INTEGRATING MATERIAL SELECTION WITH DESIGN OPTIMIZATION VIA NEURAL NETWORKS

Aaditya Chandrasekhar [*], **Saketh Sridhara** [*], **Krishnan Suresh** [†]
Department of Mechanical Engineering, University of Wisconsin-Madison
{achandrasek3, ssridhara, ksuresh}@wisc.edu

September 19, 2022

ABSTRACT

The engineering design process often entails optimizing the underlying geometry while simultaneously selecting a suitable material. For a certain class of simple problems, the two are separable where, for example, one can first select an optimal material, and then optimize the geometry. However, in general, the two are not separable. Furthermore, the discrete nature of material selection is not compatible with gradient-based geometry optimization, making simultaneous optimization challenging.

In this paper, we propose the use of variational autoencoders (VAE) for simultaneous optimization. First, a data-driven VAE is used to project the discrete material database onto a continuous and differentiable latent space. This is then coupled with a fully-connected neural network, embedded with a finite-element solver, to simultaneously optimize the material and geometry. The neural-network's built-in gradient optimizer and back-propagation are exploited during optimization.

The proposed framework is demonstrated using trusses, where an optimal material needs to be chosen from a database, while simultaneously optimizing the cross-sectional areas of the truss members. Several numerical examples illustrate the efficacy of the proposed framework. The Python code used in these experiments is available at github.com/UW-ERSL/MaTruss

***K*eywords** Material selection · Truss optimization · Ashby charts · Neural networks · Autoencoder

1 Introduction

In engineering design, we are often faced with the task of optimizing the underlying geometry and selecting an optimal material [1]. As a simple example, consider the truss design problem illustrated in fig. 1 where one must optimize the cross-sectional areas, while selecting, for simplicity, a single material for all the truss members. Formally, one can pose this as [2, 3]:

$$\underset{\boldsymbol{A}=\{A_1,A_2,\ldots,A_N\},m\in M}{\text{minimize}} \quad \phi(\boldsymbol{A},\zeta_m) \tag{1a}$$

$$\text{subject to} \quad g(\boldsymbol{A},\zeta_m)\leq 0 \tag{1b}$$

$$K(\boldsymbol{A},E_m)\boldsymbol{u}=\boldsymbol{f} \tag{1c}$$

$$\boldsymbol{A}_{min}\leq\boldsymbol{A}\leq\boldsymbol{A}_{max} \tag{1d}$$

where ϕ is the objective (for example, compliance), g denotes a set of constraints (such as yield stress and buckling), $\boldsymbol{f}$ is the applied force, K is the truss stiffness matrix, and $\boldsymbol{u}$ is the nodal displacements. The design variables are the cross-sectional areas of the members $\boldsymbol{A}=\{A_1,A_2,\ldots,A_N\}$, with limits $\boldsymbol{A}_{min}$ and $\boldsymbol{A}_{max}$, and the material choice $m\in M$. We denote the collection of relevant material properties by ζ_m; these may include, for example, the Young's modulus (E_m), cost per unit mass (C_m), mass density (ρ_m) and yield strength (Y_m).

[*]Contributed equally
[†]Corresponding author

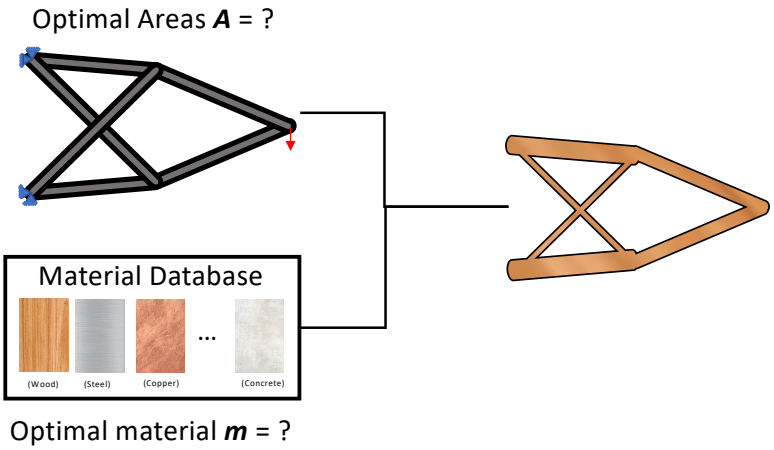

Figure 1: A truss design problem involving optimizing the cross-sectional areas of truss members, and selecting an optimal material.

1.1 Literature Review

Observe that the two sets of design variables are tightly coupled. In other words, one cannot, for example, pick a material, and then optimize the cross-sectional areas (or, vice-versa). To quote [4] *"... either approach does not guarantee the optimal combination of geometry and material"*. Furthermore, while the cross-sectional areas are continuously varying, the material choice is discrete, making the problem difficult to solve using classic gradient-based optimization. Note that treating material properties as continuous, independent design variables is not a viable option since the variables would simply converge to their optimal values. For example, Young's modulus, tensile strength can be expected to converge to the maximum (upper limit) while the variables of cost and density would converge to the lower bound. However, such a material would never exist.

If the geometry is fixed, Ashby's method [5] that relies on a *performance metric*, is the most popular strategy for material selection. Furthermore, for simple design problems where the loads, geometry and material functions are separable [6], a *material index* can be used to select the best material. This is simple, efficient and reliable. However, when there are multiple objectives or constraints, a weighted approach is recommended [7]. In practice, computing these weights is not trivial, and the material choice may be far from optimal. Several non-gradient methods have been proposed for material selection [8, 9, 10], but these cannot be integrated with gradient-based optimization.

An alternate concept of a *design index* was proposed in Ananthasuresh and Ashby [11]. It was shown that the design index can be used to construct a smooth material function, enabling simultaneous optimization of the geometry and selection of the material via gradient based optimization [4]. However, design indices were limited to simple determinate trusses. In [12], for the special case of a single stress constraint and a single load, the authors concluded that utmost two materials are sufficient and that the truss design problem can be cast as a linear programming problem. Unfortunately, this method does not apply when there are multiple constraints. Recently, the authors of [13] considered two materials (glue-laminated timber and steel) to design truss structures incorporating stress constraints as a mixed-integer quadratic problem (MIQPs). However the authors note that *"[MIQPs]... require algorithmic tuning and considerable computational power"*. Another hybrid approach was proposed in [14] where a combination of the gradient-based methods such as sequential quadratic programming (SQP) and an evolution method such as genetic algorithm (GA) are used to solve MDNLPs. But these methods can be prohibitively expensive.

