Graphical Abstract

Graded Multiscale Topology Optimization using Neural Networks
Aaditya Chandrasekhar, Saketh Sridhara, Krishnan Suresh
Highlights

Graded Multiscale Topology Optimization using Neural Networks
Aaditya Chandrasekhar, Saketh Sridhara, Krishnan Suresh

- A novel neural network representation for multiscale topology optimization.
- It guarantees partition of unity and discourages microstructure mixing.
- It efficiently handles numerous pre-selected microstructures.
- Number of design variables is only weakly dependent on the number of microstructures.
Graded Multiscale Topology Optimization using Neural Networks

Aaditya Chandrasekhar\textsuperscript{a,**}, Saketh Sridhara\textsuperscript{a,**}, Krishnan Suresh\textsuperscript{a,*}

\textsuperscript{a}Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, 53706, WI, United States

Abstract

In this paper, we propose a novel graded multiscale topology optimization framework by exploiting the unique classification capacity of neural networks. The salient features of this framework include: (1) the number of design variables is only weakly dependent on the number of pre-selected microstructures, (2) it guarantees partition of unity while discouraging microstructure mixing, (3) it supports automatic differentiation, thereby eliminating manual sensitivity analysis, and (4) it supports high-resolution re-sampling, leading to smoother variation of microstructure topologies. The proposed framework is illustrated through several examples.

Keywords: multiscale topology optimization, graded microstructure, neural networks, automatic differentiation

1. Introduction

Topology optimization (TO) is a strategy for optimally distributing material within a design domain ([1, 2]) to maximize a desired objective, while meeting one or more constraints. Various TO methods have been proposed; these include density-based methods([3]), homogenization ([4, 5]), level-set ([6, 7]), evolutionary ([8, 9]), morphable components ([10]), and topological sensitivity based methods ([11],[12]). These methods typically lead to a single-scale design where all geometric features are of similar size. As an example, for the cantilever problem posed in 1(a), a single-scale TO that min-
imizes the compliance, subject to a volume constraint, leads to the topology in 1(b).

Figure 1: TO problem and solutions: (a) Mid-cantilever problem. (b) Single-scale solution. (c) Multiscale solution. (d) Graded multiscale solution using a single X-shaped microstructure.

In contrast, in multiscale topology optimization (M-TO), geometric features are computed at two or more length scales ([13],[14]). For the problem in fig. 1(a), fig. 1(c) illustrates a topology, based on a two-scale M-TO ([15]). Such designs exhibit unique characteristics such as large surface-to-volume ratio, excellent resilience, etc. This has led to a wide range of applications ([16, 17]) including energy absorption ([18]), heat exchangers ([19]), and biomedical applications ([20]). Current M-TO methods include clustering ([21, 15, 22]), kriging ([23]), multi-material ([24]), conformal lattices ([25]), etc. The advent of additive manufacturing has further spurred research in M-TO ([16]).

1.1. Graded Multiscale TO

However, one of the challenges in M-TO is the high computational cost since one must generate and evaluate various microstructures (through homogenization) during each step of the optimization process ([13, 21, 15, 26]). To reduce this cost, researchers have proposed the use of graded variations of one or more pre-selected microstructural topologies ([14, 19, 27]). [14] expressed the mechanical properties as a density function, with a B-spline-based interpolation. [19] designed graded cellular structures with triply periodic shapes. [27] parameterized the radii of each member of a lattice with linear and sinusoidal graded variations. These methods are referred to as graded multiscale TO, or GM-TO, leading to topologies such as the one illustrated in fig. 1(d), where, as an example, graded variations of a single X-shaped microstructure are used everywhere. This leads to a significant reduction in computational cost ([28]) since homogenization is not needed within the optimization loop. However, restricting the design to a single microstructure
can significantly reduce performance ([29]). To further improve the performance of GM-TO, without substantially increasing the computational cost, one can use a finite number of graded microstructures.

