Chapter 10
Overview on Machine Learning Assisted Topology Optimization Methodologies

Ilias Chamatidis, Manos Stoumpos, George Kazakis, Nikos Ath. Kallioras, Savvas Triantafyllou, Vagelis Plevris, and Nikos D. Lagaros

10.1 Introduction

The past two decades saw tremendous developments in artificial intelligence (AI). Advancements in software, algorithms, and hardware led to the development of significantly more accurate and versatile artificial intelligence models. This rendered artificial intelligence a powerful tool that is used in diverse scientific areas, e.g. medicine and drug design, economics, and self-driving cars, among many others. These methods, having been successfully implemented in the simulation and modeling of structures (Lu et al. 2022; Solorzano and Plevris 2022), found their way to topology optimization problems, where artificial intelligence appears to have great potential for successful implementation.

In conventional topology optimization, the optimal design of a specific domain must be calculated subject to specific constraints and the objective is to minimize the total compliance of the structure and use a specific amount of material. This is typically an iterative process that involves large matrices and can be very time-consuming. By means of artificial intelligence models, referred to also as surrogate models (or surrogates), the computing time can be reduced significantly. The surrogate model is apriori trained offline. Following, during the optimization process the model is inferred based on input data, which is a lot faster due to limited matrix multiplications that the surrogate performs. The usual process involves either an artificial intelligence surrogate that complements the conventional procedure to reduce computational costs or a standalone surrogate which calculates the whole optimized

I. Chamatidis · M. Stoumpos · G. Kazakis · N. Ath. Kallioras · S. Triantafyllou · N. D. Lagaros (✉)
National Technical University of Athens, Athens, Greece
e-mail: nlagaros@central.ntua

V. Plevris
Qatar University, Doha, Qatar

T. Rabczuk and K. J. Bathe (eds.), *Machine Learning in Modeling and Simulation*, Computational Methods in Engineering & the Sciences,
https://doi.org/10.1007/978-3-031-36644-4_10

structures by itself. The AI surrogates that are used belong to two main categories, i.e. Surrogates that use density and surrogates that use images.

The surrogates that use density have similar inputs as the conventional method since the optimization process uses the density of the structure and is updated in each iteration of the AI model. The surrogates that perform optimization on images are a bit different because they use techniques like image segmentation and filtering to output the optimized image (structure) which then is mapped into density. Most surrogates can be used for 2D and 3D structures and they are transferable, meaning that once trained they can be used in another topology optimization problem (thermodynamics or different material).

The *Background* section contains an introduction to artificial intelligence, the surrogate models that will be used and an introduction to conventional topology optimization. The *Literature Survey* section provides a review of recent advancements of topology optimization using artificial intelligence models. This section is divided into two parts, the first describing the models that use density and the second the models that use image-based approaches.

10.2 Background

10.2.1 Topology Optimization

Topology optimization is the process of finding the optimal design of a structure. The term optimal refers to a structure which has the same structural properties as the initial one, serves the same purposes, but uses less material. That process is very important because a structure with reduced material demand is easier to implement and less expensive to construct. However, the process of topologically optimizing a structure is an arduous task; it is an iterative process that requires many calculations, and it is very time-consuming. The mathematical formulation of the problem is to find the optimal material distribution while keeping the compliance minimum. The most popular method used in topology optimization formulation is the so-called power-law approach introduced by Bendsøe (1989) and later suggested by Zhou and Rozvany (1991) and Mlejnek (1992). This approach is based on SIMP (Solid Isotropic Material with Penalization) or the modified SIMP (Bendsøe and Sigmund 2004) where each element e is assigned a density x_e which determines its Young's modulus E_e according to the following expression:

$$E_e(x) = E_{\min} + x_e^p(E_0 - E_{\min}), \quad x_e \in [0, 1] \tag{10.1}$$

where E_0 is the original value of Young's modulus, $E_{\min}$ is a very small positive value used to avoid a case where the stiffness matrix becomes singular and p is a penalization factor that ensures that densities belong in [0, 1]. The optimization problem is formulated as follows:

$$\begin{bmatrix} \min_x : c(x) = U^T K U = \sum_{e=1}^{N} E_e(x_e) u_e^T k_0 u_e \\ \text{subject to :} \\ V(x)/V_0 = f \\ KU = F \\ 0 \leq x \leq 1 \end{bmatrix} \tag{10.2}$$

where c is the compliance that needs to be minimized, U and F are the global displacement and force vectors, respectively, K is the global stiffness matrix, u_e is the element displacement vector, k_0 is the element stiffness matrix for an element with unit Young's modulus, x is the vector of design variables, N is the number of variables, $V(x)$ and V_0 are the material volume and design domain volume, respectively, and f is the volume fraction that is chosen beforehand. The updates of the new densities are based on the optimality criteria:

$$x_e^{\text{new}} = \begin{cases} \max(0, x_e - m) \ if \ x_e B_e^{\eta} \leq \max(0, x_e - m) \\ \min(0, x_e + m) \ if \ x_e B_e^{\eta} \geq \max(1, x_e - m) \\ x_e B_e^{\eta}, \ \text{otherwise} \end{cases} \tag{10.3}$$

where m is a positive move limit and B is obtained from the optimality condition:

$$B_e = \frac{-\frac{\partial c}{\partial x_e}}{\lambda \frac{\partial V}{\partial x_e}}, \tag{10.4}$$

where the Lagrangian multiplier λ is chosen so that the volume constraints are satisfied. The sensitivities of the parameters c and V in terms of x_e are

$$\begin{matrix} \frac{\partial c}{x_e} = -p x_e^{p-1} (E_0 - E_{\min}) u_e^T k_0 u \\ \frac{\partial V}{x_e} = 1 \end{matrix} \tag{10.5}$$

The last step of that process is to make sure that densities do not produce weird patterns. For this reason, a filtering is applied.

