.2, the size and number of convolutional kernels of the base model proposed by Nie et al. (2021) are adapted to better suit the presented framework. Due to the reduced number of input channels, the total number of convolutional kernels is decreased by reducing the number of kernels in the first convolutional layer from 128 to 8. Several configurations were evaluated. Table 2 compares 8 and 16 kernels in the first convolutional layer with respect to the generator loss $\mathcal{L}_G$, discriminator loss $\mathcal{L}_D$, mean absolute error (MAE) and volume fraction error (VFE). The results indicate that a larger number of kernels leads to improved training metrics. However, the ratio of damage-patch-based worst-case compliance between ground truth (y) and post-optimized prediction ($\tilde{y}$) (i.e. $\frac{C_{\text{wc}}^{\text{patch}}(y)}{C_{\text{wc}}^{\text{patch}}(\tilde{y})}$), demonstrates that these improvements do not translate into enhanced mechanical performance. Using 8 kernels results in a median ratio of 0.54, whereas 16 kernels yield a lower median ratio of 0.46. Moreover, the training time increases from 52 to 110 h.

In response to the larger input size processed within the presented framework, the kernel size of all convolutional layers is increased from 3×3 to 9×9 to better capture long-range spatial dependencies. Several sizes were evaluated, with 9×9 yielding the best overall performance. The training metrics reported in Table 3 indicate that the larger input size benefits from the use of larger convolutional kernels.

Appendix B Aggregation for modified local volume constraint

To employ the modified local volume fraction constraint, introduced in Sect. 4.4.2, within gradient-based optimization, the non-differentiable maximum operator must be approximated by a smooth aggregation function. Popular choices include the p-norm (A_{pn}) and the p-mean (A_{pm}) aggregation function. Their accuracy depends on the distribution of the aggregated values. A_{pn} is exact when the input consists of the maximum value and all remaining entries are zero, whereas A_{pm} yields the exact maximum when all entries are equal to the maximum value. Consequently, A_{pn} and A_{pm} are more accurate for extremely negatively and positively skewed distributions, respectively.

The proposed modified local volume constraint enforces $\frac{\bar{\rho}_{r,\text{init}}}{\bar{\rho}_r} \leq 1$. Since little to no material can be added globally, the resulting distribution is highly skewed towards 1. In this case, A_{pm} provides very accurate approximations even for small values of p. However, A_{pm} slightly underestimates the true maximum and is therefore not conservative. Using A_{pm} would thus introduce excessive design freedom and compromise structural redundancy. Ambrozkiewicz (2022) introduced a general "p-like" aggregation function,

$$A_g(\boldsymbol{\tau}) = \left(\alpha \sum_{i=1}^{n_t} \tau_i^p\right)^{\frac{1}{p}}, \quad \tau_i \geq 0, \quad \alpha \in \left[\frac{1}{n_t}, 1\right], \tag{B1}$$

where the correction factor α interpolates between A_{pm} (lower bound) and A_{pn} (upper bound) and $\boldsymbol{\tau}$ is an input vector with n_t entries. For skewed distributions, the following correction factor was proposed:

$$\alpha = \frac{pq+1}{n_t}, \tag{B2}$$

where q adjusts the aggregation for negatively ($q \geq 1$) and positively ($q \leq 1$) skewed distributions.

For the post-processing of the designs predicted by the presented model, $q = \frac{1}{8}$ is selected. Although this configuration is slightly less accurate than A_{pm}, it overestimates the true maximum and therefore provides the desired conservatism, ensuring structural redundancy is not compromised. The resulting modified local volume constraint and its sensitivities are given by

$$g(\rho) = \left(\frac{pq+1}{N_r} \sum_{i=1}^{N_r} \left(\frac{\bar{\rho}_{r,\text{init}}}{\bar{\rho}_r}\right)^p\right)^{\frac{1}{p}} - 1 \leq 0 \tag{B3}$$