Mathematically, the problem of (discrete) material selection and (continuous) area optimization can be cast as mixed-discrete nonlinear programming problems (MDNLPs). Such problems are fairly common in engineering, and several solution strategies methods have been proposed; see [15, 16, 17]. However, regardless of the strategy, these methods entail repeated solution of a sequence of nonlinear programming problems with careful relaxations and approximations,

making them sensitive to assumptions and underlying models [18]. Furthermore, the popular *branch and bound* algorithm used in solving MDNLPs [16] does not apply here since the optimization problem depends indirectly on the material index through the database.

1.2 Paper Overview

The primary contribution of this work is the use of variational autoencoders (VAEs) to solve such problems. VAEs are a special form of neural networks that can convert discrete data, such as a material database, into a continuous and differentiable representation, making the design problem amenable to gradient-based optimization. In the proposed method (see fig. 2), a data-driven VAE is trained using a material database, and the trained decoder is then integrated with neural networks to simultaneously optimize the geometry and material.

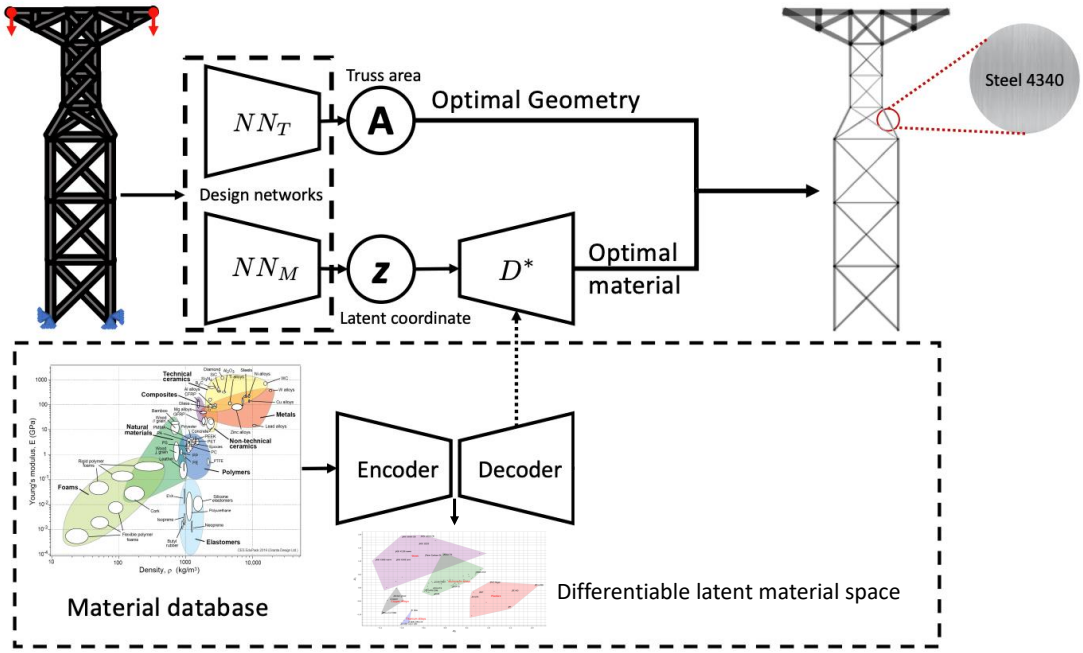

Figure 2: An overview of the proposed method.

The proposed VAE framework is discussed in section 2. In section 3, the VAE is integrated with two additional and simple neural networks to solve the truss design problem. Specifically, we leverage the differentiable material representation to simultaneously optimize the geometry and material. Section 4 demonstrates the proposed framework through several numerical examples. In section 5, limitations and future work are discussed.

2 A Differentiable Material Representation

In this section, we discuss how variational autoencoders (VAEs) can be used to obtain a continuous and differentiable representation of a discrete material database. This will serve as a foundation for the next section on design optimization.

2.1 VAE Architecture and Training

VAEs are popular generative models that have wide applicability in data compression, semi-supervised learning and data interpolation; please see [19]. In essence, a VAE captures the data in a form that can be used to synthesize new samples similar to the input data. For example, VAEs have been used to generate new microstructures from image databases [20], in designing photonic crystals [21], and heat-conduction materials [22].

However, in this paper, we do not use VAEs to synthesize new data, instead *we simply use the VAE's ability to map uncorrelated data onto an abstract latent space*. In that sense, VAEs are similar to principal component analysis (PCA) in its ability to extract useful information from the data. However, the nonlinear nature of VAEs allows for far greater generalization than PCA [23]. In particular, the proposed VAE architecture for capturing material properties is illustrated in fig. 3, and consists of the following components:

1. A four-dimensional *input* module corresponding to the four properties in table 1, namely the Young's modulus (E), cost (C), mass density (ρ) and yield strength (Y). The input set is denoted by $\boldsymbol{\zeta}$.
2. An *encoder* F consisting of a fully-connected network of 250 neurons, where each neuron is associated with an ReLU activation function and weights [24].
3. A two-dimensional *latent space*, denoted by z_0, z_1 that lies at the heart of the VAE.
4. A *decoder* D, which is similar to the encoder, consists of a fully-connected network of 250 neurons.
5. A four-dimensional *output* corresponding to the same four properties; the output set is denoted by $\hat{\boldsymbol{\zeta}}$.

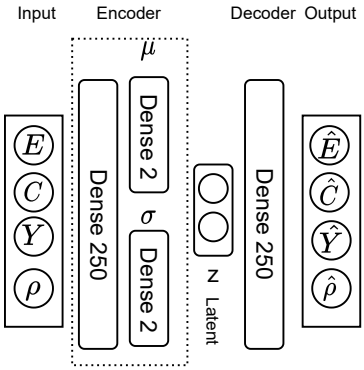

Figure 3: Architecture of the variational autoencoder.

In this work, the VAE was trained on a material database consisting of 92 materials [25], where table 1 represents a small sample.

Material	Class	E $[Pa]$	cost $C[\$/kg]$	$\rho[kg/m^3]$	$Y[Pa]$
A286 Iron	Steel	2.01E+11	5.18E+00	7.92E+03	6.20E+08
AISI 304	Steel	1.90E+11	2.40E+00	8.00E+03	5.17E+08
Gray Cast Iron	Steel	6.62E+10	6.48E-01	7.20E+03	1.52E+08
3003-H16	Al Alloy	6.90E+10	2.18E+00	2.73E+03	1.80E+08
5052-O	Al Alloy	7.00E+10	2.23E+00	2.68E+03	1.95E+08
7050-T7651	Al Alloy	7.20E+10	2.33E+00	2.83E+03	5.50E+08
Acrylic	Plastic	3.00E+09	2.80E+00	1.20E+03	7.30E+07
ABS	Plastic	2.00E+09	2.91E+00	1.02E+03	3.00E+07
PE HD	Plastic	1.07E+09	2.21E+00	9.52E+02	2.21E+07

Table 1: A curated subset of materials and their properties used in the training.