**Density-based methods:** Theoretically, using a finite number of microstructures in GM-TO is analogous to using multiple materials in TO. Therefore, various authors ([30, 31, 32, 33]) proposed the use of uniform multi-phase materials interpolation (UMMI) models for GM-TO. Given the elasticity matrix of each microstructure at a particular volume fraction, two different UMMI models have been proposed for interpolation [34, 35]. The first type (UMMI-1) proposed by [31, 33], simply adds the contributions from each microstructure, and does not penalize microstructure mixing. The UMMI of the second type (UMMI-2) penalizes microstructure mixing at the cost of non-linearity ([30]). To mitigate the effect of non-linearity, [36] suggested a gradual increase in the penalty in multi-materials, further explored as a multi-scale formulation by [34, 35]. Despite penalization, microstructure mixing is not entirely eliminated in UMMI-2. Further, while multiple materials can often co-exist at a given point, this is unacceptable in GM-TO. Finally, a handful (typically less than five) materials can be sufficient in multi-material TO, while a large number of microstructures are required in GM-TO.

**Level-set methods:** Level-set methods are also a popular choice for GM-TO. [39, 40] considered a level-set based method where the microstructural shape is represented and parametrized by implicit functions thereby circumventing the need for homogenization within every loop; instead one can rely on curve-fitting. [41] utilized the level set function to evolve on topologies on both the micro and macroscale, with connectivity ensured by the higher-order continuity of the level-set function. [42] addressed the challenge of connectivity of microstructures through shape metamorphosis to build graded transition zones using a connectivity index. Finally, GM-TO with graded variations of a single microstructure was implemented by [43] based on the parametrized level-set functions using radial basis functions.

**Data-driven methods:** Recently, data-driven techniques have been proposed to address GM-TO. For instance, [44, 45] utilize a latent variable Gaussian process that embeds discrete microstructures in a continuous and differentiable latent space. While the latent space is trained on the discrete homogenized data, the procedure of snapping and re-optimization may lead to sub-optimal results. Several authors ([46, 47, 48]) replaced the expensive homogenization process with neural networks that map the design variables.
to the homogenized elasticity matrix. [33] used neural networks to interpolate
the stiffness matrices from homogenized data. [49] used two neural networks
trained on topology optimized data: one for determining the microstructural
topology and the other to improve the connectivity of microstructures. One
of the challenges in data-driven methods is the computational expense in
data generation and training of the machine learning models. Another chal-
lenge with the data-driven methods is that physical bounds on the elasticity
matrices (such as positive-definiteness) cannot be enforced explicitly, which
can lead to spurious stiffness matrices and non-convergence.

1.2. Paper Contributions

In this paper, we exploit the unique classification capability of neural net-
works to address some of the limitations of current GM-TO methods. Specif-
ically, we propose a graded multiscale topology optimization using neural-
networks (GM-TOuNN) (section 2) framework. GM-TOuNN extends the
mesh-independent neural-network (NN) based representation of the macro-
scale topology proposed by [50] to GM-TO. The mesh-independent repre-
sentation allows us to consider a large number of candidate microstructures
without increasing the number of design variables. Further, the partition of
unity is implicitly guaranteed by the NN construction. Finally, one can lever-
age automatic differentiation (AD) of the NN computational framework to
provide end-to-end differentiability and automated sensitivity analysis. Nu-
merical experiments are presented in section 3. We conclude with limitations
and future work in section 4.

2. Proposed Method

2.1. Problem Specification

Consider a design domain with prescribed loads, restraints and a set of
microstructural topologies (fig. 2). In the current work, we seek to com-
pute an optimal multiscale design where we determine, at every location, the
appropriate microstructure and its size-parameter (gradation). For simplic-
ity, we will assume the compliance must be minimized subject to a volume
constraint.

