INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022
https://doi.org/10.1017/pds.2022.161

An AI-Assisted Design Method for Topology Optimization without Pre-Optimized Training Data

A. Halle ✉, L. F. Campanile and A. Hasse

Chemnitz University of Technology, Germany
✉ alex.halle@mb.tu-chemnitz.de

Abstract

Engineers widely use topology optimization during the initial process of product development to obtain a first possible geometry design. The state-of-the-art method is iterative calculation, which requires both time and computational power. This paper proposes an AI-assisted design method for topology optimization, which does not require any optimized data. The presented AI-assisted design procedure generates geometries that are similar to those of conventional topology optimizers, but require only a fraction of the computational effort.

Keywords: topology optimisation, AI-assisted design, computational design methods, design evaluation, design to x (DtX)

1. Introduction

The presented paper deals with the solution of optimization problems by means of artificial intelligence (AI) techniques. Topology optimization (TO) was chosen as an application example, even though the described method is applicable to many optimization problems and thus has generality.
TO is a method of optimizing the geometry of structures. In TO, the material distribution over a given design domain is the subject of optimization, i.e. minimization of a given objective function while satisfying given constraints (Sigmund and Maute, 2013). In most cases, suitable search algorithms solve the optimization problem mathematically.
The combination of AI and TO in the state-of-the-art research mostly requires optimized geometries generated by conventional TO or FEM results as a basis for training. For this reason, they are subject to several limitations that affect those techniques, such as large computational effort and the need to prepare representative data.
The approach proposed here aims at removing those drawbacks by generating all the artificial knowledge required for optimization during the learning phase, with no need to rely on pre-optimized results.

1.1. Topology Optimization

In this work, only the case of mono-material topology optimization is considered. The material of which the structure is to be build is a constant of the problem, and the geometry remains unknown.
The function to be minimized in stiffness optimization is usually the scalar measure of structural compliance. In addition, the filling degree condition must be fulfilled. This filling degree corresponds to the fraction of the maximum possible amount of material (degree of filling) which is to be used in the design and is often also referred to as the volume fraction. Typically considered restrictions are the

available design domain, the static and kinematic boundary conditions for the regarded load cases as well as strength thresholds.
There are numerous possible approaches to TO (Sigmund and Maute, 2013). According to the "Solid Isotropic Material with Penalization" (SIMP) approach of Bendsøe (Bendsøe and Sigmund, 2003), a subdivision of the design domain into elements takes place. A factor, yet to be determined, scales the contribution of each element to the overall stiffness of the structure.
The SIMP approach is able to provide optimized geometries for many practical cases by means of an iterative process. Each iteration involves computationally intensive operations: the most critical ones are assembling the stiffness matrix and solving the system's equation. The involvement of restrictions, such as stress restrictions, increases the complexity of the optimization problem (Lee, 2012; Picelli et al., 2018).

1.2. Artificial Neural Networks

Artificial neural networks (ANNs) belong to the area of machine learning (ML), which, in turn, is assigned to AI. ANNs are able to learn and execute complex procedures, which has led to remarkable results in recent years. ANNs or, more precisely, feedforward neural networks, consist of layers connected in sequence. These layers contain so-called neurons (Karayiannis and Venetsanopoulos, 1993). A neuron is the basic element of an ANN. The combination of all layers is called a network.
The neurons receive inputs, which are linearly combined and added to a bias value and passed as argument to an activation function. The coefficients of the linear combination are called weights. The weights and bias are also sometimes referred to as the trainable parameters.
It is usual that several neurons have the same input. All neurons with the same inputs are grouped together in one layer (also called fully connected layer). The number of layers is also named depth of the network, which also originates the attribute "deep" in the term deep learning (DL).
More details about the learning of an ANN can be found in literature, for example in (Basheer and Hajmeer, 2000; Goodfellow et al., 2016; James et al., 2021; Mohammed et al., 2016).

1.3. Deep Learning-Based Topology Optimization

In the current state of research, there are several publications dealing with ML in the field of TO. Most of these use conventionally topology-optimized geometries (Abueidda et al., 2020; Ates and Gorguluarslan, 2021; Malviya, 2020; Nie et al., 2021; Rawat and Shen, 2019; Yu et al., 2019) for the training of the ANN. The ANNs learned to provide a geometry for specific boundary conditions that has similarities to the training data. As a consequence, the underlying mathematical relationship between the inputs and geometry, such as compliance, was not explicitly part of the training, which was merely data-driven. Thus, while these methods are able to provide directly nearly optimal geometries, they can also produce not interpretable results, such as disconnected structures.
Alternative approaches establish ML as part of the iterative topology optimization process to reduce computation time by partially replacing some of the FEM algorithms with ML algorithms (Behzadi and Ilies, 2021; Chi et al., 2021; Kallioras et al., 2020; Qian and Ye, 2020; Sosnovik and Oseledets, 2017; Yamasaki et al., 2021; Zhang et al., 2019). Even though these approaches sometimes reduce the computation time considerably, they often have to be trained again for new boundary conditions/inputs.
This paper investigates the possibility, which differs from the state-of-the-art methods, to train an ANN without the use of optimized or computationally prepared data. The generation of training data points and the training itself are merged in one single procedural step.