As mentioned earlier, the VAE's primary task is to match the output to the input as closely as possible. This is done through an optimization process (also referred to a training), using the weights associated with the encoder and decoder as optimization parameters. In other words, we minimize $||\boldsymbol{\zeta} - \hat{\boldsymbol{\zeta}}||$. Additionally, a KL divergence loss is imposed to ensure that latent space resembles a standard Gaussian distribution $z \sim \mathcal{N}(\mu = 0, \sigma = 1)$ [19]. Thus, the net loss can then expressed as:

$$L = ||\boldsymbol{\zeta} - \hat{\boldsymbol{\zeta}}|| + \beta \, KL(z||\mathcal{N}) \tag{2}$$

where β is set to a recommended value of $5 * 10^{-5}$. Further the input is scaled between $(0, 1)$ for all attributes, and the output is re-scaled back after training. This ensures that all material properties are weighted equally. In this work, we used PyTorch [26] to model the VAE, and the gradient-based Adam optimizer [27] to minimize eq. (2) with a learning

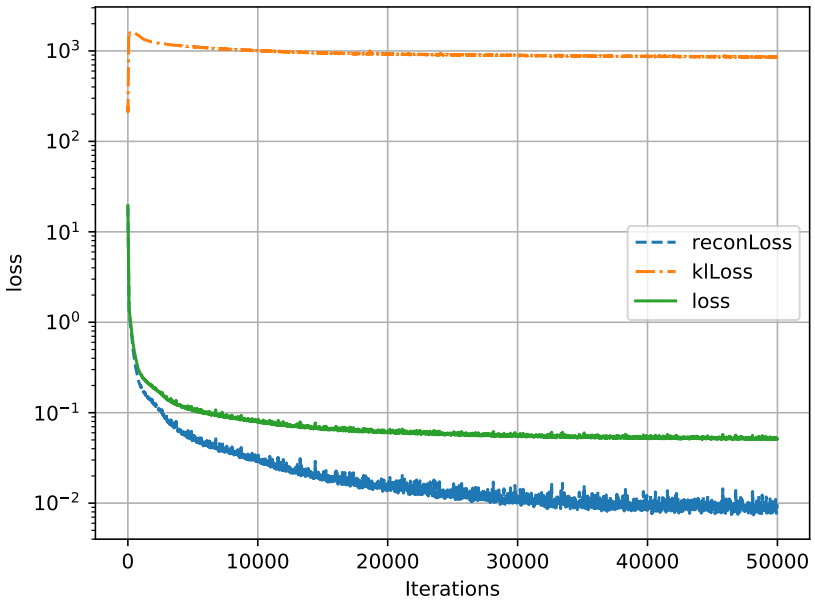

Figure 4: Convergence plot during training of the VAE.

rate of 0.002 and $50,000$ epochs. The convergence is illustrated in fig. 4; the training took approximately 51 seconds on a Macbook M1 pro.

Thus, the VAE captures each material uniquely and unambiguously using a non-dimensional latent space, i.e., z_0, z_1 in our case; this is visualized in fig. 5. For example, annealed AISI 4340 is represented by the pair $(-1.2, 1.0)$ while Acrylic is represented by $(0.6, -0.4)$. The VAE also clusters similar materials together in the latent space, as can be observed in fig. 5. In an abstract sense, the latent space in fig. 5 is similar to the popular Ashby charts [5].

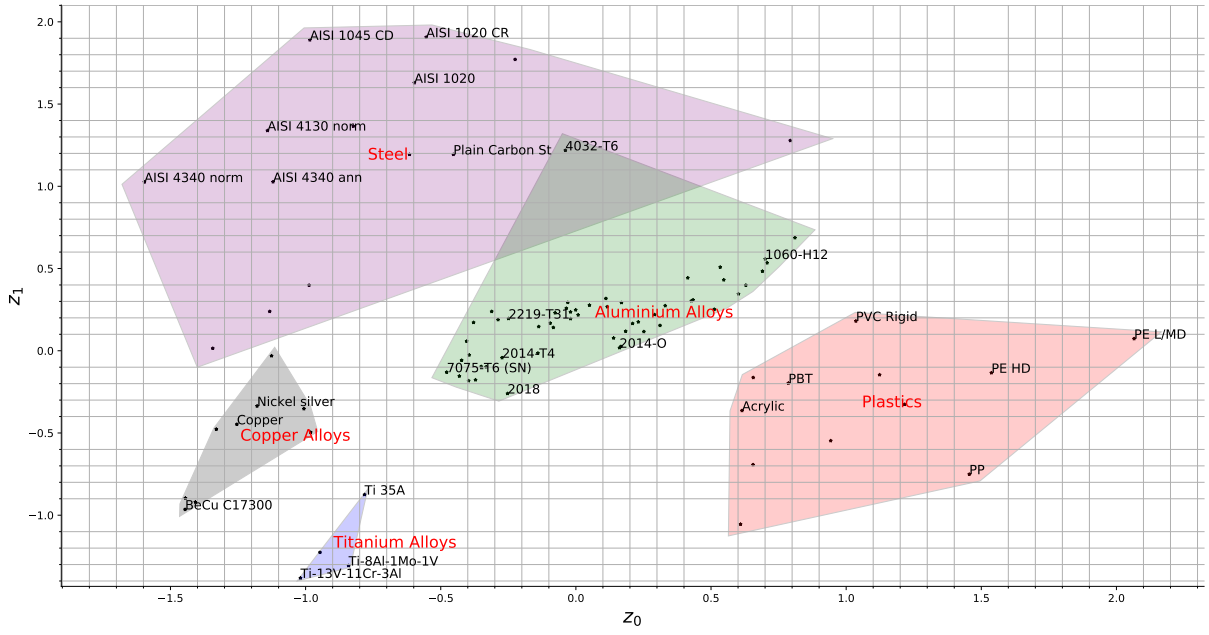

Figure 5: Encoding of the materials in the latent space; only a few materials are annotated for clarity.

2.2 Representational Accuracy

One can expect the values reconstructed using the decoder to deviate from the true material data; table 2 summarizes the errors. The following observations are worth noting:

1. Despite the lack of correlation between material properties, and two orders of magnitude difference in values, the VAE captures the entire database of 92 materials reasonably well, using a simple two-dimensional latent space.
2. The error can be further reduced by either increasing the dimension of the latent space, or limiting the type of materials considered (see section 4).
3. Finally, as will be discussed in the next section, the original material data will be used, as part of a post-processing step, at the end of optimization.

Material	$\Delta E\%$	$\Delta C\%$	$\Delta \rho\%$	$\Delta Y\%$
A286 Iron	0.8	0.7	0.8	0.0
ABS	1.4	0.0	0.7	1.7
AISI 304	4.3	3.1	0.4	1.0
Gray Cast Fe	1.3	1.8	0.2	0.2
3003-H16	3.2	6.8	1.3	1.1
5052-O	1.4	2.4	0.6	1.2
7050-T7651	0.9	1.9	1.0	1.9
Acrylic	2.3	0.5	0.2	0.5
PE HD	0.7	0.9	1.1	1.8
Max error	**5.0**	**6.8**	**3.3**	**7.1**

Table 2: Percentage error between actual and decoded data.