To assist in optimization, we introduce the following design variables. The
presence or absence of a microstructure \( m \) at any point \( \mathbf{x} \) will be denoted by
the variable \( \rho_m(\mathbf{x}) \), where, ideally, \( \rho_m(\mathbf{x}) \in \{0, 1\} \). However, for continuous
gradient-based optimization, we will let \( 0 \leq \rho_m \leq 1 \) and drive it towards
0/1 through penalization. Thus, at any point \( x \), one can define the vector \( \rho(x) = \{\rho_1(x), \rho_2(x), \ldots, \rho_M(x)\} \) that captures the presence or absence of the \( M \) microstructures with the \textit{partition of unity} constraint \( \sum \rho_m(x) = 1 \).

Further, since the microstructures are graded, we control their size by the scalar design variable \( 0 \leq v(x) \leq 1 \) where 0 denotes a void and 1 denotes a complete fill. Thus, in conclusion, \( \rho(x) \) dictates the type of microstructure, while \( v(x) \) dictates the size or volume fraction of the microstructure at \( x \).

Consequently, one can pose the GM-TO problem in a discrete finite-element setting as:

\[
\begin{align*}
\text{minimize} & \quad J(\rho, v) = f^T u \\
\text{subject to} & \quad K(\rho, v) u = f \\
& \quad g_v(v) : = \frac{\sum e v(x_e) A_e}{V_f^* \sum e A_e} - 1 \leq 0 \\
& \quad \sum_{m=1}^{M} \rho_m(x) = 1, \ \forall x \\
& \quad 0 \leq \rho_m(x) \leq 1, \ \forall x, \ \forall m \\
& \quad 0 \leq v(x) \leq 1, \ \forall x
\end{align*}
\]

where \( J \) is the structural compliance, \( K \) is the finite element stiffness matrix, \( u \) is the displacement field, \( f \) is the applied force, \( V_f^* \) the maximum allowed volume fraction, with \( A_e \) is the area (2D) of the element \( e \).

\subsection*{2.2. Design Representation using Neural Networks}

Typically, for such problems, the design variables are captured via the underlying FE mesh ([51]), i.e., for the above problem, design variables \( \rho_e \).
and \( v_e \) are defined at each element \( e \). Thus the number of design variables grows linearly with the mesh size. Further, observe that the partition of unity constraint must be imposed over each element.

In this paper, we avoid this undesirable growth in complexity by indirectly controlling the design variables via a coordinate-based neural-network ([50, 52]). The proposed neural-network (NN) architecture (see fig. 3) consists of the following entities:

1. **Input Layer**: The input to the NN are points \( \mathbf{x} \in \mathbb{R}^d \) (\( d = 2 \) for 2D) within the domain.

2. **Hidden Layers**: The hidden layers consists of a series of Swish \((x \cdot \text{sigmoid}(x), [53])\) activated dense layers.

3. **Output Layer**: The output layer consists of \( M + 1 \) neurons corresponding to the microstructure composition \( \mathbf{\rho} = \{\rho_1, \rho_2, \ldots, \rho_M\} \) and the volume fraction \( v \). Further, the neurons associated with \( \mathbf{\rho} \) are activated by a softmax function. This guarantees that the partition of unity (\( \sum \rho_i = 1 \)) and physical validity \( 0 \leq \rho_i \leq 1 \) are automatically satisfied. The output neuron associated with volume fraction is activated via a Sigmoid function, ensuring that \( 0 \leq v(\mathbf{x}) \leq 1 \). Thus, no additional constraint are needed.

4. **Design Variables**: The weights and bias, denoted by the \( \mathbf{w} \), now become the primary design variables, i.e., we have \( \mathbf{\rho}(x, y; \mathbf{w}) \) and \( v(x, y; \mathbf{w}) \).

![Neural network architecture used for GM-TOuNN.](image)

Figure 3: Neural network architecture used for GM-TOuNN.
Thus the strategy is to perform GM-TO via the NN weights \( w \), i.e., the GM-TO problem in eq. (1) reduces to:

\[
\begin{align*}
\text{minimize} & \quad J(w) = f^T u(w) \\
\text{subject to} & \quad K(w) u = f \\
& \quad g_v(w) := \frac{\sum_e v(x_e; w) A_e}{V_f \sum_e A_e} - 1 \leq 0
\end{align*}
\] (2a)

Observe that: (1) no additional constraint is needed since they are automatically satisfied by the NN, and (2) increasing the number of candidate microstructures only increases the number of output neurons but not the number of design variables.