2. Method

The presented method is based on an ANN architecture called predictor-evaluator-network (PEN), which was developed by the authors for this purpose. The predictor is the trainable part of the PEN and its task is to generate, based on input data, optimized geometries.
As mentioned, unlike the state-of-the-art methods, no conventionally topology-optimized or computationally prepared data are used in the training. The geometries used for the training are created by the predictor itself on the basis of randomly generated input data and evaluated by the remaining components of the PEN, called evaluators (see Figure 1).

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press

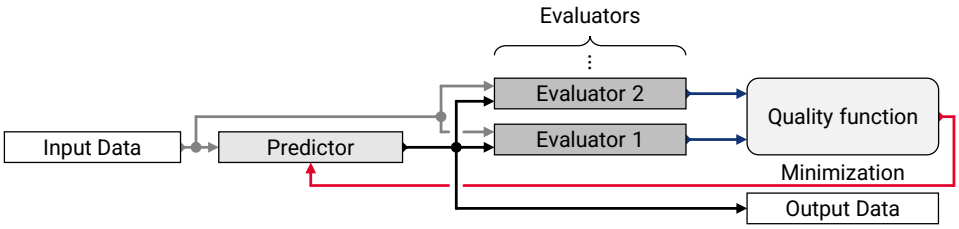

Figure 1. The basic principle of the Predictor-Evaluator-Network (PEN)

The evaluators perform mathematical operations. Other than the predictor, the operations performed by the evaluators are pre-defined and do not change during the training.
Each evaluator assesses the outputs of the predictor with respect to a certain criterion and returns a corresponding scalar value as a measure of the criterion's fulfillment. The deviation from the fulfillment of the considered criterion is the loss or the error of this evaluator.
A scalar function of the evaluator outputs (the quality function) combines the individual losses. During the training, the objective function computed for a set of geometries (batch) is minimized by changing the predictor's trainable parameters. In this way, the predictor learns how to produce optimized geometries.

2.1. Basic Definitions

In topology optimization, the design domain is typically subdivided into elements by appropriate meshing. Figure 2 visualizes the elements (with one element hatched) and nodes.

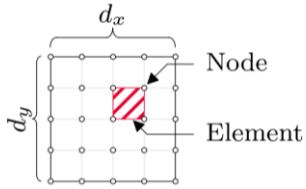

Figure 2. Design space overview with elements, nodes and dimension d (square case)

In this work, we examined only square meshes with equal numbers of rows and columns. However, this method can be used for non-square and three-dimensional geometries.
The total number of elements is as follows:

$$n = d_x d_y \tag{1}$$

where d_y is the number of rows and d_x the number of columns (see Figure 2). In the square case, the number of rows and columns are equal ($d_x = d_y = d$), so that the total number of elements is d^2.
The d^2 design variables x_i $\{i = 1, \dots, d^2\}$, termed density values, scale the contributions of the single elements to the stiffness matrix. The density has a value of one when the stiffness contribution of the element is fully preserved and zero when it disappears.
The density values are collected in a vector $\mathbf{x}$. In general, the density values x_i are defined in the interval $[0, 1]$. In order to prevent possible singularities of the stiffness matrix, a lower limit value $x_{\min}$ for the entries of $\mathbf{x}$ is set as follows (Bendsøe and Sigmund, 2003):

$$0 < x_{\min} \leq x_i < 1, \quad i = 1,2, \dots, d^2. \tag{2}$$

Although a binary selection of the density is desired (discrete TO, material present/not present), values between zero and one are permitted for algorithmic reasons (continuous TO). To get closer to the desired binary selection of densities, the so-called penalization can be used in the calculation of the compliance. The penalization is realized by an element-wise exponentiation of the densities by the penalization exponent $p > 1$ (Sigmund, 2001).
The arithmetic mean of all x_i defines the degree of filling of the geometry as follows:

$$M_{\text{is}} = \frac{1}{d^2} \sum_{i=1}^{d^2} x_i \tag{3}$$

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press

The target value M_{tar} is the degree of filling that is to be achieved by the predictor.
Investigations showed that the training speed (see section 2.8) could be increased, for high-resolution geometries, by dividing the training into levels with increasing resolution. Since smaller geometries are trained several orders of magnitude faster and the knowledge gained is also used for higher resolution geometries, the overall training time is reduced, compared to the training that uses only high-resolution geometries. The levels are labeled with the integer number Λ.
Increasing Λ by one results in doubling the number d of rows or columns of the design domain's mesh. This is done by quartering the elements of the previous level. In this way, the nodes of the previous level are kept in the new level. The number of row or columns at the first level is denoted as d_{inp}.

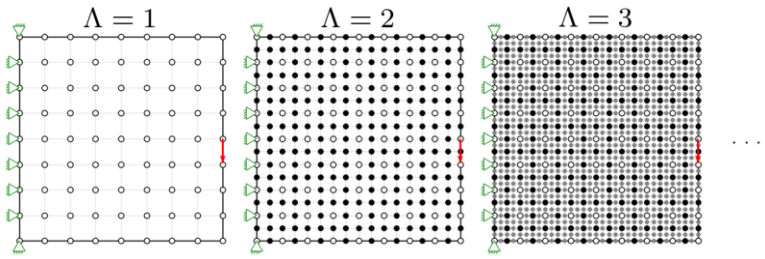

Figure 3. Nodes and elements at different levels Λ (resolutions). The boundary conditions do not change

Input data can be only defined at the initial level and do not change when the level is changed. Hence, new nodes cannot be subject to static or kinematic boundary conditions (see Figure 3). When the level is changed, only the dimension of the outputs changes; the dimension of the inputs remains constant. The change in level occurs after a certain condition, which will be described later, is fulfilled.

2.2. Predictor

The predictor is responsible for generating, after training, the optimized result for a given input data point. Its ANN-architecture consists of multiple hidden layers, convolutional layers and output layers. All parameters that can be changed during training in order to minimize the target function, such as the bias and the weights of the hidden layers, are generally referred to as trainable parameters in the following. The predictor's topology is shown in Figure 4 in a simplified form.

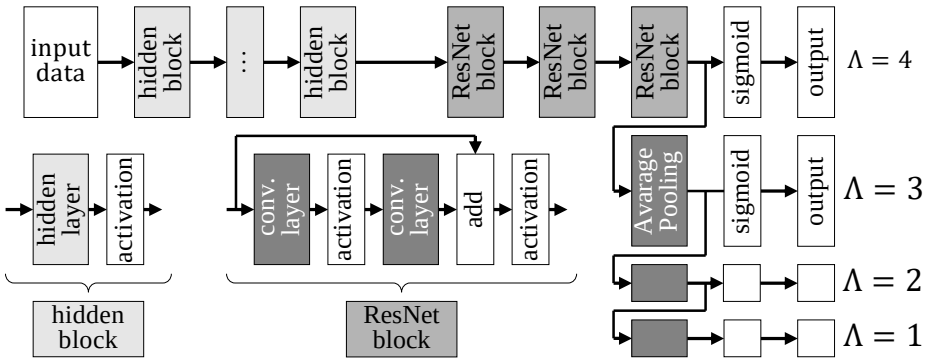

Figure 4. Predictor's artificial neural network (ANN) topology (simplified)

An input data (top left) is processed by several successive hidden blocks and then passed on to some ResNet-blocks. A hidden block is a combination of a hidden layer and an activation. ResNet-blocks consist of multiple convolutional and activation layers (He et al., 2016) (see Figure 4). At this stage, the output is at the highest resolution. The sigmoid function is well suited as an activation function for the output layer because it provides results in the interval $(0, 1)$. This makes the predictor's output directly suitable to describe the density values of the geometry. Average pooling is used in order to reduce the resolution to a lower level Λ.

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press

2.3. Evaluator: Compliance

The task of the compliance evaluator is the computation of the global mean compliance. For this purpose, an algorithm based on FEM (Sigmund, 2001) is used. The compliance

$$c = \mathbf{U}^T\mathbf{K}\mathbf{U} = \mathbf{U}^T\mathbf{F} \tag{4}$$

can be calculated using the stiffness matrix **K**, the forces **F** and the displacements **U**. The stiffness matrix is linearly dependent on the geometry **x** and is defined as follows:

$$\mathbf{K} = \sum_{i=1}^{d^2} x_i^p \mathbf{K}_i \tag{5}$$

Where p is the penalization parameter and $\mathbf{K}_i$ is the unscaled element stiffness matrix. As is usual in (Andreassen et al., 2011; Sigmund, 2001), in the following, the units will be omitted for the sake of simplicity.