2.3 Differentiability of Latent Space

A crucial aspect of the latent space is its differentiability. In particular, we note that using the decoder, each of the output material properties is represented via analytic activation functions; for example, $\hat{E} = D^*_E(z_0, z_1)$ where $D^*_E(\cdot)$ is a combination of activation functions and weights. It means that one can back propagate through the decoder to compute analytical sensitivities; for example, the sensitivity ($\frac{\partial \hat{E}}{\partial z_0}$). This plays an important role in gradient-driven design optimization. The proposed method thus falls under the category of *differentiable programming*, which has recently gained popularity motivated by deep learning methods in other fields, such as computer graphics [28] and physical simulation [29].

2.4 Material Similarity

While the accuracy and differentiability of the VAE lends itself to design optimization, the latent space also provides key insights into material characteristics. For instance, one can compute the Euclidean distance between materials in the latent space. These distances are reported in fig. 6. As one can expect, steels are closer to aluminum alloys, than they are to plastics.

Furthermore, one can overlay the latent space in fig. 5 with specific material properties to gain further insight. This is illustrated in fig. 7 where the contour plots of the Young's moduli are illustrated in the latent space. This allows designers to visualize similarities between materials, specific to a material property.

3 Design Optimization

Having constructed a differentiable representation of material properties, one can now pose the optimization problem discussed in section 1, now using $\{\boldsymbol{A}, z_0, z_1\}$ as continuous design variables eq. (3). Without a loss in generalization, we consider a specific instance of the truss optimization problem, where the objective is the compliance J eq. (3a) with three sets of constraints: cost constraint g_c, buckling constraint g_b, and tensile yield constraint g_y as referred in eq. (3c)-eq. (3e). We assume here that members will fail due to buckling, before failing due to compressive yield [4].

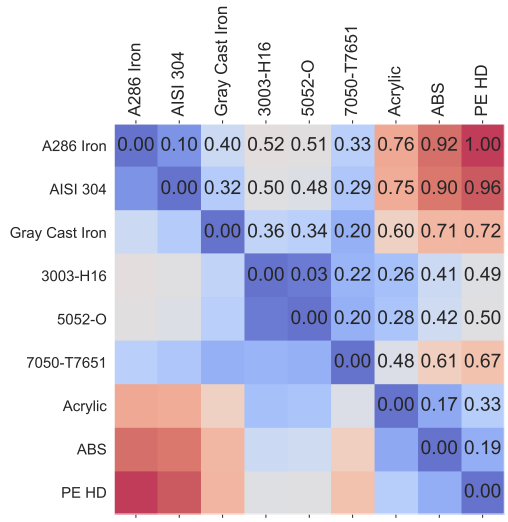

Figure 6: Symmetric distance in the latent space between materials

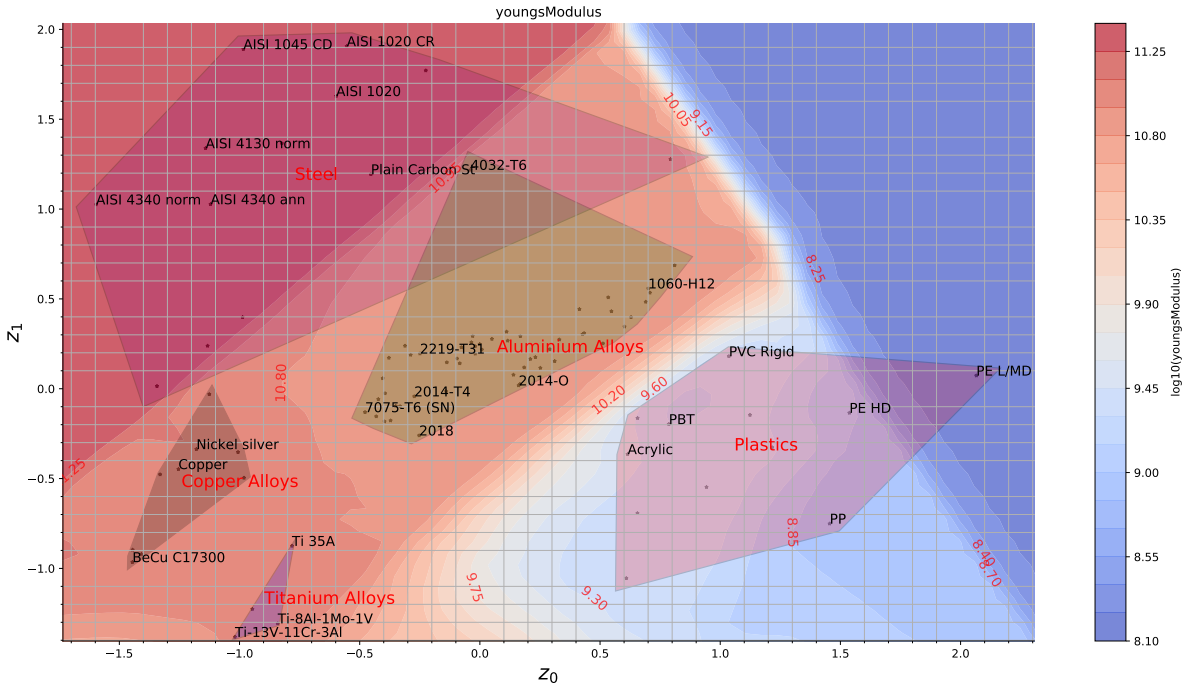

Figure 7: Contours of the Young's moduli in the latent space.

(a) Topology network NN_T

(b) Material network NN_M with decoder.

Figure 8: The truss and material neural networks.

The resulting optimization problem for circular truss members can be posed as [30]:

$$\underset{\boldsymbol{A}=\{A_1,A_2,\ldots A_N\},z_0,z_1}{\text{minimize}} \quad J = \boldsymbol{f}^T\boldsymbol{u}(\boldsymbol{A},\hat{E}) \tag{3a}$$

$$\text{subject to} \quad [K(\boldsymbol{A},\hat{E})]\boldsymbol{u} = \boldsymbol{f} \tag{3b}$$

$$g_c \coloneqq \left(\frac{\hat{\rho}\hat{C}}{C^*}\sum_{k=1}^{N} A_k L_k\right) - 1 \leq 0 \tag{3c}$$

$$g_b \coloneqq \max_k\left(\frac{-4P_k L_k^2}{\pi^2 \hat{E} A_k^2}\right) - \frac{1}{F_s} \leq 0 \tag{3d}$$

$$g_y \coloneqq \max_k\left(\frac{P_k}{\hat{Y} A_k}\right) - \frac{1}{F_s} \leq 0 \tag{3e}$$

$$\boldsymbol{A}_{min} \leq \boldsymbol{A} \leq \boldsymbol{A}_{max} \tag{3f}$$

where P_k is the internal force in member k, L_k is its length, C^* is the allowable cost, F_s is the safety factor, and the material properties $(\hat{E},\hat{\rho},\hat{Y},\hat{C})$ are decoded from the latent space coordinates (z_0,z_1). Further, to facilitate gradient driven optimization, the max operator is relaxed here using a p-norm, i.e., $\max_i(x_i) \approx ||\boldsymbol{x}||_p$, with $p = 6$.