10.2.2 Artificial Intelligence and Neural Networks

Artificial intelligence is a large area that involves many scientific fields such as mathematics, computer science and pertains to many algorithms that exist, which can "learn" by example in order to solve a problem. By examining many different data points, these algorithms can approximate the underlying function that describes the problem. Thus, after the learning process has been completed, when a new example

is presented, they can predict a result based on the previous training. There are three main categories that describe those algorithms:

- **Supervised learning**: In this kind of learning the samples that the model uses for training have labels, meaning that after the model predicts an output based on the input it received, it has the labels of the "ground truth" value of the sample and then it compares the "ground truth" value with the one that it predicted and corrects itself accordingly. Some of the most popular supervised learning models used are neural networks, support vector machines, and k-nearest neighbor. Especially in the case of neural networks, the last decades have seen great advancements by using deep neural networks which utilize many hidden layers and a very large number of nodes.
- **Unsupervised learning**: This can be considered as the opposite of supervised learning. Here the data are unlabeled, and the model draws conclusions based on the statistical properties of the data, e.g. the relevant clusters and distances. The most popular models for unsupervised learning are K-means, self-organizing maps, and principal component analysis. The main difference between unsupervised and supervised learning is that in unsupervised learning there are no labels ("ground truth" values) used during the training of the model.
- **Reinforcement learning**: This is a different method of learning, which does not have many applications in topology optimization. Central to this method is the notion of the agent (robot) that learns an optimal behavior by interacting with its environment and changing its behavior accordingly.

In topology optimization literature the process is done, most of the times, using deep neural networks. Neural networks are universal function approximators (Hornik et al. 1989). Hence, by properly training an artificial neural network, this can approximate the function that underlies the topology optimization problem.

Each node of the neural network performs an operation between the input and the weights and biases of the model. After passing all the layers it calculates the error and by backpropagating it corrects its weights to minimize the prediction error. This process is iterative, where each iteration in which the model sees the entire dataset is termed an epoch. The number of epochs that are chosen depends on the problem, the amount of data, and the type of the model. The process of training a neural network consists of three steps. The first step is to split the data into the training set, the validation set, and the testing set. This split must be uniformly distributed, and the class representation must be the same in all three sets. The second step is to train the neural network using the training set and use the validation set periodically during the training to test that the model doesn't overfit. Overfitting is an undesirable outcome of the training, where the neural network keeps reducing the error associated with the training data, while the error associated with other data (validation data, testing data, or others) increases, which means that the network overfocuses on the specific training data and has lost its generalization capabilities. The last step is to use the testing set, which the model has not seen before, to measure how well the network performs when confronted with new data. Another issue that must be solved during the training is defining the architecture of the neural network, which has to do with the

number of hidden layers and the number of node in each hidden layer. If the number of layers/nodes is too small, the model will not have enough learning capacity to approximate the function properly, while if the number is too large the model will overfit the dataset and it will not perform well on the test set (low variance-high bias model).

Another variation of deep neural networks that is often used in topology optimization is the convolutional neural networks (CNNs). These models take as their input an image (either 2D or 3D) instead of a simple 1D output. Then, in each node, instead of the usual multiplication between weights and the input, perform a convolution using a small square kernel:

$$(f \cdot g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{10.6}$$

The advantage of these models is that they can unravel localized relations on the image because they use information from an area instead of a single number.

10.3 Literature Survey

10.3.1 Density-Based Methods

Patel and Choi (2012) harness the power of Probabilistic Neural Networks (PNN) and develop an optimization methodology that treats probabilistic constraints under uncertainty. Probabilistic Neural Networks (PNN) rely on Bayesian inference (Clarke 1974) to make decisions, and the Parzen nonparametric estimator (Parzen 1962) for the estimation of the probability density functions. The three main benefits of using a probabilistic approach are the following: (i) Easy interpretation of the results, (ii) Efficiency in treating nonlinear structures or disjoint failures, and (iii) Useful in treating uncertainty. The described network is both easy to implement and to interpret. Its training strategy relies on reducing the expected risk of each class (failure or not). For example, suppose that ϑ belongs either to class $\vartheta_A$orϑ_B, and the data vector is p dimensional $X^T = [X_1, X_2, \ldots, X_p,]$, the Bayes decision rule is

$$d(x) = \begin{cases} \vartheta_A if\ h_A l_A f_A(X) > h_B l_B f_B(X) \\ \vartheta_B if\ h_A l_A f_A(X) < h_B l_B f_B(X) \end{cases} \tag{10.7}$$

where $f_A(X)$ and $f_B(X)$ are the probability density functions (PDF) for the two classes A and B, and l_A is the loss function associated with the decision $d(x) = \vartheta_B$ when $\theta = \theta_a$. Also h_A is the a priori probability of occurrence of class A and $h_B = 1 - h_A$. The loss function used during the training is

$$l = (h_A - \theta_A)^2 + (h_B + \theta_B)^2 \tag{10.8}$$

The architecture of the probabilistic neural network consists of four layers: (i) the input layer, (ii) the pattern layer, (iii) the summation layer, and (iv) the output layer. The output of the pattern layer using the Parzen window nonlinear function forms the PDF. A reliability analysis can be incorporated into the deterministic topology optimization method. This process is called Reliability-Based Topology Optimization (RBTO) and employs a probabilistic constraint such that

$$\begin{cases} \max / \min\limits_{b} : f(b) \\ \text{subject to :} \\ P_j[g_j(b, x)0] \leq R_{Rj} \\ \sum\limits_{i=1}^{N} A_i L_i - V^* \leq 0 \\ K \cdot u = F \\ b_l \leq b \leq b_u \end{cases} \tag{10.9}$$

where $f(.)$ represents the objective function, $g_j(.)$ the limit-state function, b is the vector of deterministic design variables, and x is a random vector. P_j denotes the probability of the event, and the probability of failure can be expressed as $P_j[g_j(.) < 0]$, A_i is the cross-sectional area of the elements and L_i is the length of the particular element, V^* denotes the volume of the material that can be used in the final design, A_l and A_u are the upper and lower bounds of the cross-sectional area of the elements, K is the global stiffness matrix, u is the global displacement vector, and F is the nodal load vector. The PNN consists of two blocks, where the first block performs the topology optimization and the second block the reliability analysis. The force, boundary conditions, and the final volume V^* are used as an input. The PNN is used to reduce the instances where the Finite Element Analysis routine is invoked, thus reducing the computational cost of the entire process.

In Liu et al. (2015), a nonlinear multi-material topology optimization is developed using unsupervised machine learning algorithms. Unsupervised algorithms construct clusters of data based on their similarity. The model takes as an input the normalized material parameter x_e, where $0 \leq x_e \leq 1$. The K-means algorithm provides the initial design of the structure. The final optimization design is obtained with a metamodel-based multi-objective optimization strategy. This strategy consists of five steps: *Sampling*, *Simulation*, *Metamodel fit*, *Optimization,* and *Point selection* for the metamodel update. Sampling and simulation consist of choosing various design experiments and functions evaluated on those designs required to fit the metamodel. The fit of the metamodel pertains to fitting all known functions to approximate the design responses that have not yet been evaluated. The model that is used in this stage is the Kriging metamodel with a linear regression kernel and spherical correlation. The last stage of the optimization step uses a multi-objective genetic algorithm to

find the Pareto front. The Pareto front consists of solutions whose objectives are not dominated by other solutions. The algorithm continues until the difference between the solution of the model and the "ground truth" value is acceptable. The solution of the model was compared with solutions from the SIMP optimization algorithm. The proposed method achieves highly similar designs from metamaterials with SIMP, at a reduced computational cost.