Table 2 Training results for base convolutional kernel numbers of 8 and 16

Number	$\mathcal{L}_G$	$\mathcal{L}_D$	MAE training	MAE validation	VFE training (%)	VFE validation (%)
8	0.0943	0.4257	0.0427	0.0449	1.10	1.72
16	0.0397	0.4065	0.0397	0.0425	0.89	0.85

Table 3 Training results for convolutional kernel sizes of 3×3 and 9×9

Size	$\mathcal{L}_G$	$\mathcal{L}_D$	MAE training	MAE validation	VFE training (%)	VFE validation (%)
3×3	0.1433	0.4112	0.0796	0.0804	1.45	1.41
9×9	0.0943	0.4257	0.0427	0.0449	1.10	1.72

Table 4 Loss and evaluation metric values of all cross-validation folds

Fold	$\mathcal{L}_G$	$\mathcal{L}_D$	MAE training	MAE validation	VFE training (%)	VFE validation (%)
1	0.0943	0.4257	0.0427	0.0449	1.10	1.72
2	0.1007	0.4206	0.0472	0.0481	1.14	1.33
3	0.0892	0.4056	0.0409	0.0437	0.80	0.79
4	0.1101	0.4498	0.0472	0.0482	2.37	1.15
5	0.1047	0.4343	0.0480	0.0487	1.53	1.08
6	0.1067	0.4221	0.0515	0.0514	1.26	0.86
7	0.1047	0.4431	0.0496	0.0500	1.31	1.01
8	0.1014	0.4905	0.0412	0.0455	2.42	5.39
9	0.1079	0.4091	0.0535	0.0529	1.13	1.15
AVG	0.1022	0.4334	0.0469	0.0482	1.45	1.61
STD	0.0067	0.0258	0.0045	0.0030	0.57	1.44
CV	0.0658	0.0596	0.0953	0.0633	0.3923	0.8974

and

$$\frac{\partial g}{\partial \bar{\rho}_r} = -\frac{\bar{\rho}_{r,\text{init}}^{p}(pq+1)}{\bar{\rho}_r^{p+1} N_r}\left(\frac{pq+1}{N_r}\sum_{r=1}^{N_r}\left(\frac{\bar{\rho}_{r,\text{init}}}{\bar{\rho}_r}\right)^{p}\right)^{\frac{1}{p}-1} \tag{B4}$$

Appendix C Training results of all cross-validation folds

To ensure that the training results are not dependent on a specific data distribution, a cross-validation procedure is employed. The dataset is divided into 10 equally sized subsets. One subset is reserved as a fixed test set, while the remaining nine subsets are used for 9-fold cross-validation. In each fold, eight subsets are used for training and one for validation. In Sect. 5, the training progress with respect to the generator loss $\mathcal{L}_G$, discriminator loss $\mathcal{L}_D$, mean absolute error (MAE), and volume fraction error (VFE) is discussed in detail for the cross-validation fold 1. Table 4 reports the final loss and evaluation metric values for all folds. In addition, the average (AVG), standard deviation (STD), and coefficient of variation (CV) are provided. The CV values indicate that the results are relatively stable across the different folds.

Author contributions Benedikt Hamann developed the methodology, produced results and wrote the manuscript. Benedikt Kriegesmann acquired funding, developed the methodology, and edited the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. Financial support of the German Research Foundation (Project Number 515743381) is acknowledged.

Data availability The data produced for training the neural network are available in the "TUHH Open Research" repository (TORE) of the Hamburg University of Technology and can be accessed via the following link: https://doi.org/10.15480/882.17073.

Code availability The code for handling the dataset is available in the "TUHH Open Research" repository (TORE) of the Hamburg University of Technology and can be accessed via the following link: https://doi.org/10.15480/882.17073.

Declarations

Conflict of interest The authors state that there is no conflict of interest.

Replication of results The author states that the paper contains all information necessary to reproduce the results. Furthermore, the generated training data are provided via an open access repository (see above).