2.3. Material Model

Given the NN-architecture, one can now proceed to construct the material model for analysis. Towards this end, let \([C_m(v)]\) be the elasticity matrix of microstructure \( m \) at volume fraction \( v \), where \([C_m(v)]\) consists of six independent components (in 2D):

\[
[C_m(v)] = \begin{bmatrix} C_{11}(v) & C_{12}(v) & C_{13}(v) \\ C_{22}(v) & C_{23}(v) & \text{sym.} \\ C_{33}(v) & \end{bmatrix}
\] (3)

To obtain \([C_m(v)]\), we adopt a simple constrained polynomial scheme ([54]). Specifically, the homogenized constitutive matrix [55] for microstructure \( m \) is evaluated at a few instances of volume fractions. Then, a polynomial ensuring positive definiteness is interpolated to these instances for each of the components. As an example, the polynomials for an X-type microstructure are illustrated in fig. 4.

Given \([C_m(v)]\) for each microstructure \( m \), the effective elasticity matrix at any location \( x \) is defined here as a weighted average ([34]):

\[
[C(\rho, v)] = \sum_{m=1}^M \rho_m^p [C_m(v)]
\] (4)

where \( p \) is the solid isotropic material with penalization (SIMP) constant. The penalization discourages intermediate volume fractions, i.e., discourages
2.4. Finite Element Analysis

We will use conventional finite element analysis (FEA) as part of the framework. Here, we use a structured quadrilateral mesh due to its simplicity. The element stiffness matrix is defined as:

\[
[K_e] = \int_{\Omega_e} [\nabla N_e]^T [C(x_e)] [\nabla N_e] d\Omega_e \tag{5}
\]

where \([\nabla N_e]\) is the gradient of the shape matrix, and \([C(x_e)]\) is the elasticity tensor evaluated at the center of the element.

In a straightforward, but naïve, implementation, one would evaluate \(\rho\) and \(v\) at the center of each element via the NN. Then, given the pre-computed polynomials \([C_m(v)]\), one would then find the effective \([C]\) for that element via eq. (4). Finally the element stiffness matrices will be computed via eq. (5). Thus the element stiffness matrices must be computed for each element, and for each step of the optimization process. This can be computationally very expensive. To significantly reduce the computation, we exploit the concept of template stiffness matrices proposed by [15].

Observe that one can express the element stiffness matrix as follows:

\[
[K_e] = C_{11}\hat{K}^1 + C_{22}\hat{K}^2 + C_{33}\hat{K}^3 + C_{12}\hat{K}^4 + C_{13}\hat{K}^5 + C_{23}\hat{K}^6 \tag{6}
\]
where,
\[
\hat{K}_i = \int_{\Omega_e} [\nabla N_e]^T [\hat{C}_i] [\nabla N_e] d\Omega_e, \quad i = 1, 2, \ldots, 6
\] (7)

and
\[
\hat{C}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \hat{C}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \hat{C}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \hat{C}_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \hat{C}_5 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \hat{C}_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}
\] (8)

Thus the matrices \(\hat{K}_i\) are computed at the start of the optimization.

Then, during optimization, the element stiffness matrices are computed efficiently via eq. (6). In other words, one would evaluate \(\rho\) and \(v\) at the center of each element via the NN. Then, given the pre-computed polynomials \([C_m(v)]\), one would then find the effective \([C]\) for that element (as before). But, to compute the element stiffness matrices, we will rely on eq. (6) This will be followed by the assembly of the global stiffness matrix \(K\).