Lei et al. (2018) develop a real-time topology optimization procedure using machine learning. The method proposed is based on the Moving Morphable Component (MMC), where a set of morphable components are used as the basic building blocks. The optimization process consists of morphing, merging, and overlapping operations on those elements to achieve the final structure. The machine learning problem is formulated as follows: Suppose there is a set of parameters $p = (p_1, ..., p_{np})^T$ denoting the location of the concentrated load and D^{opt} the vector of optimal designs as an approximation of linear combination of eigenvectors $v_1 \in \Re^{7n}, ..., v_m \in \Re^{7n}$, where n is the number of components. There are seven design variables for each component. Hence, D^{opt} can be expressed as

$$D^{\text{opt}}(p) = \sum_{i=1}^{M} w_i(p) v_i \tag{10.10}$$

where $w_i(p)$, $i = 1, \ldots, M$ is a set of weights depending on p and M is the number of eigenvectors that represent D^{opt}, $M << 7n$ to achieve substantial dimensionality reduction. Assuming that there are K number of vectors of optimal design $D_1^{\text{opt}}, \ldots, D_K^{\text{opt}}$ as K number of vectors of parameters $p_1 = (p_1^1, \ldots, p_{np}^1)^T, \ldots, p_K = (p_1^K, \ldots, p_{np}^K)^T$ are obtained from direct optimization. By resampling the above set, a larger set is constructed and $(p_1, \ldots, p_K)$ is expanded to $(p_1, \ldots, p_L)$ with $L >> K$. Furthermore, matrix $Y^T Y$ with size $7n \times 7n$ where $Y^T = (D_1^{\text{opt}}, \ldots, D_L^{\text{opt}}) \in \Re^{7n \times L}$, and D_i^{opt} denotes the optimal design variables for $\boldsymbol{p_i}$ in the expanded set $(p_1, \ldots, p_L)$. Then, the eigenvectors $v_1 \in \Re^{7n}, \ldots, v_M \in \Re^{7n}$ can be obtained by solving the eigenvalue problem, i.e. a standard principal components procedure (PCA): $(Y^T Y)v = \lambda v$. From the above the first M vectors are selected. Subsequently, $Y^T = V W^T$, where $V = (v_1, \ldots, v_M) \in \Re^{7n \times M}$ and $W^T = (w_1, \ldots, w_L) \in \Re^{7n \times M}$ with $w_i = (w_1^i, \ldots, w_M^i)^T \in \Re^M, i = 1, \ldots, L$. The following relationship is defined:

$$p_1 \to w_i = (w_1^1, \ldots, w_M^i)^T, \ldots, p_L \to w_L = (w_1^L, \ldots, w_M^L)^T \tag{10.11}$$

The above equation can be approximated by any nonlinear regressor to approximate the mapping between $p \in \Re^{np}$ and $D^{\text{opt}} \in \Re^{7n}$. The machine learning algorithms used are Support Vector Regressor (SVR) and KNN (k-Nearest Neighbors). This dimensionality reduction and the fact that training is performed offline allow the optimization procedure to be performed in real time. Furthermore, the reason that the MMC method was selected has to do with the numbers of components n that are

required to describe a material distribution approach, $O(10^2)$, both in 3D and 2D; conversely the SIMP method would require m pixels that approach $O(10^6 - 10^7)$ to achieve a similar high-resolution result of the structural layout. The results show that SVR performs better than KNN. Both algorithms output different structures than the conventional optimized structure, especially in complex areas.

In Kallioras et al. (2020), deep belief networks are used to produce a higher-level of representation and try to produce a mapping between the input and the output of the structure. Deep Belief Networks (DBN) use stochastic variables to find high-level correlations in the training data called feature detectors. These feature detectors and input data used with Restricted Boltzmann Machines (RBM) form a Deep Belief network. The architecture of the RBM consists of two layers, the input layer and the hidden layer. The hidden layer is the so-called feature detector layer where each node of that layer is only connected with the input layer. Similarly to the cost functions that classical neural networks use, RBM use an energy function:

$$E(v, h) = -\sum_{i=1}^{i\,\max} \alpha_i v_i - \sum_{j=1}^{j\,\max} b_j h_j - \sum_{i=1}^{i\,\max}\sum_{j=1}^{j\,\max} v_i h_j w_{ij} \tag{10.12}$$

where v and b are the state and bias of the ith visible unit, h and b are the state and bias of the jth hidden unit and w_{ij} is the weight coefficient of the connection between those units. The state of the network with the lowest energy is the one with the higher probability.

$$p(v, h) = \frac{1}{Z} e^{-E(v,h)} \tag{10.13}$$

where Z is

$$Z = \sum_{i=1}^{i\,\max}\sum_{j=1}^{j\,\max} e^{-E(v,h)} \tag{10.14}$$

The difference between restricted and normal Boltzmann Machines is that in the restricted ones there are no connections between the hidden units. A Deep Belief Network (DBN) is eventually created by combining multiple RBM. The hidden layer of one RBM is the visible layer (input layer) of the next RBM. The training of the whole model involves two steps: first each RBM is trained individually using unsupervised learning and then the whole model is trained using supervised learning. The proposed method outputs a density value for each point and is also integrated with SIMP to accelerate the optimization process. At the beginning, some initial iterations are performed using SIMP and then the DBM performs the predictions of the density. The training, validation, and test dataset are constructed using SIMP to solve the optimization problem. The size of the dataset of different samples using the cantilever examples is 480,000 samples. The proposed methodology succeeds

with a reduction in SIMP iterations that reaches as high as 90% with a loss similar to the one achieved by SIMP. Also, it is scalable for many finite elements and can be applied both in 2D and 3D structures.