2.4. Evaluator: Degree of Filling

The task of this evaluator is to determine the deviation of the degree of filling M_{is} (see (3)) from the target value M_{tar} as follows:

$$M = |M_{\mathrm{tar}} - M_{\mathrm{is}}| \tag{6}$$

By considering the filling degree's deviation M in the objective function, the predictor is penalized proportionally to the extent of the deviation from the target degree of filling M_{tar}.

2.5. Evaluator: Filter

The filter evaluator searches for checkerboard patterns in the geometry and outputs a scalar value F that points to the amount and extent of checkerboard patterns detected. These checkerboard patterns consist of alternating high and low density values of the geometry. They are undesirable because they are difficult to transfer to real parts. These checkerboard patterns exist due to bad numerical modeling (Díaz and Sigmund, 1995).

Several solutions for the checkerboard problem were developed in the framework of conventional topology optimization (Sigmund and Petersson, 1998). In this work, a new strategy was chosen, which allows for inclusion of the checkerboard filter into the quality function. In the present approach, checkerboard patterns are admitted but detected and penalized accordingly.

Since the type of implementation is fundamentally different, it is not possible to compare the conventional filter methods with the filter evaluator.

2.6. Evaluator: Uncertainty

When calculating the density values of the geometry **x**, the predictor should, as far as possible, focus on the limit values $x_{\min}$ and 1 and penalize intermediate values. The deviation from this goal is expressed by the uncertainty evaluator with the scalar variable P. This value increases if the predicted geometry deviates significantly from the limit values and thus penalizes the predictor. This approach differs from the conventional penalization approach: conventional penalization has a direct influence on the compliance. The presented approach only influences the objective function in the form of an evaluation. For this reason the penalization factor (see (5)) is set to $p = 1$ for the training.

2.7. Quality Function and Objective Function

The quality function combines all evaluator values into one scalar. In general, the following formula is used:

$$f_Q = \prod_{i=1}^{n}(\alpha_i E_i + 1), \tag{7}$$

where E_i represents an evaluator output, α_i is the corresponding weighting factor and n is the total number of evaluators. Using the presented evaluators, this equation becomes

$$f_Q = (\alpha c + 1)(\beta M + 1)(\gamma F + 1)(\delta P + 1). \tag{8}$$

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press

Since the training based on single geometries requires large computational effort and would lead to instabilities of the training process (large jumps of the objective function output), a given number of geometries (batch) is predicted and the corresponding quality functions are combined to one scalar value, which acts as the objective function for the optimization determining the training.
During the training α, β, γ and δ have the values 2, 5, 1 and 1 respectively.

2.8. Training

Within one batch, the input data points are randomly generated, and the predictor creates the corresponding geometries **x**. Afterwards, the quality function is computed from the evaluators' losses. The value of the objective function is then calculated for the whole batch. Then, the gradient of the objective function, with respect to the trainable parameters, is calculated. The trainable parameters of the predictor for the next batch are then adjusted according to the gradient descent method to decrease the value of the objective function. In order to apply the gradient descent method, the functions must be differentiable with respect to the trainable parameters (Kingma and Ba, 2017). For this reason, the evaluators and the objective function use only differentiable functions.
When the level increases, the predictor outputs a geometry with higher resolution, and the process starts again.
It is important to stress that, unlike conventional topology optimization, the PEN method does not optimize the density values of the geometry, but only the weights of the predictor.

3. Application

3.1. Implementation

The implementation of the presented method takes place in the programming language Python. The framework Tensorflow with the Keras programming interface is used. Tensorflow is an open-source platform for the development of machine-learning applications (Abadi et al., 2015). In Tensorflow, the gradients necessary for the predictor learning are calculated using automatic differentiation, which requires the use of {differentiable} functions available in Tensorflow (Baydin et al., 2015).
The predictor's topology, with all layers and all hyperparameters, is shown in Figure 5. The chosen hyperparameters were found to be the best after a grid search of all parameters in which the deviations of the predictions from the ones obtained by conventional TO were evaluated. The hyperparameters are displayed by the shape (numerical expression over the arrow pointing outside the block) of the output matrix of a block or by the comment near the convolutional block. The label of the output arrow describes the dimensions of the output vector or matrix. The names of the shapes in Figure 5, e.g. "Conv2D", correspond to the Keras layer names.