2.5. Loss Function

We now consider solving the NN-based optimization problem in eq. (2). Neural networks are designed to minimize a loss function using well-known optimization techniques such as Adam procedure ([56]). We therefore convert the constrained minimization problem into a loss function minimization by employing a log-barrier scheme as proposed in [57]. Specifically, the loss function is defined as

\[
L(w) = J + \psi(g_v)
\] (9)

where,
\[
\psi_t(g) = \begin{cases} 
-\frac{1}{t} \log(-g), & g \leq \frac{-1}{t} \\
tg - \frac{1}{t} \log\left(\frac{1}{t}\right) + \frac{1}{t}, & \text{otherwise}
\end{cases}
\] (10)

where the parameter \(t\) is updated during each iteration, making the enforcement of the constraint stricter as the optimization progresses.
2.6. Sensitivity Analysis

Since Adam is a gradient-based optimizer, it requires sensitivities, i.e., derivative of the loss function (eq. (9)) with respect to the design variable (weights of the NN $w$). Fortunately, one can exploit modern automatic differentiation frameworks ([58]) to avoid manual sensitivity calculations. In particular, expressing all our computation, including FEA within JAX ([59]), results in an end-to-end differentiable framework, as illustrated in fig. 5.

2.7. Algorithm

We summarize the proposed framework in algorithm 1. We will assume that a GM-TO problem, a desired volume fraction, an NN configuration and a set of microstructures with their interpolated coefficients are given.

The first step in the algorithm is to discretize the domain with a finite element mesh, and sample the mesh (line 3) at the center of each element (these serve as inputs to the NN). We compute the template stiffness matrices (line 4). The log barrier penalty $t$ and SIMP penalty parameter are also initialized (line 5).

In the main iteration, the element volume fraction $v$ and microstructure field $\rho$ are computed using the NN using the current values of $w$ (line 7). These fields are then used to construct the stiffness matrix and to solve for the displacement (line 8 - line 12). Then we compute the objective (line 13) and volume constraint (line 14), leading to the loss function (line 15). The weights $w$ are then updated using the Adam optimization scheme (line 17). The optimizer requests the sensitivities which are computed in an automated fashion (line 16). Finally the log-barrier penalty and SIMP penalty parameters are updated (line 19 and line 20). The process is repeated until termination, i.e., till the relative change in loss is below a certain threshold or

![Figure 5: Optimization loop of the proposed GM-TOuNN framework.](image-url)
the iterations exceed a maximum value. Upon termination, each element is replaced with an image of the associated microstructure with the desired volume fraction. The framework is schematically depicted in fig. 5.

### Algorithm 1 Graded Multiscale TO

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>procedure GMTOuNN($\Omega_0$, BC, $V_f^*$, ...)</td>
</tr>
<tr>
<td>2:</td>
<td>$\Omega^0 \rightarrow \Omega_h^0$</td>
</tr>
<tr>
<td>3:</td>
<td>$x = {x_e; y_e}_{e \in \Omega^0_h}$</td>
</tr>
<tr>
<td>4:</td>
<td>$\Omega_h^0 \rightarrow [K]$</td>
</tr>
<tr>
<td>5:</td>
<td>$k = 0; t = t_0; p = 1$</td>
</tr>
<tr>
<td>6:</td>
<td>repeat</td>
</tr>
<tr>
<td>7:</td>
<td>$NN(x_e; w) \rightarrow {\rho, v}_e$</td>
</tr>
<tr>
<td>8:</td>
<td>$v \rightarrow [C_m] m = 1, \ldots, M \forall e$</td>
</tr>
<tr>
<td>9:</td>
<td>${[C_m]; \rho_m}_e \rightarrow [C]_e$</td>
</tr>
<tr>
<td>10:</td>
<td>${[C], [K]}_e \rightarrow [K]_e$</td>
</tr>
<tr>
<td>11:</td>
<td>$\bigcup_e [K]_e \rightarrow [K]$</td>
</tr>
<tr>
<td>12:</td>
<td>${[K], f} \rightarrow u$</td>
</tr>
<tr>
<td>13:</td>
<td>${f, u} \rightarrow J$</td>
</tr>
<tr>
<td>14:</td>
<td>${v_e, V_f^*}_e \rightarrow g_v$</td>
</tr>
<tr>
<td>15:</td>
<td>${J, g_v}_e \rightarrow L$</td>
</tr>
<tr>
<td>16:</td>
<td>$AD(L \leftarrow w) \rightarrow \nabla L$</td>
</tr>
<tr>
<td>17:</td>
<td>$w + \Delta w(\nabla L) \rightarrow w$</td>
</tr>
<tr>
<td>18:</td>
<td>$k + +$</td>
</tr>
<tr>
<td>19:</td>
<td>$t = t_0\mu^k$</td>
</tr>
<tr>
<td>20:</td>
<td>$p = p + \Delta p$</td>
</tr>
<tr>
<td>21:</td>
<td>until $|\Delta L| &lt; \epsilon^*$ and $k &lt; k_{max}$</td>
</tr>
<tr>
<td>22:</td>
<td>return $w, \rho, v$</td>
</tr>
<tr>
<td>23:</td>
<td>end procedure</td>
</tr>
</tbody>
</table>