In Kallioras and Lagaros (2021b), deep belief neural networks are used to accelerate the topology optimization process by skipping SIMP iterations while using the AI models to predict the desired density. SIMP is run for the initial iterations and then the models finalize the design. The proposed method improves the method introduced above by harnessing the power of DBN for quick calculations to operate on higher orders (fine mesh) and use SIMP on a coarse mesh to assist the models. This method accelerates the whole process by at least one order of magnitude. In Kallioras and Lagaros (2020), a sequential collection of DBM is introduced where they take as an input the initial iterations of SIMP and try to find hidden patterns and correlations between initial densities of finite elements and the final densities. An improvement of the previous models is introduced in Kallioras et al. (2021), where the models are reduced in their order and by using deep learning the results are extrapolated to a fully refined model thus achieving great accuracy and speeding up as high as 80%. Another interesting method is introduced in Kallioras and Lagaros (2021a) which is about a tool to generate equivalent shapes given an input with a number of elements, forces, and supports. The produced shapes are not optimized but are a collection of different shapes that act as a design inspiration. The output of the process is compatible with 3D printers. The tool is powerful for prototyping designs. It uses SIMP and Long Short-Term Memory Neural Networks (LSTM NNs) and image processing methods to generate the shapes.

The work by White et al. (2019) focuses on large macroscale structures with spatially varying metamaterials. To calculate the density of each element, a neural network is used with a Gaussian activation function, i.e.

$$\Psi(x) = e^{-x^2} \tag{10.15}$$

With the addition of the Gaussian function, the neural network emulates radial basis function interpolation. The weights and biases of the neural network consist of the scaling and offset of the Gaussian function and are calculated from the training process. The neural network uses both the actual densities as an input and their derivatives, for better accuracy. Apart from the use of the Gaussian function, the architecture is a classical, one hidden layer neural network. Experiments were tested with different numbers of neurons in the hidden layer. The results show that when the dataset is small, using derivative data is largely beneficial (leading to 9 times smaller error when using derivatives). However, the use of derivatives becomes irrelevant when the dataset is large and the neural network has enough capacity.

In Chandrasekhar and Suresh (2021), the density function is represented by the weights of the Neural Network. The difference in this approach is that it does not try to accelerate the classic SIMP process by skipping some steps using NNs (image segmentation methods, etc.). Rather, the NN is used to directly perform Topology Optimization using its weights. Instead of representing the density field by a finite element mesh, it is represented by the activation functions of the NN. The NN

outputs a density value for each point of the domain thus converting the optimization setup from a constrained into an unconstrained problem with penalization. A fully connected NN is used that may treat 2D and 3D structures and outputs a value p which is the density value at any point. The loss function needed to train the neural network is given by the following expression:

$$L(w) = \frac{u^T K u}{J^0} + a \sum_e \left(\frac{\rho_e \upsilon_e}{V^*} - 1 \right)^2 \tag{10.16}$$

where α is a penalty parameter and J^0 is the initial compliance of the system. The penalty term α is progressively increased with each iteration. Compared with SIMP, the output is similar in terms of compliance but the methods require half the computational cost of the conventional SIMP algorithm (Andreassen et al. 2011).

In Qian and Ye (2021), a dual neural network is used to solve the problem. The first network is used for forward calculation and the second one to perform the sensitivity analysis. The two networks are integrated with SIMP to replace the conventional Finite Element Analysis. The results of the proposed method are tested in two benchmark tests: minimum compliance design and metamaterial design. The architecture of both neural networks is fully connected, where the neural network used for the sensitivity analysis has the inverse architecture from the one used for the forward calculation. The loss function for both models consists of two parts:

$$\text{Loss} = L_1[x_1^3 - (x_1^3)_0]^2 + L_2 \sum_{i=1}^{3} \left(\frac{\partial x_1^3}{x_1^1} - \left(\frac{\partial x_1^3}{x_1^1} \right)_0 \right)^2 \tag{10.17}$$

To further improve the forward model, a convolutional neural network is used that contains the structure and the loading conditions. The architecture of the CNN is compact as it contains only 2 convolutional layers followed by 7 dense layers. This network takes as an input a 2-channel image, where one channel contains the density distribution of the structure and the second one contains the force distribution and outputs the compliance of the structure. The results of the system described above with images of size 64×64 are 137 times faster for the forward calculations and 74 times faster in sensitivity analysis. The authors show that a small dataset containing only 2000 training data suffices to achieve a 95% accuracy.

In Zhang et al. (2021) a physics informed neural network is used to perform the optimization, where the density of the elements is calculated by the reparameterization of the weights of the neural network. The main idea is this work is that a well-trained neural network can reconstruct an image based on a portion of it, where the image that is reconstructed is the final structure of the design variables. To achieve this, the mechanical properties and physics are introduced into the loss function of the neural network. The method is divided into four parts: *Neural reparameterization*, *Design constraints*, *Applied physics model,* and *Calculation of loss function.* In neural reparameterization, the neural network takes as an input the density of

the elements. The size of the input depends on the discretization of the mesh. The density is considered as a dependent variable and the NN weights are the independent variables. The second part requires converting the density outputted by the NN to physical density corresponding to the physical constraints set in the formulation of the problem. Filtering is used to achieve this. The third part is to perform Finite Element Analysis on each iteration after obtaining the structure topology. The final step of the process is to calculate the loss function to be minimized. In traditional SIMP or BESO, the derivatives of the objective function need to be calculated to perform sensitivity analysis. However, in this case, these are directly calculated via automatic differentiation during the backpropagation step.

The architecture that the neural network is using is a decoder network starting with a fully connected layer to linearly transform features from one space to another. This is then followed by sequential up sampling and convolutional layers. Furthermore, a direct copy of the input to the output is implemented, the same as the ones U-nets use. An important step after the calculation of the density from the neural network is to convert it to physical density according to the definition of the problem. Conventional methods are not suffering from this problem because the design constraints are considered when calculating the density. This conversion happens in two steps:

(1) Make $\boldsymbol{x}$ satisfy the [0, 1] constraint by applying the sigmoid transformation to the output layer:

$$x_i = \frac{1}{1 + e^{x_i - b(x,V)}} \tag{10.18}$$

(2) Eliminate intermediate density values by using "Projection" method:

$$x_{\text{phy}} = \frac{\tanh(\beta\eta) + \tanh(\beta(x - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta)} \tag{10.19}$$

where η is a threshold and β controls the sharpness of the projection. To ensure that the volume constraints are preserved when applying projection, a volume preserving Heaviside is added, i.e.:

$$\sum_{i=1}^{N} x_{\text{phy}} v_i = \sum_{i=1}^{N} x_i v_i \tag{10.20}$$

that denotes the volume before and after the projection. Solving Eq. (10.20) results in the value of η, which is used in the Projection equation. Higher values of β correspond to thinner branches that will be eliminated from the final structure, which also makes the manufacturing process easier. Compared with the SIMP method used in Andreassen et al. (2011) similar results are achieved in terms of the shape of the final structure. One big difference is that the structure of the neural network has larger and more rigid branches on the inside of the structure due to the projection step. The proposed method can also be used

in stress-constrained problems, structural natural frequency optimization problems, compliant mechanism design problems, heat conduction system design problems, and the optimization problem of hyperelastic structures. The big advantage of this method is that it does not need to construct a dataset apriori and that the final structure does not suffer from structural disconnection.