To solve the above optimization we use two additional neural networks (NNs) (see fig. 8a): a truss network NN_T, parameterized by weights $\boldsymbol{w}_T$, and a material network NN_M, parameterized by weights $\boldsymbol{w}_M$. The truss network NN_T is a simple feed-forward NN with two hidden layers with a width of 20 neurons, each containing an ReLU activation function. The input to the NN_T is a unique identifier for each truss member, in this case we use the vector of coordinates of the truss member centers. The output layer of NN_T consists of N neurons where N is the number of truss members, activated by a Sigmoid function, generates a vector $\boldsymbol{O_T}$ of size N whose values are in $[0,1]$. The output is then scaled as $\boldsymbol{A} \leftarrow \boldsymbol{A}_{min} + \boldsymbol{O_T}(\boldsymbol{A}_{max} - \boldsymbol{A}_{min})$ to satisfy the area bounds eq. (3f). The material network NN_M (fig. 8b) is similar to NN_T in construction, for simplicity. Since we are selecting a single material for all the truss members, we do not require a vector of inputs corresponding to each member to the NN_M and instead pass a scalar identifier 1 as its input.. The output layer consists of two output neurons activated by Sigmoid functions. The outputs $\boldsymbol{O_M}$ are scaled as $\boldsymbol{z} \leftarrow -3 + 6\boldsymbol{O_M}$, resulting in $z_i \in [-3,3]$ corresponding to six Gaussian deviations. These outputs now interface with the trained decoder D^* from fig. 3 . Thus, by varying the weights $\boldsymbol{w}_M$ of the material network one can create points in the latent space, that then feeds to the trained decoder resulting in values of material constants.

3.1 Loss Function

With the introduction of the two NNs, the weights $\boldsymbol{w}_T$ now control the areas $\boldsymbol{A}$, while the weights $\boldsymbol{w}_M$ control the material constants $\hat{\boldsymbol{\zeta}}$. In other words, the weights $\boldsymbol{w}_T$ and $\boldsymbol{w}_M$ now form the design variables. Further, since NNs are designed to minimize an unconstrained loss function, we convert the constrained minimization problem in eq. (3) into an unconstrained minimization by employing a log-barrier scheme as proposed in [31]:

$$L(\boldsymbol{w}_T,\boldsymbol{w}_M) = J + \psi(g_c) + \psi(g_b) + \psi(g_y) \tag{4}$$

where,

$$\psi_t(g) = \begin{cases} -\frac{1}{t}\log(-g)\ , & g \leq \frac{-1}{t^2} \\ tg - \frac{1}{t}\log(\frac{1}{t^2}) + \frac{1}{t}\ , & \text{otherwise} \end{cases} \tag{5}$$

with the parameter t updated during each iteration (as described in the next section). Thus the optimization problem reduces to a simple form:

$$\underset{\boldsymbol{w}_T, \boldsymbol{w}_M}{\text{minimize}} \quad L(\boldsymbol{w}_T, \boldsymbol{w}_M) \tag{6a}$$

$$\text{subject to} \quad [K(\boldsymbol{A}(\boldsymbol{w}_T), \hat{E}(\boldsymbol{w}_M))]\boldsymbol{u} = \boldsymbol{f} \tag{6b}$$

A schematic of the proposed framework is presented in fig. 9.

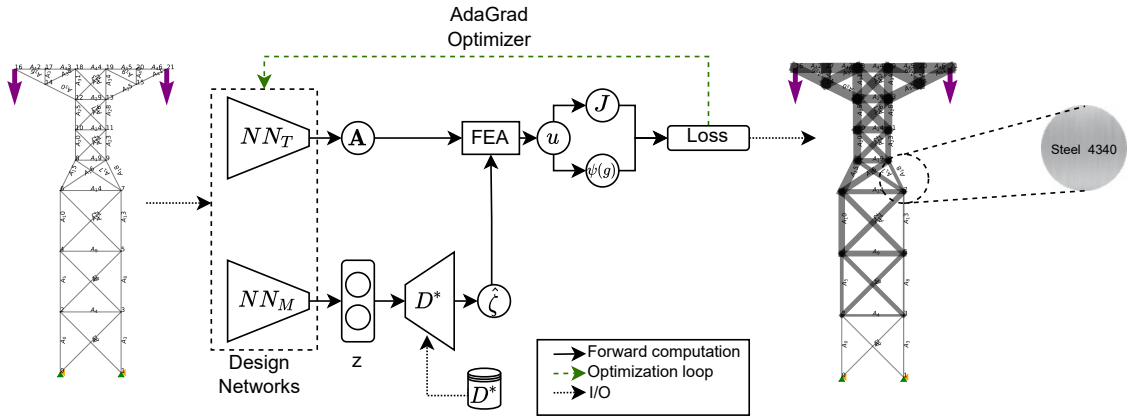

Figure 9: Forward computation

3.2 Structural Analysis

We rely on classical structural analysis to solve the state equation eq. (6b) [32] and evaluate the performance of the truss structure during each iteration. The solver computes the stiffness matrix for each member based on the corresponding area, length and material. Upon assembling the global stiffness matrix, the nodal displacement vector u is computed using the standard linear solver `torch.linalg.solve` in PyTorch [26]. Since this is part of the PyTorch library, this allows us to exploit backward propagation for automatic differentiation [33], resulting in an end-to-end differentiable solver with automated sensitivity analysis as described next.