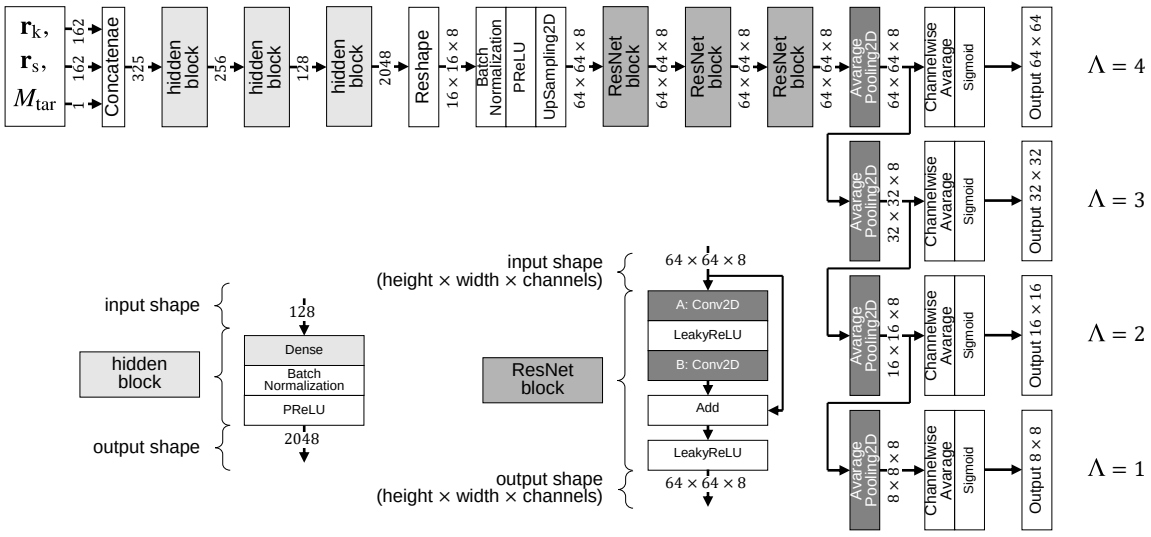

Figure 5. Predictor's artificial neural network (ANN) architecture

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press

As already mentioned, the training of the predictor is based on randomly generated input data sets. All randomly chosen input data are uniformly distributed in the corresponding interval. They are generated according to the following features:

- Kinematic boundary conditions $\mathbf{r}_\mathrm{k}$
  - Fixed degrees of freedom along the left side in x- and y-directions.
- Static boundary condition $\mathbf{r}_\mathrm{s}$
  - Position randomly chosen among all nodes (except the nodes that have a fixed degree of freedom) of level one.
  - Randomly chosen direction in the interval $[0°, 360°]$
  - Fixed magnitude
- Target degree of filling M_tar
  - Randomly chosen direction in the interval [0.2,0.8]

3.2. Results

The training of the predictor lasted 3.25 h (3:15:56), which can be subdivided according to the individual levels as follows: 16%, 7%, 42%, 35%. The training processed approximately 7.6 million randomly generated training data points. As expected, the training time increases proportionally with the size of the geometry. While the first level processed over 3400 data points per second (dps), it became less with each level ($928\ dps, 32\ dps, 4\ dps$). This is due to the additional computational effort and the need to reduce the batch size due to higher memory requirements with higher levels at constant available memory.

The training history shows the progression of the objective function (see Figure 6). The smaller batch size at higher levels produces more oscillation of the curve and therefore, makes it difficult to identify a trend. For this reason, the curves shown in the figures are filtered, using the exponential moving average and a constant smoothing factor of 0.873 (Nicolas, 2017) for all levels. This filtering does not affect the original objective function and serves only visual purposes.

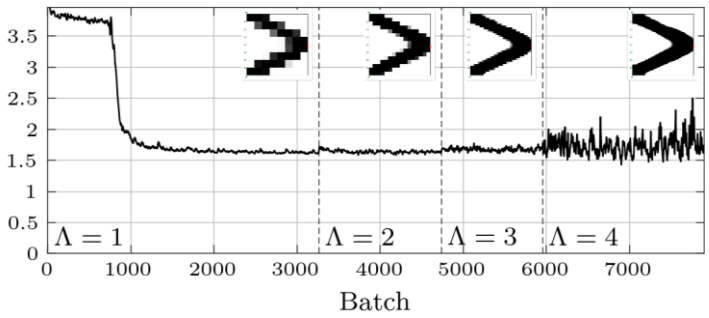

Figure 6. Progression of the objective function value during training

The dashed vertical lines (labeled with the value of Λ) in Figure 6 indicate the change in level. It can be seen that the objective function value improves significantly in the first level Λ and in the following levels only slightly. The reason for this is that learning in one level optimizes the resolution in subsequent levels as well.