In Abueidda et al. (2020), a convolutional neural network is used to calculate the optimized designs without the use of conventional methods like SIMP, BESO, etc., thus achieving substantial speed up in the process. The proposed method works for linear and nonlinear materials; each material has its own CNN. A synthetic dataset is constructed, with each pair of optimized designs and their corresponding boundary conditions, loads and volume constraints. The architecture of the neural network is based on ResUnet which combines the benefits of semantic segmentation of Unet (Ronneberger et al. 2015) and residual learning of ResNet (Kollmann et al. 2020) to further improve the performance of the Unet. The Unet combines low-level extracted features with high-level information to further improve the accuracy of the segmentation. The deeper a neural network becomes, the harder it gets to avoid the vanishing gradient problem where the gradient becomes too miniscule to calculate. ResNet tackled this problem by using residual blocks. The entire network consists of three components: (i) Encoder: compresses the input image into a more compressed representation, (ii) Decoder: reconstructs the original image, and (iii) Direct transfer of the input to the output layer. Also, the convolutional layers contain skip connections (residual blocks) that apart from the vanishing gradient problem that they solve, also reduce the number of parameters that the network must use. The loss used during the training of the neural network is Mean Square Error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N} \| Net(I, W) - s_i \|^2 \tag{10.21}$$

Another metric used is the Dice Similarity Coefficient (DSC), which measures the similarity of the output image obtained from the neural network with the ground truth image. The DSC assumes a value of 1 if the two images are identical:

$$DSC = \frac{2\left|y \cap \underline{y}\right|}{|y| + \left|\underline{y}\right|} \tag{10.22}$$

After training the model, a simple threshold of 0.5 is applied to discretize the densities in $\{0, 1\}$. Both the linear and nonlinear models achieve robust $DSC = 0.958$ and $DSC = 0.964$ on the test and the train set, respectively. Since the method does not rely on any external FEA solver, it is very fast. By transferring the inference of the neural network to a lower-level hardware, the method can perform instantaneous optimization of structures for linear and nonlinear materials.

In Deng and To (2021), a new parametric method using deep learning is introduced, where the level set function is described by a deep neural network. The proposed method utilizes the ability of the deep neural networks to approximate any function, thus it can approximate the level set function too. A critical aspect for the convergence of the objective function during training is the initialization of the weights with random zero-mean values. The level set theory uses a zero contour (2D) or isosurface (3D) to represent the boundaries of geometry of the structure. The interface of the structure is described by the zero-level set functions:

$$\begin{bmatrix} \Phi(\boldsymbol{x}, \boldsymbol{t}) > 0, (\boldsymbol{x} \in \boldsymbol{\Omega}) \\ \Phi(\boldsymbol{x}, \boldsymbol{t}) = 0, (\boldsymbol{x} \in \partial\boldsymbol{\Omega}) \\ \Phi(\boldsymbol{x}, \boldsymbol{t}) < 0, (\boldsymbol{x} \in \boldsymbol{D}/\boldsymbol{\Omega}) \end{bmatrix} \tag{10.23}$$

where D is the design domain, Ω is the total number of admissible designs, $\partial\Omega$ the boundary of the shape, and t the pseudo time. By differentiating the zero-level set, the Hamilton–Jacobi partial differential equation (PDE) can be obtained:

$$\frac{\partial \Phi}{\partial t} - V_n |\nabla \Phi| = 0 \tag{10.24}$$

And the objective function is

$$\min : J(\Phi) = \int_D (\varepsilon(u) : C : \varepsilon(u)) H(\Phi) d\Omega \tag{10.25}$$

The deep neural network converts the PDE to an ordinary differential equation (ODE). Hence, instead of solving Hamilton–Jacobi equations to update $\Phi(x)$ and finding the optimal design, $\Phi(x)$ is represented by the parameters of the neural network. The neural network is trained using the values resulting from solving Hamilton–Jacobi, these values are used as the ground truth values. The resulting designs have similar structural performance with the traditional methods, while with different neural networks different conceptual designs can be produced.

In Patel et al. (2022) a method to overcome challenges that traditional topology optimization struggles with, such as geometric frustration, non-smooth edges, dangling structures at boundaries is introduced. In addition, the method accelerates the entire process. This method uses two deep neural networks, one that predicts the optimized microstructures and one that improves connectivity between them. The method has three stages:

- A macroscale topology optimization solver (SIMP) which predicts optimized macroscale topology optimizations. It takes as an input the finite elements in each direction, boundary conditions, Poisson ratio, Young's modulus, and optimization parameters.

- The second stage contains a deep learning neural network which predicts microstructures. This model takes as an input the density and nodal deflections of every macroscale unit of the previous step and outputs optimized microstructures.
- The third stage contains another deep neural network that improves the connectivity of the whole structure and outputs the final optimized structure.

The first neural network is trained using only corner displacement nodes rather than the whole domain, which makes the calculation of the microscale structures faster. Stage 1 features a modified density-based SIMP approach, having the objective to minimize the compliance using conventional methods. The second stage of a model predicts the optimized microstructures that fit well into the macrostructure. That model has 3 sub-models, one deep neural network that maps the design variable vector to a density distribution image, another convolutional neural network that predicts the optimized structure, and a third post-processing solver that ensures volume fractions constraints and optimal solutions. The third stage contains two neural networks that improve the connectivity of the predicted microstructures. The first neural network is a *UNet* that predicts an improvement in connectivity between 4 neighboring elements, and the second neural network uses the pre-optimized corners to predict the output to reduce the number of iterations. The connectivity of the optimized structure is improved by 17% by the third stage and an overall 14.6% improvement in compliance. Also, there is a great improvement in the speed of the calculation of the optimized structure by a factor of $\times 10$ faster than the conventional method. Also, the proposed method works in both 2D and 3D structures.