3.3 Sensitivity Analysis

In the proposed framework, the loss function in eq. (4) is minimized using gradient-based Adagrad optimizer [34]. Further, the sensitivities are computed automatically using back propagation. A schematic representation of the backward computation graph is shown in fig. 10. For instance the sensitivity of the loss function with respect to the weights $\boldsymbol{w}_T$ can be expressed as:

$$\frac{\partial L}{\partial \boldsymbol{w}_T} = \Bigg[\underbrace{\frac{\partial L}{\partial J}}_{I} \underbrace{\frac{\partial J}{\partial \boldsymbol{u}}}_{II} \underbrace{\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{A}}}_{III} + \dots \Bigg] \underbrace{\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{w}_T}}_{IV} \tag{7}$$

This is illustrated in fig. 10 where each term corresponds to an edge in the graph. Similarly, the sensitivity with respect to $\boldsymbol{w}_M$ is given by:

$$\frac{\partial L}{\partial \boldsymbol{w}_M} = \Bigg[\underbrace{\frac{\partial L}{\partial J}}_{I} \underbrace{\frac{\partial J}{\partial \boldsymbol{u}}}_{II} \underbrace{\frac{\partial \boldsymbol{u}}{\partial \hat{\zeta}}}_{V} + \dots \Bigg] \underbrace{\frac{\partial \hat{\zeta}}{\partial \boldsymbol{z}}}_{VI} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{w}_M}}_{VII} \tag{8}$$

These are once again illustrated in fig. 10. Note that the backward computation is carried out automatically in the proposed framework.

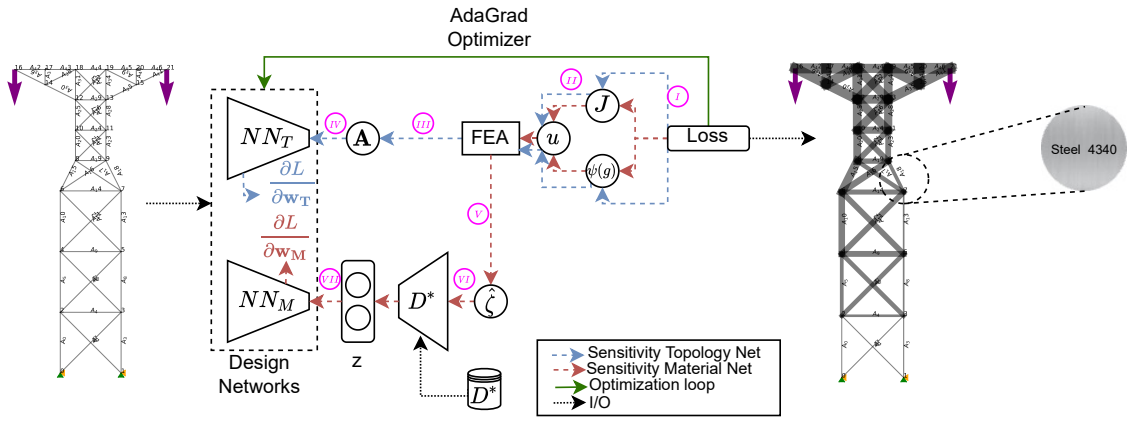

Figure 10: Backward computation

3.4 Post Processing

Once the optimization process is complete, we obtain an optimal set of latent coordinates $\boldsymbol{z}^*$. However, there might not exist a material in the database corresponding precisely to $\boldsymbol{z}^*$. We therefore define a *confidence metric* γ_m for each material m by using the distance from $\boldsymbol{z}^*$ to the material m in the latent space:

$$\gamma(\boldsymbol{z}_m, \boldsymbol{z}^*) = 1 - \frac{||\boldsymbol{z}^* - \boldsymbol{z}_m||}{\max\limits_{\forall k \in M}(||\boldsymbol{z}^* - \boldsymbol{z}_k||)} \tag{9}$$

The metric serves to rank the materials based on their distance from $\boldsymbol{z}^*$. After finding the closest material, we repeat the geometry optimization to compute the optimal areas $\boldsymbol{A}^*$. This is analogous to the *rounding* method in discrete optimization [16]: "*After rounding, it is usually best to repeat the optimization once more, allowing only the continuous design variables to vary. This process may not lead to the exact optimum, and sometimes may not even lead to a feasible solution, but for many problems this is an effective approach.*" This is explored further through numerical experiments in section 4.

3.5 Algorithm

The proposed algorithms are summarized in this section.

Algorithm 1 As mentioned earlier, we assume that a material database has been provided. The procedure begins with training the VAE as described in algorithm 1, with the input training data being the material database. Encoder F is a neural network that takes in the set of material data ζ and encodes the four-dimensional data to the two-dimensional latent space denoted by z_0, z_1 through a probabilistic latent distribution governed by μ, σ. The VAE Loss is then used to drive the training per eq. (2) until a sufficiently high representational accuracy is attained. The encoder is then discarded and the decoder D is retained for material selection. The decoder takes the latent space coordinates as input and returns the expected material properties.

Algorithm 2 In algorithm 2, starting with the given truss, specified loads, restraints, area bounds and other constraints, the areas are optimized and simultaneously the best material is selected as follows. The truss neural network and material neural network are constructed as described in section 3; these output the cross sectional area and the latent space coordinates respectively. The decoder D (from above) then returns the material properties, and now with the area and material information, the state equation can be solved and the objective J can be evaluated per section 3.2. The constraints can also be computed post finite element analysis and the loss function eq. (4) is next evaluated. Sensitivity of the objective and constraints is available through automatic differentiation as described in section 3.3. In our experiments $t_0 = 3$ and $\mu = 1.01$ for the log-barrier implementation eqs. (4) and (5), and the learning rate for the Adagrad optimizer was set to the value of $2 * 10^{-3}$ to solve the eq. (6). Once convergence is reached, we re-optimize for the optimal areas using the nearest material in the latent space as described in section 3.4.

Algorithm 1 Encode Materials

procedure MATENCODE(ζ , F, D) ▷ Input: Training data, encoder and decoder
  epoch = 0 ▷ iteration counter
  repeat ▷ VAE training
    $F(\zeta) \rightarrow \{\mu, \sigma\}$ ▷ Forward prop. encoder
    $\{\mu, \sigma\} \rightarrow z$ ▷ Reparameterization [19]
    $\{\mu, \sigma\} \rightarrow KL(z||\mathcal{N})$ ▷ KL loss
    $D(z) \rightarrow \hat{\zeta}$ ▷ Forward prop. decoder
    $\{\zeta, \hat{\zeta}, KL\} \rightarrow L$ ▷ VAE Loss
    $\boldsymbol{w} + \Delta\boldsymbol{w}(\nabla L) \rightarrow \boldsymbol{w}$ ▷ Update VAE weights, Adam Optimizer
    epoch $++$
  until error is acceptable ▷ Iterate
  return D ▷ Trained decoder