The results were validated using 100 optimization problems. The input data for validation was randomly generated and not used for training. The corresponding optimized geometries were conventionally calculated by the top88 algorithm by (Andreassen et al., 2011).

The results of the comparison (PEN and top88) of the 100 validation data points are summarized in the plots in Figure 7.

On average, the ANN-based TO can deliver almost the same result as the conventional method in about 5.2 ms, while the conventional topology optimizer according to (Andreassen et al., 2011) requires, on average, 1.9 s (and is, therefore, roughly 364 times slower); see Figure 7 (left). It can also be seen that the majority of geometries generated by PEN have a compliance that is close to the

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press

geometries generated by top88; see Figure 7 (middle). The geometries deviate in the degree of filling to about one percent; see Figure 7 (right).

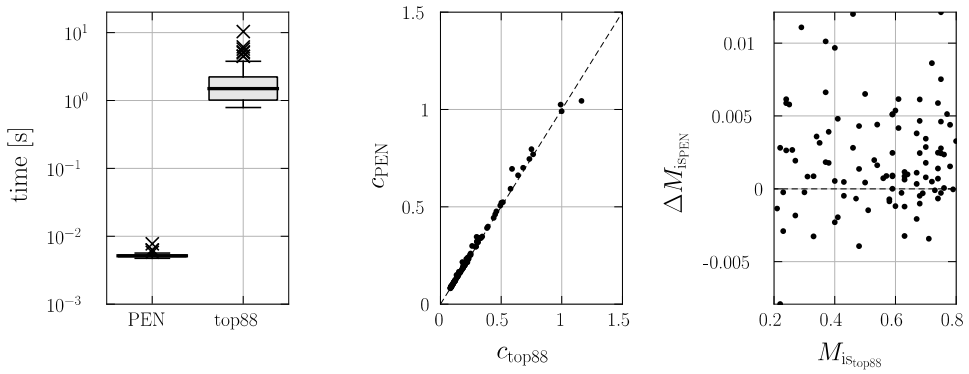

Figure 7. Computing time (left) and compliance (middle) and degree of filling (right) comparison

The examples in Figure 8 show that the predictor can deliver geometries that are similar to the conventional method, as well as some weaknesses. For instance, some geometries lack details (see Figure 8, column four or five). This may be improved by an appropriate choice of layers or hyperparameters of the predictor and by adapting the quality function. For all sample geometries in Figure 8, the compliance is reported under the geometry diagram.

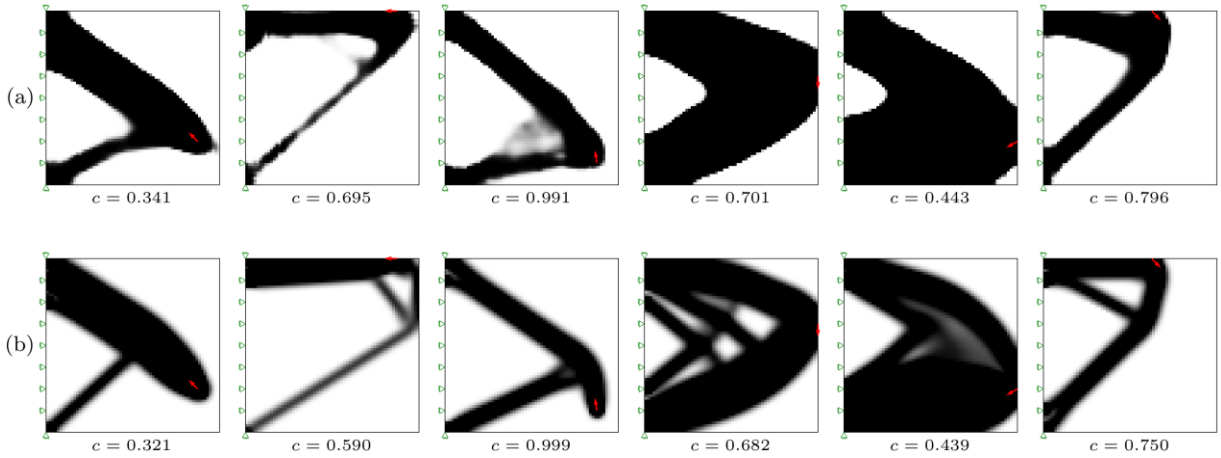

Figure 8. Sample geometries: a) generated with predictor-evaluator network (PEN), b) conventionally generated validation data

It was mentioned that the PEN method is orders of magnitude faster than top88 in predicting geometries. However, the predictor profits from a computationally intensive training. If this training has to be performed globally once, and the topology optimization application is based on this trained predictor, the training time for the end users can be effectively neglected.