10.3.2 Image-Based Methods

In Banga et al. (2018), a 3D approach is explored where a convolutional neural network is used to calculate the final output of the structure. The idea is to train a neural network with enough degrees of freedom to directly map an input to its optimized structure output. The dataset used as ground truth are 3D images obtained from TopOpt and it is based on SIMP methodology. The dataset is sampled with different Volume fraction V_0, Number of Nodes N_L, Load direction vectors V_L, Load positions P_L , and Displacement Boundary Constraint B_C. The total number of samples generated with TopOpt is 6,000. Each sample took 70 to 100 iterations to converge in the conventional topology optimization process.

$$\text{Loss} = \frac{-1}{n}\left[\sum_{i=1}^{n} X^i_{\text{true}} log(X^i_{\text{pred}}) + (1 - X^i_{\text{true}}) log(X^i_{\text{pred}})\right] + \beta\left[\frac{1}{n}\sum_{i=1}^{n}(X^i_{\text{pred}} - X^i_{\text{true}})^2\right] \tag{10.26}$$

Three types of inputs are given to the neural network, which are (i) 3D density distribution of voxels at iteration m ($m < T$), (ii) Gradient of voxel densities between iterations m and n ($m < n$), and (iii) Forces and Boundary Conditions along x, y, and z directions, where T is the total number of iterations and m and n are intermediate numbers of iterations. Also at the output of the network a Density Filter Function was applied to smooth the output based on the neighbors of each voxel:

$$\text{Density Filter Function} : \underline{x} = \frac{\sum_{j \in N} h_{ij} v_j x_j}{\sum_{j \in N_j} h_j v_j} \tag{10.27}$$

The architecture used is a convolutional neural network without any dense layers, as it uses only 3D convolutional layers. Specifically, it follows an encoder–decoder architecture and the output of the neural network is the same as the input. The results are compared with standard linear elasticity solvers both in terms of accuracy and speed. Another hyperparameter that is finetuned is the number of iterations at which TopOpt is stopped, and it needs to be balanced in accuracy and number of iterations performed. The best neural network experiment from the ones that have been tried achieved 40% reduction in computational time and achieved 96% accuracy. The results show that the calculated compliances of the structures by the traditional method and the conventional method slightly differ. Also 4.82% of the samples have huge compliance errors due to emergence of the structural disconnection. The compliance error compared with the conventional method is 4.16% and volume fraction error is 0.13%.

In Sosnovik and Oseledets (2019), topology optimizations followed by a neural network are used to calculate the final structure. An initial number of conventional topology optimization iterations is obtained N_0 using SIMP, the output of SIMP is turned into an image I and used as an input to the neural network. Image I is a blurred/distorted representation of the final structure. If only topology optimization was performed the final structure contains only material and void with no intermediate values, this structure is represented by I^*. So, after performing N_0 steps image I is not the same as image I^*. Thus, neural networks are used to perform image segmentation to converge image I to image I^* and resulting in binary densities $\{0, 1\}$. The neural network architecture used in a fully connected convolutional neural network takes as an input 2 grayscale images, the first image is the densities X_n as outputs by the last step of topology optimization and the second image is the difference of the densities between 2 consecutive updates $\delta X = X_n - X_{n-1}$. The output of the network is a grayscale image of the same resolution that contains the final structure. The neural network follows the encoder–decoder architecture with 6 convolutional layers in the encoder layer and another 6 in the decoder layered, which are the same shape as the encoder network but reversed. Also, between the convolution layers Max Pooling operation is used to introduce variance to the next layer.

The dataset used is synthetic based on SIMP solver for 2D structures. For the generation of the dataset 100 iterations of SIMP are performed for each problem,

each individual problem is defined by its contains and load. To generate the dataset the following constraints are used:

- The number of nodes with fixed x and y translations and the number of load is sampled from the Poisson distribution with $N_x \simeq P(\lambda = 2)$, $N_y \simeq P(\lambda = 1)$
- The load values are -1 and the probability of choosing a boundary node is 100 times higher than that of an inner node
- Volume is sampled from normal distribution $f_0 \simeq N(\mu = 0.5, \sigma = 0.1)$

The total size of the dataset is 10,000 samples. Each sample consists of a tensor with shape $100 \times 40 \times 40$, where 100 is the number of iterations and 40×40 is the grid size. During the training data augmentation is applied to the data to increase the size of the dataset and the variation. The objective function used is:

$$L = L_{\text{conf}}(X_{\text{true}}, X_{\text{pred}}) + \beta L_{\text{vol}}(X_{\text{true}}, X_{\text{pred}}) \tag{10.28}$$

where L_{conf} is binary cross-entropy and L_{vol} is MSE of the prediction and the target. The results are compared against SIMP solver in terms of accuracy and time consumption. The metrics used are Binary Accuracy and Intersection over Union (IoU). Where Binary Accuracy measures the pixels classified correctly over the total number of pixels of the structure and IoU measures the area of overlap over the area of the union of the correctly classified pixels. Four different policies were tested using different stopping iterations for the SIMP algorithm. The number of iterations that SIMP stops is sampled from uniform distribution $U \sim [1, 100]$ and Poisson $\lambda = 5, 10, 30$. The output of the structure is similar to the one produced by SIMP and its calculation is 20 times faster. Higher Accuracy and IoU is achieved with more SIMP iterations, the highest one achieved is Accuracy $= 99.6\%$ and $IoU = 99.2\%$.

Another image-based approach is the study by Wang et al. (2022), where a convolutional neural network with strong generalization capabilities is used. The dataset used consists of 80,000 samples using (Andreassen et al. 2011) which uses SIMP. The volume fraction, number of forces, and direction of each force are sampled from uniform distribution. Input of the neural network is a tensor $40 \times 80 \times 6$ tensor, each one tensor contains an image of Volume fraction, Nodal displacement in X and Y directions, Nodal Normal Stains $\varepsilon_x, \varepsilon_y$ and Shearing γ_{xy}. The ground truth that is used for training is the optimized output of SIMP. The architecture of the neural network is an encoder–decoder where the encoder part reduces the size of the input gradually up to 8 times and the decoder part restores it and outputs it to its original size. Because the probability distribution of each element is between (0, 1) that denotes the probability of existence of the element, thus a suitable loss function is Kullback-Leible divergence which tries to minimize the distance between the know distribution and the output distribution:

$$D_{KL}(p||q) = \sum_x p(x) log \frac{p(x)}{q(x)} + \frac{\lambda}{2} \sum_i \theta_i^2 \tag{10.29}$$

where the first term is the loss function with $p(x)$ the ground truth distribution and $q(x)$ the neural network output. The second term of the loss function is the L_2 regularization term to reduce overfitting, where λ is the weight of the regularization term and θ is the network parameters. The difference compared to SIMP is similar, as only a 4.12% showed large compliance errors. Also, the neural network provides a huge speed up in calculation, about 99% faster calculation of the optimal design structure.