Algorithm 2 Truss Optimization

procedure TRUSSOPT(truss, loads, restraints, D^*, ...) ▷ Input: Initial truss, trained decoder, max cost
  k = 0 ▷ iteration counter
  repeat ▷ Optimization (Training)
    $NN_T(\boldsymbol{w}_T) \rightarrow \boldsymbol{A}$ ▷ Fwd prop NN_T; compute truss areas
    $NN_M(\boldsymbol{w}_M) \rightarrow z$ ▷ Fwd prop NN_M; compute latent coordinate
    $D^*(z) \rightarrow \hat{\boldsymbol{\zeta}}$ ▷ Fwd prop D^*; compute material properties
    $\{\boldsymbol{A}, \hat{E}\} \rightarrow [K]$ ▷ compute stiffness matrix
    $\{[K], \boldsymbol{f}\} \rightarrow \boldsymbol{u}$ ▷ State Eq., FEA eq. (3b)
    $\{\boldsymbol{u}, \boldsymbol{f}\} \rightarrow J$ ▷ Compliance, design objective eq. (3a)
    $\{\boldsymbol{A}, \hat{\rho}, \hat{C}\} \rightarrow g_c$ ▷ cost constraint eq. (3c)
    $\{\boldsymbol{A}, \hat{E}\} \rightarrow g_b$ ▷ Buckling constraint eq. (3d)
    $\{\boldsymbol{A}, \hat{Y}\} \rightarrow g_y$ ▷ Yielding constraint eq. (3e)
    $\{J, g_c, g_b, g_y\} \rightarrow L$ ▷ Loss eq. (4)
    $AD(L \leftarrow \boldsymbol{w}_T\ ,\ g_c \leftarrow \boldsymbol{w}_T\ ,\ g_b \leftarrow \boldsymbol{w}_T,\ g_y \leftarrow \boldsymbol{w}_T) \rightarrow \nabla_{\boldsymbol{w}_T} L$ ▷ auto diff. for sens. w.r.t. NN_T eq. (7)
    $AD(L \leftarrow \boldsymbol{w}_M\ ,\ g_c \leftarrow \boldsymbol{w}_M,\ g_b \leftarrow \boldsymbol{w}_M,\ g_y \leftarrow \boldsymbol{w}_M) \rightarrow \nabla_{\boldsymbol{w}_M} L$ ▷ auto diff. for sens. w.r.t. NN_M eq. (8)
    $\boldsymbol{w}_j - \Delta\boldsymbol{w}_j(\nabla_{\boldsymbol{w}_j} L) \rightarrow \boldsymbol{w}_j\ ;\quad j = \{T, M\}$ ▷ AdaGrad optimizer step
    $t \leftarrow t_0\mu^k$ ▷ Increment t; log-barrier term
    k $++$
  until $||\Delta w|| < \epsilon^*$ ▷ Check for convergence
  return $\{z, \boldsymbol{A}\}$
  Find nearest material and optimize for area ▷ section 3.4

4 Numerical Experiments

In this section, we present results from several experiments that illustrate the method and algorithm.

4.1 Simultaneous Optimization of Area and Material

The central hypothesis of this paper is that simultaneous optimization of material and geometry leads to superior performance compared to sequential optimization. To validate this hypothesis, we consider the 6-bar truss system illustrated in fig. 11, and perform the following three experiments:

1. **Material optimization**: Here the areas are fixed at the initial value of $A_k = 2 * 10^{-3} m^2, \forall k$, and only the material is optimized using the decoder.
2. **Area optimization**: Using the optimal material from the first experiment, the areas are optimized by disregarding the decoder.
3. **Simultaneous material and area optimization**: Here the material and areas are optimized simultaneously.

The results are reported in table 3 for a cost constraint of $C^* = \$60$ and a safety factor $F_S = 4$. Observe that scenario-1 (material optimization) leads to the lowest performance since the areas are not optimized. For this scenario, we also

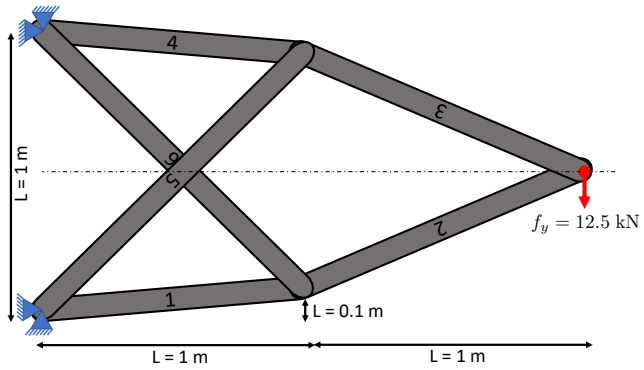

Figure 11: Loading of 6-bar mid-cantilever truss.

performed a brute-force search wherein the compliance was evaluated for all the materials; the brute-force search also resulted in 'AISI 4130 norm' as the best choice. In scenario-2, when the areas are optimized using 'AISI 4130 norm' as the material, the performance improved significantly. Finally, the best performance is achieved in scenario-3 when both material and areas are optimized simultaneously. Observe that the optimal material in scenario-3 differs from the one found in scenario-1.

Scenario	J^*	Closest Material (confidence)	Area ($10^{-3}m^2$)
1	3.79	AISI 4130 norm (89.76%)	[2,2,2,2,2,2]
2	2.95	AISI 4130 norm (89.76%)	[3.2,2.6,2.6,3.2,1,1]
3	2.58	AISI 1010 (90.69%)	[3.6,3.1,3.1,3.5,1.3,1.2]

Table 3: Cost constrained optimization of the truss in fig. 11

For scenario-3, table 4 lists the three closest materials and their confidence values.

Closest Material	Confidence (γ)
1. AISI 1010	90.69%
2. 4032-T6	86.64%
3. AISI 1020	80.01%

Table 4: Three closest materials under cost constraint.

As an additional experiment, we replaced the cost constraint (eq. (3c)) in the above problem with a mass constraint:

$$g_m \coloneqq \left(\frac{\rho}{M^*}\sum_{k=1}^{N} A_k L_k\right) - 1 \leq 0 \tag{10}$$

with $M^* = 40$kg; all other parameters being the same. The results are presented in table 5. Once again we observe that simultaneous optimization results in the best performance. Furthermore, when cost constrained was imposed, a steel alloy was chosen as the best material, whereas when a mass constrained is imposed, an aluminium alloy was chosen.

Scenario	J^*	Closest Material (confidence)	Area ($10^{-3}m^2$)
1	10.5	Al 380F die (91.63%)	[2,2,2,2,2,2]
2	8.75	Al 380F die (91.63%)	[3,2.5,2.4,3.3,1,1]
3	8.56	Al 2018 (94.16%)	[3.1,2.4,2.5,3.1,1,1]

Table 5: Mass constrained optimization of the truss in fig. 11

For scenario-3 under mass constraint, table 6 lists the three closest materials and the confidence values.

The location of the optimal material in the latent space for this scenario is illustrated in figs. 12 and 13. We observe that the closest catalog material for scenario-3 is Al 2018. The latter is chosen, and the areas are subsequently re-optimized.

Closest Material	Confidence (γ)
1. Al 2018	94.16%
2. Al 380F die	90.64%
3. Al 2014-T4	90.13%

Table 6: Three closest materials under mass constraint.