3.3. Interactivity

Due to the ability to quickly obtain the optimized geometry by the predictor, the ANN-based TO can be executed interactively in the browser. Under the address: https://www.tu-chemnitz.de/mb/mp/forschung/ai-design/TODL/ (accessed on 22 February 2022), it is possible to perform investigations with different degrees of filling as well as static boundary conditions.

4. Conclusions

In this work, a method was presented to train an ANN using online deep learning and use it to solve optimization problems. In the context of the paper, Topology optimization (TO) was chosen as the

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press

problem to be solved. The ANN in charge of generating topology-optimized geometries does not need any pre-optimized data for the training. The generated geometries are, in most cases, very similar to the results of conventional topology optimization, according to (Andreassen et al., 2011).

This topology optimizer is much faster, due to the fact that the computationally intensive part is shifted into the training. After the training, the artificial neural network based topology optimizer is able to deliver geometries that are nearly identical to the ones generated by conventional topology optimizers (top88 was used as mathematical optimization algorithm). This is achieved by using a new approach: the predictor-evaluator-network (PEN) approach. PEN consists of a trainable predictor that is in charge of generating geometries, and evaluators that have the purpose of evaluating the output of the predictor during the training.

The method was tested for the 2D case up to an output resolution of 64×64. This choice is not a limitation of the method and can be improved by using better hardware for training or by high-performance computing. The use of the method for the 3D case and higher resolutions is conceivable. For this, the predictor would have to output a 3D geometry and the evaluator for compliance would have to be adapted for the 3D case. The optimization of the computational efficiency of the training phase was not the first priority of this project since the training is performed just once and, therefore, affects the performance of the method only in a limited fashion.

A critical step is the calculation of the displacements in the compliance evaluator. The use of faster algorithms (e.g., sparse solvers) could remove the mentioned limitations. One approach to improve the learning process would be to train only in areas where there is a high potential for improvement, through appropriate selection of training data points.

The results of the PEN method are comparable to the ones of the conventional method. However, the PEN method could prove superior in handling applications and optimization problems of higher complexity, such as stress limitations, compliant mechanisms and many more. This expectation is related to the fact that no optimized data are needed. All methods that process pre-optimized data suffer from the difficulties encountered by conventional optimization, while managing the above-mentioned problems. Because the PEN method works without optimized data, it could also be applied to problems that have no optimal solutions or solutions that are hard to calculate, such as the fully stressed truss optimization.

To date, variable kinematic boundary conditions have not been tested. This will be done in future research, together with resolution improvement and application to three-dimensional design domains and different optimization problems.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., et al. (2015), "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", ArXiv:1603.04467 [Cs], available at: https://www.tensorflow.org/.

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

Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B.S. and Sigmund, O. (2011), "Efficient topology optimization in MATLAB using 88 lines of code", Structural and Multidisciplinary Optimization, Vol. 43 No. 1, pp. 1–16.

Ates, G.C. and Gorguluarslan, R.M. (2021), "Two-stage convolutional encoder-decoder network to improve the performance and reliability of deep learning models for topology optimization", Structural and Multidisciplinary Optimization, available at:https://doi.org/10.1007/s00158-020-02788-w.

Basheer, I.A. and Hajmeer, M. (2000), "Artificial neural networks: fundamentals, computing, design, and application", Journal of Microbiological Methods, Vol. 43 No. 1, pp. 3–31.

Baydin, A.G., Pearlmutter, B.A., Radul, A.A. and Siskind, J.M. (2015), "Automatic differentiation in machine learning: a survey", ArXiv:1502.05767 [Cs, Stat], available at: http://arxiv.org/abs/1502.05767 (accessed 23 September 2019).

Behzadi, M.M. and Ilies, H.T. (2021), "GANTL: Towards Practical and Real-Time Topology Optimization with Conditional GANs and Transfer Learning", Journal of Mechanical Design, pp. 1–32.

Bendsøe, M.P. and Sigmund, O. (2003), Topology Optimization: Theory, Methods, and Applications, Springer, Berlin; Heidelberg; New York.