In Kollmann et al. (2020), deep learning is used to optimize 2D structures of metamaterials. The proposed method uses a convolutional neural network (CNN) and non-iteratively optimizes metamaterials for either maximizing the bulk modulus, maximizing the shear modulus, or minimizing Poisson's ratio that also include negative values. The data used for the training of the neural network are created by randomly sampling optimization parameters. These optimization parameters are a filter radius, a design constraint (volume fraction) and a design objective (maximum bulk modulus, maximum shear modulus, or minimum Poisson's ratio), and are sampled from uniform distribution. And then the optimized design is calculated using SIMP. The neural network follows the architecture of the encoder–decoder network and takes as an input 3 images, one for each optimization parameter described before and outputs an image which represents the optimized structure. More specifically it utilizes the ResUnet architecture, where the Unet part is utilized for the semantic segmentation of the image and the skip connections of the ResNet which help train more efficiently deep neural networks without the issue of the vanishing gradient (too small values of the gradient in very deep neural networks). The loss function used during the training of the model is MSE between the ground truth optimized image and the output of the model, also the Dice Similarity Coefficient is used, which denotes the similarity of 2 images. To train the model the dataset created is split into training, validation, and test set. The validation set is used during the training to ensure that the model is not overfit to the training. Then the model is evaluated with the test set, which the model has never "seen" before. The model achieves: $MSE = 0.007$ and $DSC = 0.97$, especially the similarity coefficient shows that the produced optimized design image is almost similar to the ground truth. Final step of the process is to apply a threshold of 0.5 to binarize the predicted densities, because the model produces densities ranging from [0, 1] but the desired final optimized design must have values in {0,1}.

In Chi et al. (2021), a large-scale solution is proposed without a loss in accuracy. The proposed method has three distinct features: A novel component that's being trained from previous iterations, A two-scale topology optimization method using a localized strategy, and A component that generates new data from actual physical simulations that constantly improves the machine learning models. In contrast with other methods that use deep learning, where the training of the neural networks happens before the optimization process. In the proposed method training happens online in 2 stages. One initial online training session and several online updates during the process. There are 4 key parameters that control that process N_j, N_F which controls the initial online training step and the frequency of online updating frequency. The other 2 parameters W_I, W_U are a window that controls how much

back in the data the neural network can use for training in the initial step and the update stage respectively. The process of the online initial training of the neural network starts by solving the traditional equation for $N_I + N_W - 1$, where during the training the network can "see" only W_I steps back. Using the trained model, the most computational expensive steps can be avoided (calculation of the state equations and sensitivity analysis). To ensure that the model stays accurate during the whole process, the weights of the model change with regular frequency by switching back to the regular method of solving the equations with the standard procedure and updating the model. To make the proposed framework efficient and scalable, a two-scale topology optimization setup is used, where a fine-mesh and a coarse-mesh are used separately in different stages. Fine-mesh is used to solve the state equations to collect new data and also the design variables update is performed there. On the coarse-mesh no design variables updates happen, but the state equation is solved at every step of the optimization on the stiffness distribution that is mapped from the fine-scale mesh. The architecture of the neural network used is a fully connected deep neural network (DNN) with 4 hidden layers and with 1000 neurons at each hidden layer. A notable addition to the architecture of the DNN is the Parametric Rectified Linear Unit (PReLU) which is a generalization of ReLU that also contains a learnable parameter α. The use of PReLU has been shown great performance in image recognition tasks (He et al. 2015, 2016):

$$\sigma(x) = \max(0, x) + \alpha \min(0, x) \tag{10.30}$$

Deep Neural networks are often called universal approximators because, with the right architecture, they can approach practically every function. In topology optimization the design variables are millions, so one model cannot have the capacity to scale and produce an accurate mapping from the input to the output, as the design variables increase. That is why a two-scale localized setup is used to ensure the scalability of the model. The proposed setup does not require as much memory for the calculations. Training examples are produced from the coarse-mesh discretizations, which are also enclosed in the fine-mesh ones. Also, the whole global design of the fine-mesh is not treated as an individual example but each element of the mesh is used individually as a training example. The results show that the localized training strategy is more efficient in terms of memory efficiency and scalability. To measure the accuracy of the models, the angle of deviation from the original sensitivity is used:

$$\theta_{\text{error}} = \arccos\left(\frac{G^T G}{\|G^T\| \|G\|}\right) \tag{10.31}$$

With sufficient training steps with the proposed method, θ_{error} approaches zero proving that the method is also accurate. It is also suggested that the strain vector instead of the nodal displacement vector from the coarse mesh should be used as an input for the deep neural network.

In Li et al. (2022), a cross-resolution method to map intermediate designs to final high-resolution designs is introduced, also the method works with geometrical non-linearities. Geometrical non-linearities occur when the relationship between displacement and strain becomes nonlinear. The dataset is constructed by calculating both the intermediate designs and the final high-resolution design of the cantilever beam and the short cantilever beam with forces chosen randomly from uniform distribution. The model used here consists of two neural networks and is based on the generator-discriminator architecture. A cross-resolution is used as a generator and a Markovian discriminator as a discriminator. The generator creates the high-resolution image of the output design and takes as an input the low-resolution intermediate image the discriminator is used during the training to distinguish between real and fake configurations. The generator consists of 3 components, a Unet to make the abstract connections from the low-resolution image to the higher one, a cross-resolution layer that expands the dimensions and keeps the number of channels the same and a ResNet which is added to achieve deeper neural networks and by using its skip connections the phenomenon of the vanishing gradient can be avoided. The discriminator used is a Markovian discriminator which is more suitable for image prediction problems compared to conventional CNN discriminators. Markovian discriminator uses a sliding kernel which produces a score across the image and the total mean score is used to judge if the image is real or fake, also the sliding kernel preserves the continuity and can reveal more detail. The similarity that the generator achieves is ~95% compared with the original image and reduces computational cost from ~300 s to ~17 s.

10.4 Conclusions

The present chapter presents a review of methods used to perform topology optimization using artificial intelligence-related methodologies. Artificial intelligence is a useful tool employed in many scientific areas during the last decades. It is no surprise that such tools have found their way to topology optimization problems either by completely replacing the conventional methods or by assisting the conventional methods to reduce the required computational cost. The main advantage of using artificial intelligence to perform topology optimization is that these models have a large enough learning capacity that can map the input to an output, even in complex engineering problems. As a result, by properly training the AI model, it can be used to map an input to an output during the topology optimization process.