References

Ambrozkiewicz O (2022) Structural optimization of components and joints in assemblies considering fail-safety. PhD thesis, Technische Universität Hamburg, Hamburg, Germany. https://doi.org/10.15480/882.4618

Ambrozkiewicz O, Kriegesmann B (2020) Density-based shape optimization for fail-safe design. J Comput Design Eng 7(5):615–629. https://doi.org/10.1093/jcde/qwaa044

Ambrozkiewicz O, Kriegesmann B (2018) Adaptive strategies for fail-safe topology optimization. In: International conference on engineering optimization. Springer, pp 200–211

Arora JS, Haskell DF, Govil AK (1980) Optimal design of large structures for damage tolerance. AIAA J 18(5):563–570. https://doi.org/10.2514/3.7669

Banga S, Gehani H, Bhilare S, Patel S, Kara L (2018) 3D topology optimization using convolutional neural networks. arXiv. arXiv:1808.07440 [cs] . https://doi.org/10.48550/arXiv.1808.07440

Behzadi MM, Ilieş HT (2021) GANTL: toward practical and real-time topology optimization with conditional generative adversarial networks and transfer learning. J Mech Des 144:021711. https://doi.org/10.1115/1.4052757

Behzadi MM, Chen J, Ilies HT (2024) Taming connectedness in machine-learning-based topology optimization with connectivity graphs. Comput Aided Des 168:103634. https://doi.org/10.1016/j.cad.2023.103634

Bendsøe MP (1989) Optimal shape design as a material distribution problem. Struct Optim 1(4):193–202. https://doi.org/10.1007/BF01650949

Bourdin B (2001) Filters in topology optimization. Int J Numer Meth Eng 50(9):2143–2158. https://doi.org/10.1002/nme.116

Bruns TE, Tortorelli DA (2001) Topology optimization of non-linear elastic structures and compliant mechanisms. Comput Methods Appl Mech Eng 190(26):3443–3459. https://doi.org/10.1016/S0045-7825(00)00278-4

Cang R, Yao H, Ren Y (2019) One-shot generation of near-optimal topology through theory-driven machine learning. Comput Aided Des 109:12–21. https://doi.org/10.1016/j.cad.2018.12.008

Chandrasekhar A, Suresh K (2021) TOuNN: topology optimization using neural networks. Struct Multidisc Optim 63(3):1135–1149. https://doi.org/10.1007/s00158-020-02748-4

Cheng F, Jia H, Ding W, Zuo W, Fang Y (2025) Fail-safe topology optimization for fiber-reinforced composite structures. Compos Struct 364:119145

Chi H, Zhang Y, Tang TLE, Mirabella L, Dalloro L, Song L, Paulino GH (2021) Universal machine learning for topology optimization. Comput Methods Appl Mech Eng 375:112739. https://doi.org/10.1016/j.cma.2019.112739

Fairclough HE, He L, Asfaha TB, Rigby S (2023) Adaptive topology optimization of fail-safe truss structures. Struct Multidisc Optim 66(7):148

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

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 770–778

Herrero-Pérez D, Picó-Vicente S (2024) Adaptive fail-safe topology optimization using a hierarchical parallelization scheme. Comput Struct 291:107205

Hochreiter S (1991) Untersuchungen zu dynamischen neuronalen netzen. Diploma Technische Universität München 91(1):31

Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 7132–7141

Huang X, Li S, Miao C, Hou L, Chen Y (2025) Generative adversarial network for stress-minimizing topology optimization. Int J Mech Mater Des. https://doi.org/10.1007/s10999-025-09811-2

Isola P, Zhu J-Y, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 1125–1134

Jansen M, Lombaert G, Schevenels M, Sigmund O (2013) Topology optimization of fail-safe structures using a simplified local damage model. Struct Multidisc Optim 49(4):657–666. https://doi.org/10.1007/s00158-013-1001-y

Jiang X, Wang H, Li Y, Mo K (2020) Machine learning based parameter tuning strategy for MMC based topology optimization. Adv Eng Softw 149:102841. https://doi.org/10.1016/j.advengsoft.2020.102841