### 3. Numerical examples

In this section, we conduct several experiments to illustrate the method and algorithm. The default settings are as follows:

- **Mesh**: A mesh size of $60 \times 30$ with elements of size $1 \times 1$ is used for all experiments, unless otherwise stated; the force is assumed to be 1 unit.
• **Material**: The microstructures are assumed to be composed of an isotropic material with a Poisson’s ratio $\nu = 0.3$ and Young’s modulus $E = 1$.

• **Neural Network**: The NN comprises of 4 Swish-activated hidden layers with 20 neurons in each layer.

• **Candidate Microstructures**: A set of 11 predefined microstructures (fig. 6) are used in the experiments. Observe that connectivity between these microstructures is guaranteed. A quintic polynomial is used to interpolate the components of the constitutive matrix.

• **Material Penalization**: The SIMP penalization constant $p$ in eq. (4) is incremented every iteration by 0.02 using the continuation scheme [60], starting from a value of 1, capped at 8.

• **Loss Function**: The constraint penalty values of $t_0 = 1$ and $\mu = 1.01$ is used.

• **Optimizer**: Adam optimizer with a learning rate of 0.01 is used. Further, the gradients are clipped at a maximum norm of 1 to improve stability of convergence ([61]). A maximum of 300 iterations were allowed with $\epsilon^* = 0.01$.

• **Computing Environment**: All experiments are conducted on a Mac-Book M1 Pro, using the JAX ([59]) environment.

Figure 6: Microstructures considered in the experiments.

3.1. Validation

Consider the simple tensile bar problem in fig. 7a. Although eleven microstructures were provided (see fig. 6), the final topology in fig. 7b consists of a single microtopology (number 1) with a complete fill for all elements in the middle and void at the top and bottom, i.e., the result is consistent.
with expectations. The optimization converged in 240 iterations, taking 47 seconds.

Next, we consider an MBB beam in fig. 8a. [54] reported compliance values between 76 and 88 when considering only one microstructure at a time. In comparison, we achieve a compliance of 68 when all microstructures are allowed; see fig. 8b.

Next, we consider an L-bracket (800 elements) illustrated in fig. 9. Using only a single X-type microstructure, [54] achieved a compliance of 1130, 283, 171 for volume fractions of 0.1, 0.3 and 0.5 respectively. By considering all microstructures (fig. 10), we achieved compliance of 1048, 263 and 158 respectively.

Finally, we consider an edge-loaded beam with a mesh of 32×20 elements illustrated in fig. 11a. Here, the design is constrained to have the same microstructure along the X-axis (but can vary along the Y-Axis), with a desired volume fraction $V_f^* = 0.6$. The topology and compliance obtained (fig. 11c) is validated against the one reported by [15] (fig. 11b).
(a) Loading of an MBB beam.

Figure 8: GM-TO of an MBB beam.

(b) Optimized MBB beam.

Figure 9: Loading of an L-bracket
Figure 10: GM-TO of an L-bracket at various volume fractions.

Figure 11: Validation for an edge-loaded beam.

(a) Edge-loaded beam.