There are many models and algorithms that have been developed during the most recent years for AI-assisted topology optimization problems. The two most popular families of methods are density based and image-based ones. Density-based methods use the mechanical properties of the model as an input and output a density. The second family, image-based methods, use an image as an input, either in 2D or 3D. Most methods use a deep neural network or a convolutional neural network to calculate the output of the optimized structure. Methods using density-based approaches

exhibit usually better performance both in terms of accuracy and computational cost reduction. Advancements both in software and hardware can improve even further the performance of these methods, as AI models rely on GPU computations.

Acknowledgements The research was supported by the National Funding for European Competitive Research Projects, for the financial year 2019, with the beneficiary being the National Technical University of Athens (NTUA), project: 67120100, GSRT AWARD 2019 "Data Driven Surrogate Models and Applications".

The fifth author acknowledges the support of the European Union's Horizon research and innovation programme under the Marie Sklodowska-Curie Individual Fellowship grant "AI2AM: Artificial Intelligence driven topology optimisation of Additively Manufactured Composite Components", No. 101021629.

References

Abueidda DW, Koric S, Sobh NA (2020) Topology optimization of 2D structures with nonlinearities using deep learning. Comput Struct 237:106283. https://doi.org/10.1016/j.compstruc.2020.106283

Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2011) Efficient topology optimization in MATLAB using 88 lines of code. Struct Multidisc Optim 43(1):1–16. https://doi.org/10.1007/s00158-010-0594-7

Banga S, Gehani H, Bhilare S, Patel S, Kara L (2018) 3D topology optimization using convolutional neural networks. arXiv:1808.07440v1. https://doi.org/10.48550/arXiv.1808.07440

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

Bendsøe MP, Sigmund O (2004) Topology optimization: theory, methods, and applications. Springer, Heidelberg. https://doi.org/10.1007/978-3-662-05086-6

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

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

Clarke MRB (1974) Pattern classification and scene analysis. J R Stat Soc: Ser A (general) 137(3):442–443. https://doi.org/10.2307/2344977

Deng H, To AC (2021) A parametric level set method for topology optimization based on deep neural network. J Mech Des 143(9). https://doi.org/10.1115/1.4050105

He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: 2015 IEEE international conference on computer vision (ICCV), pp 1026–1034. https://doi.org/10.1109/ICCV.2015.123

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778. https://doi.org/10.1109/CVPR.2016.90

Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366. https://doi.org/10.1016/0893-6080(89)90020-8

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

Kallioras NA, Lagaros ND (2020) DL-scale: deep learning for model upgrading in topology optimization. Procedia Manuf 44:433–440. https://doi.org/10.1016/j.promfg.2020.02.273

Kallioras NA, Lagaros ND (2021a) DL-SCALE: a novel deep learning-based model order upscaling scheme for solving topology optimization problems. Neural Comput Appl 33(12):7125–7144. https://doi.org/10.1007/s00521-020-05480-8

Kallioras NA, Lagaros ND (2021b) MLGen: generative design framework based on machine learning and topology optimization. Appl Sci 11(24):12044

Kallioras NA, Nordas AN, Lagaros ND (2021) Deep learning-based accuracy upgrade of reduced order models in topology optimization. Appl Sci 11(24):12005

Kollmann HT, Abueidda DW, Koric S, Guleryuz E, Sobh NA (2020) Deep learning for topology optimization of 2D metamaterials. Mater Des 196:109098. https://doi.org/10.1016/j.matdes.2020.109098

Lei X, Liu C, Du Z, Zhang W, Guo X (2018) Machine learning-driven real-time topology optimization under moving morphable component-based framework. J Appl Mech 86(1). https://doi.org/10.1115/1.4041319

Li J, Ye H, Yuan B, Wei N (2022) Cross-resolution topology optimization for geometrical nonlinearity by using deep learning. Struct Multidisc Optim 65(4):133. https://doi.org/10.1007/s00158-022-03231-y

Liu K, Tovar A, Nutwell E, Detwiler D (2015) Towards nonlinear multimaterial topology optimization using unsupervised machine learning and metamodel-based optimization. In: ASME 2015 international design engineering technical conferences and computers and information in engineering conference. https://doi.org/10.1115/detc2015-46534

Lu X, Plevris V, Tsiatas G, De Domenico D (2022) Editorial: artificial intelligence-powered methodologies and applications in earthquake and structural engineering. Frontiers Built Environ 8. https://doi.org/10.3389/fbuil.2022.876077

Mlejnek HP (1992) Some aspects of the genesis of structures. Struct Optim 5(1):64–69. https://doi.org/10.1007/BF01744697

Parzen E (1962) On Estimation of a probability density function and mode. Ann Math Stat 33(3):1065–1076

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

Patel J, Choi S-K (2012) Classification approach for reliability-based topology optimization using probabilistic neural networks. Struct Multidisc Optim 45(4):529–543. https://doi.org/10.1007/s00158-011-0711-2

Qian C, Ye W (2021) Accelerating gradient-based topology optimization design with dual-model artificial neural networks. Struct Multidisc Optim 63(4):1687–1707. https://doi.org/10.1007/s00158-020-02770-6

Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. Cham, pp 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Solorzano G, Plevris V (2022) Computational intelligence methods in simulation and modeling of structures: a state-of-the-art review using bibliometric maps. Frontiers Built Environ 8. https://doi.org/10.3389/fbuil.2022.1049616

Sosnovik I, Oseledets I (2019) Neural networks for topology optimization. Russ J Numer Anal Math Model 34(4):215–223. https://doi.org/10.1515/rnam-2019-0018

Wang D, Xiang C, Pan Y, Chen A, Zhou X, Zhang Y (2022) A deep convolutional neural network for topology optimization with perceptible generalization ability. Eng Optim 54(6):973–988. https://doi.org/10.1080/0305215X.2021.1902998

White DA, Arrighi WJ, Kudo J, Watts SE (2019) Multiscale topology optimization using neural network surrogate models. Comput Methods Appl Mech Eng 346:1118–1135. https://doi.org/10.1016/j.cma.2018.09.007

Zhang Z, Li Y, Zhou W, Chen X, Yao W, Zhao Y (2021) TONR: An exploration for a novel way combining neural network with topology optimization. Comput Methods Appl Mech Eng 386:114083. https://doi.org/10.1016/j.cma.2021.114083

Zhou M, Rozvany GIN (1991) The COC algorithm, Part II: Topological, geometrical and generalized shape optimization. Comput Methods Appl Mech Eng 89(1):309–336. https://doi.org/10.1016/0045-7825(91)90046-9