Kallioras NA, Kazakis G, Lagaros ND (2020) Accelerated topology optimization by means of deep learning. Struct Multidisc Optim 62(3):1185–1212. https://doi.org/10.1007/s00158-020-02545-z

Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. arXiv:1412.6980 [cs.LG]

Kranz M, Lüdeker JK, Kriegesmann B (2021) An empirical study on stress-based fail-safe topology optimization and multiple load path design. Struct Multidisc Optim 64:2113–2134. https://doi.org/10.1007/s00158-021-02969-1

Lee S, Kim H, Lieu QX, Lee J (2020) CNN-based image recognition for topology optimization. Knowl-Based Syst 198:105887. https://doi.org/10.1016/j.knosys.2020.105887

Li M, Cheng Z, Jia G, Shi Z (2019) Dimension reduction and surrogate based topology optimization of periodic structures. Compos Struct 229:111385. https://doi.org/10.1016/j.compstruct.2019.111385

Mirza M, Osindero S (2014) Conditional generative adversarial nets. arXiv. arXiv:1411.1784 [cs] https://doi.org/10.48550/arXiv.1411.1784

Napier N, Sriraman S-A, Tran HT, James KA (2019) An artificial neural network approach for generating high-resolution designs from low-resolution input in topology optimization. J Mech Des 142:011402. https://doi.org/10.1115/1.4044332

Nie Z, Lin T, Jiang H, Kara LB (2021) TopologyGAN: topology optimization using generative adversarial networks based on physical fields over the initial domain. J Mech Des 143:031715. https://doi.org/10.1115/1.4049533

Odena A, Dumoulin V, Olah C (2016) Deconvolution and checkerboard artifacts. Distill https://doi.org/10.23915/distill.00003

Parrott CM, Abueidda DW, James KA (2023) Multidisciplinary topology optimization using generative adversarial networks for