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press

Chi, H., Zhang, Y., Tang, T.L.E., Mirabella, L., Dalloro, L., Song, L. and Paulino, G.H. (2021), "Universal machine learning for topology optimization", Computer Methods in Applied Mechanics and Engineering, Vol. 375, p. 112739.
Díaz, A. and Sigmund, O. (1995), "Checkerboard patterns in layout optimization", Structural Optimization, Vol. 10 No. 1, pp. 40–45.
Goodfellow, I., Bengio, Y. and Courville, A. (2016), Deep Learning, The MIT Press, Cambridge, Massachusetts, available at: http://www.deeplearningbook.org.
He, K., Zhang, X., Ren, S. and Sun, J. (2016), "Deep Residual Learning for Image Recognition", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), presented at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.
James, G., Witten, D., Hastie, T. and Tibshirani, R. (2021), An Introduction to Statistical Learning: With Applications in R, Springer US, New York, NY, available at:https://doi.org/10.1007/978-1-0716-1418-1.
Kallioras, N.Ath., Kazakis, G. and Lagaros, N.D. (2020), "Accelerated topology optimization by means of deep learning", Structural and Multidisciplinary Optimization, Vol. 62 No. 3, pp. 1185–1212.
Karayiannis, N.B. and Venetsanopoulos, A.N. (1993), Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications, Kluwer Academic, Boston.
Kingma, D.P. and Ba, J. (2017), "Adam: A Method for Stochastic Optimization", ArXiv:1412.6980 [Cs], available at: http://arxiv.org/abs/1412.6980 (accessed 12 February 2020).
Lee, E. (2012), Stress-Constrained Structural Topology Optimization with Design-Dependent Loads, University of Toronto, September, available at: https://tspace.library.utoronto.ca/handle/1807/32254.
Malviya, M. (2020), A Systematic Study of Deep Generative Models for Rapid Topology Optimization, preprint, engrXiv, available at:https://doi.org/10.31224/osf.io/9gvqs.
Mohammed, M., Khan, Muhammad Badruddin, and Bashier, Eihab Bashier Mohammed. (2016), Machine Learning : Algorithms and Applications, CRC Press, available at:https://doi.org/10.1201/9781315371658.
Nicolas, P.R. (2017), Scala for Machine Learning - Second Edition, Packt Publishing, Limited, Birmingham, UNITED KINGDOM, available at: http://ebookcentral.proquest.com/lib/tuchemnitz/detail.action?docID=5061334.
Nie, Z., Lin, T., Jiang, H. and Kara, L.B. (2021), "TopologyGAN: Topology Optimization Using Generative Adversarial Networks Based on Physical Fields Over the Initial Domain", Journal of Mechanical Design, Vol. 143 No. 3, p. 031715.
Picelli, R., Townsend, S., Brampton, C., Norato, J. and Kim, H.A. (2018), "Stress-based shape and topology optimization with the level set method", Computer Methods in Applied Mechanics and Engineering, Vol. 329, pp. 1–23.
Qian, C. and Ye, W. (2020), "Accelerating gradient-based topology optimization design with dual-model artificial neural networks", Structural and Multidisciplinary Optimization, available at:https://doi.org/10.1007/s00158-020-02770-6.
Rawat, S. and Shen, M.-H.H. (2019), "A Novel Topology Optimization Approach using Conditional Deep Learning", ArXiv:1901.04859 [Cs, Stat].
Sigmund, O. (2001), "A 99 line topology optimization code written in Matlab", Structural and Multidisciplinary Optimization, Vol. 21 No. 2, pp. 120–127.
Sigmund, O. and Maute, K. (2013), "Topology optimization approaches: A comparative review", Structural and Multidisciplinary Optimization, Vol. 48 No. 6, pp. 1031–1055.
Sigmund, O. and Petersson, J. (1998), "Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima", Structural Optimization, Vol. 16 No. 1, pp. 68–75.
Sosnovik, I. and Oseledets, I. (2017), "Neural networks for topology optimization", ArXiv:1709.09578 [Cs, Math], available at: (accessed 11 May 2020).
Yamasaki, S., Yaji, K. and Fujita, K. (2021), "Data-driven topology design using a deep generative model", ArXiv:2006.04559 [Physics, Stat].
Yu, Y., Hur, T., Jung, J. and Jang, I.G. (2019), "Deep learning for determining a near-optimal topological design without any iteration", Structural and Multidisciplinary Optimization, Vol. 59 No. 3, pp. 787–799.
Zhang, Y., Chen, A., Peng, B., Zhou, X. and Wang, D. (2019), "A deep Convolutional Neural Network for topology optimization with strong generalization ability", ArXiv:1901.07761 [Cs, Stat].

https://doi.org/10.1017/pds.2022.161 Published online by Cambridge University Press