(b) M-TO design ($J = 1.45e5$) obtained by [15].

(c) GM-TO design ($J = 1.41e5$) obtained using the proposed method.
3.2. Convergence

We illustrate the typical convergence of the proposed algorithm for a mid-cantilever beam (fig. 1(a)) for $V_f^* = 0.5$. The compliance, volume fraction and the evolving topology at various instances are illustrated in fig. 12. We observe that the log-barrier formulation leads to a stable convergence. Similar convergence behavior was observed for all other examples. The computation took 53 secs.

Figure 12: Convergence of compliance and volume fraction for a mid-cantilever beam. Topologies are illustrated at the 0th, 50th, 100th, 150th, 200th and 250th (final) iterations.

3.3. Varying number of microstructures

Two central hypotheses of the current work is that one can achieve better designs with larger number of candidate microstructures, and the framework is (computationally) insensitive to the number of candidates. To validate these, consider the problem in fig. 8a. The topologies and compliance values when allowing for varying number of microstructures, in the order in which they are present in fig. 6, are illustrated in fig. 13, As expected, the compliance improves as we allow for larger number of microstructures. As noted earlier, each additional candidate microstructure requires only one additional output neuron (2 additional design variables), enabling us to consider large set of microstructures if necessary. The computational time was approximately 56 seconds, independent of the number of microstructures.
3.4. Pareto Designs

Exploring the Pareto-front is critical in making design choices and understanding the trade-off between the objective and constraint. Consider once again the mid-cantilever beam in fig. 1(a). We illustrate the compliance values and topologies for varying volume fractions $V^*_f$ in fig. 14.

3.5. Mesh and NN Dependency

Next we demonstrate the effect of NN size and FE mesh on the computed topology using the tip cantilever (fig. 2) as an example; all other parameters are kept constant. For varying NN size, the topologies are reported in fig. 15; no appreciable difference in computational time was observed. Similarly, fig. 16 captures the topologies for varying mesh size; the computational times were 7.5, 24.7 and 114.3 seconds per 100 iterations for the 40 × 20, 60 × 30 and 80 × 40 mesh respectively.
3.6. High Resolution Design

In the proposed method, one can perform optimization on a coarse mesh and then re-sample at a higher resolution to populate the microstructures, resulting in smoother variation in the microstructure topologies. Note that, even after re-sampling, the framework guarantees partition of unity. However, due to NN-interpolation, microstructure mixing can occur on the finer mesh at the interfaces. This was resolved by choosing the microstructure \( m \) with the largest \( \rho_m \). This is illustrated in fig. 17 where we optimize on a 40 \( \times \) 20 mesh and then re-sample on a 80 \( \times \) 40 mesh.
4. Conclusions

A framework for graded multi-scale topology optimization using neural network was proposed and demonstrated. The salient features of this framework are: (1) the number of design variables are only weakly dependent on the number of pre-selected microstructures, (2) the partition of unity constraint is automatically satisfied, and (3) manual sensitivity calculations is avoided.

The framework was limited to 2D compliance minimization problems, involving microstructures governed by a single size parameter. Extension to 3D, non-compliance problems (such as energy absorption [62], orthopedic implants [63], resonant frequencies [64]) using more generic multi-parameter microstructures need to be explored. Furthermore, while we relied on polynomials for directly interpolating the elasticity components, it is desirable to consider an Eigen-value decomposition ([54]) for increased robustness. The framework can complement and might benefit from data driven approaches ([44, 45]). Finally, while we relied on the simple Adam optimization procedure, second order optimization methods such as LBFGS might result in better/faster convergence ([65]).

Acknowledgements

The authors would like to thank the support of National Science Foundation through grant CMMI 1561899, and the U. S. Office of Naval Research under PANTHER award number N00014-21-1-2916 through Dr. Timothy Bentley.

Replication of Results

The Python code pertinent to this paper is available at https://github.com/UW-ERSL/GMTOuNN

Compliance with ethical standards

The authors declare that they have no conflict of interest.
References