physics-based design enhancement. J Mech Des 145:061704. https://doi.org/10.1115/1.4056929
Patel D, Bielecki D, Rai R, Dargush G (2022) Improving connectivity and accelerating multiscale topology optimization using deep neural network techniques. Struct Multidisc Optim 65(4):126. https://doi.org/10.1007/s00158-022-03223-y
Peng X, Sui Y (2021) Lightweight topology optimization with consideration of the fail-safe design principle for continuum structures. Eng Optim 53(1):32–48
Pereira L, Driemeier L (2024) Wasserstein generative adversarial networks for topology optimization. Structures 67:106924. https://doi.org/10.1016/j.istruc.2024.106924
Qiu C, Du S, Yang J (2021) A deep learning approach for efficient topology optimization based on the element removal strategy. Mater Design 212:110179. https://doi.org/10.1016/j.matdes.2021.110179
Rade J, Balu A, Herron E, Pathak J, Ranade R, Sarkar S, Krishnamurthy A (2020) Physics-consistent deep learning for structural topology optimization. CoRR
Ramachandran P, Zoph B, Le QV (2017) Searching for Activation Functions. arXiv. arXiv:1710.05941 [cs] . https://doi.org/10.48550/arXiv.1710.05941
Ramu P, Thananjayan P, Acar E, Bayrak G, Park JW, Lee I (2022) A survey of machine learning techniques in structural and multidisciplinary optimization. Struct Multidisc Optim 65(9):266. https://doi.org/10.1007/s00158-022-03369-9
Rawat S, Shen M-HH (2019) A novel topology optimization approach using conditional deep learning. arXiv. arXiv:1901.04859 [cs] . https://doi.org/10.48550/arXiv.1901.04859
Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical image segmentation. In: Navab N, Hornegger J, Wells WM, Frangi AF (eds) Medical image computing and computer-assisted intervention – MICCAI. Springer, Cham, pp 234–241. https://doi.org/10.1007/978-3-319-24574-4_28
Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016) Improved techniques for training GANs. Adv Neural Inf Process Syst 29
Seo J, Kapania RK (2023) Topology optimization with advanced CNN using mapped physics-based data. Struct Multidisc Optim 66(1):21. https://doi.org/10.1007/s00158-022-03461-0
Shin S, Shin D, Kang N (2023) Topology optimization via machine learning and deep learning: a review. J Comput Design Eng 10(4):1736–1766. https://doi.org/10.1093/jcde/qwad072
Sigmund O (2007) Morphology-based black and white filters for topology optimization. Struct Multidisc Optim 33(4):401–424. https://doi.org/10.1007/s00158-006-0087-x
Sim E-A, Lee S, Oh J, Lee J (2021) GANs and DCGANs for generation of topology optimization validation curve through clustering analysis. Adv Eng Softw 152:102957. https://doi.org/10.1016/j.advengsoft.2020.102957
Stolpe M (2019) Fail-safe truss topology optimization. Struct Multidisc Optim 60(4):1605–1618
Sun PF, Arora JS, Haug EJ Jr (1976) Fail-safe optimal design of structures. Eng Optim 2(1):43–53. https://doi.org/10.1080/03052157608960596
Wang F, Lazarov BS, Sigmund O (2011) On projection methods, convergence and robust formulations in topology optimization. Struct Multidisc Optim 43(6):767–784. https://doi.org/10.1007/s00158-010-0602-y
Wang H, Liu J, Wen G, Xie YM (2020) The robust fail-safe topological designs based on the von mises stress. Finite Elem Anal Des 171:103376
Wang L, Shi D, Zhang B, Li G, Helal WMK, Qi M (2023) Deep learning driven real time topology optimization based on improved convolutional block attention (Cba-U-Net) model. Eng Anal Bound Elem 147:112–124. https://doi.org/10.1016/j.enganabound.2022.11.034
Whiteside EA, Fairclough HE, Rigby SE (2025) A review & critique of optimal fail-safe structural design. Eng Struct 336:120300. https://doi.org/10.1016/j.engstruct.2025.120300
Woldseth RV, Aage N, Bærentzen JA, Sigmund O (2022) On the use of artificial neural networks in topology optimisation. Struct Multidisc Optim 65(10):294. https://doi.org/10.1007/s00158-022-03347-1
Wu J, Aage N, Westermann R, Sigmund O (2018) Infill optimization for additive manufacturing-approaching bone-like porous structures. IEEE Trans Visual Comput Graph 24(2):1127–1140. https://doi.org/10.1109/TVCG.2017.2655523
Xu S, Cai Y, Cheng G (2010) Volume preserving nonlinear density filter based on heaviside functions. Struct Multidisc Optim 41(4):495–505. https://doi.org/10.1007/s00158-009-0452-7
Xue L, Liu J, Wen G, Wang H (2021) Efficient, high-resolution topology optimization method based on convolutional neural networks. Front Mech Eng 16(1):80–96. https://doi.org/10.1007/s11465-020-0614-2
Yamasaki S, Yaji K, Fujita K (2021) Data-driven topology design using a deep generative model. Struct Multidisc Optim 64(3):1401–1420. https://doi.org/10.1007/s00158-021-02926-y
Zhang Y, Zhang H, Qiu L, Wang Z, Zhang S, Qiu N, Fang J (2023) A stochastic framework for computationally efficient fail-safe topology optimization. Eng Struct 283:115831
Zheng S, He Z, Liu H (2021) Generating three-dimensional structural topologies via a U-Net convolutional neural network. Thin-Walled Struct 159:107263. https://doi.org/10.1016/j.tws.2020.107263
Zheng Y, Qiu W, Liu X, Huang Z, Xia L (2025) Damage-tolerant mechanical metamaterials designed by fail-safe topology optimization. Mater Design 249:113546
Zhou M, Fleury R (2016) Fail-safe topology optimization. Struct Multidisc Optim 54(5):1225–1243. https://doi.org/10.1007/s00158-016-1507-1