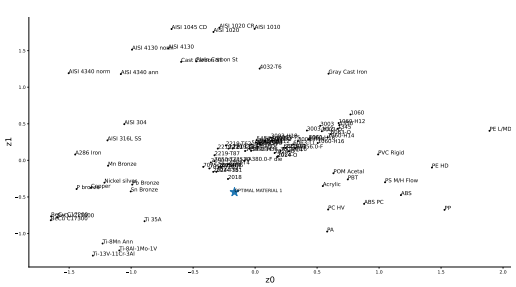

Figure 12: Optimal material in latent space.

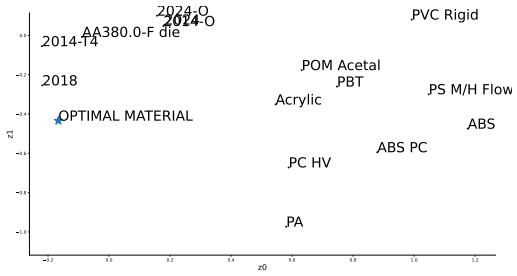

Figure 13: A close-up of fig. 12; the closest material to the optimal one is Al 2018.

4.2 Material Database Subsets

While we have used the entire material database in the previous experiment, we now consider subsets of the database. Subsets of materials can often be justified based on experience. The VAE reconstruction error for the full database and three sample subsets is reported in table 7. As one expect, the VAE error reduces for the subsets.

Material class	No. materials	$\Delta E\%$	$\Delta C\%$	$\Delta\rho\%$	$\Delta Y\%$
All Matls.	92	5.0	6.8	3.3	7.1
Steel	14	0.58	2.6	0.07	1.4
Aluminum	52	0.09	0.22	0.04	2.9
Plastic	12	2.3	0.36	0.5	0.7

Table 7: VAE reconstruction error trained with the full and various material subsets.

We now carry out a few experiments using these subsets, and report the results. First, for the truss problem in fig. 11, we carry out a cost-constrained simultaneous optimization for the full database, as in the previous section. However, we now report additional results in table 8. When all materials are considered (row-2), the compliance prior to snapping to the closest material is reported as $J = 2.25$. The closest material is AISI 1010, resulting in a final compliance of $J^* = 2.58$. As one can expect, $J^* > J$.

Next, since AISI 1010 is a steel alloy, we now consider only the Steel subset, and carry out simultaneous optimization. The results are reported in row-3. Observe that the raw compliance $J = 2.6$ is surprisingly larger than $J = 2.25$ in the previous row. This can be attributed to the larger VAE reconstruction error when all materials are considered, i.e., $J = 2.25$ is not as reliable as the estimate $J = 2.6$. Further, after the closest material is selected (AISI 1020), the final compliance is $J^* = 2.6$. This is slightly larger than the value in the previous row. This discrepancy is analogous to rounding errors in integer optimization [16].

Material class	J	Optimal material	E	ρ	Y	J^*	Area ($10^{-3}m^2$)
All Matls.	2.25	AISI 1010	2E11	7.9E3	3.3E8	2.58	[3.6,3.1,3.1,3.5,1.3,1.2]
Steel	2.6	AISI 1020	2E11	7.9E3	4.2E8	2.6	[3.7,2.9,3.1,3.8,1.1,1.2]

Table 8: Material refinement with cost constraint.

Next, we repeated the above experiment for a mass constraint (instead of a cost constraint) and report the results in table 9. When all materials are considered (row-2), the algorithm converges to the aluminum class, with an optimal material Al 2018, as before. Further, $J^* > J$ as expected. The results using the aluminum subset is reported in row-3. Now, the optimal material is AA356.0-F, and it differs from Al 2018. Now the performance $J^* = 8.38$ has significantly improved compared to $J^* = 8.53$.

Material class	J	Optimal material	E	ρ	Y	J^*	Area ($10^{-3}m^2$)
All Matls.	8.21	Al 2018	7.40E10	2.80E3	4.21E8	8.53	[3.1,2.4,2.5,3.1,1,1]
Aluminum	8.23	AA356.0-F	7.24E10	2.68E3	1.45E8	8.38	[3.2,2.7,2.6,3.2,0.9,1]

Table 9: Material refinement with mass constraint.

4.3 Convergence

To study the convergence characteristics of simultaneous optimization, we consider the antenna tower illustrated in fig. 14. We impose a cost constraint of \$50, factor of safety of $F_S = 4$, and area bounds $[10^{-9}, 10^{-2}]m^2$. Then the areas and material are simultaneously optimized, resulting in an optimal material of 'AISI 4130 norm' and $J^* = 32.5$. The convergence in compliance is illustrated in fig. 15. We observe that the material choice converges early while the areas continue to be optimized; this was typical. The optimization took 1.89 seconds, with forward propagation and FEA, accounting for 16% each, while back-propagation consumed 54%.

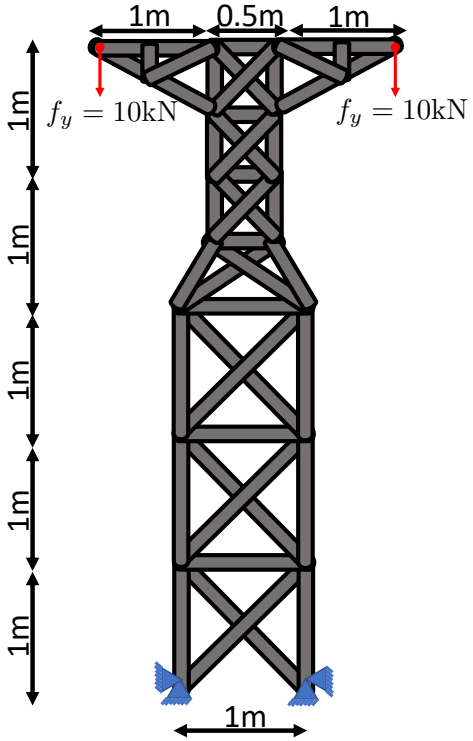

Figure 14: Loading of 47-bar antenna tower.

4.4 Extension to 3D and Pareto Design

In this section we demonstrate two examples of simultaneous design optimization with material selection in 3D. The first example is that of a cantilever, and the second is a transmission tower, as illustrated in fig. 16. We further demonstrate that the proposed method generates Pareto designs under mass and cost constraints.

First, we consider the mid-loaded cantilever in fig. 16 with a $20kN$ force, and seek to optimize the truss cross sectional areas and select the optimal material from the database. In this problem, along with the safety constraints on buckling and yielding (with a factor of safety of 4), we impose a mass constraint. The final compliance values (J^*) and the selected optimal material, for various mass constraints, are summarized in table 10. The resulting truss designs are illustrated in fig. 17 (cross sectional values not included for brevity). From fig. 17 we observe that at lower target masses, aluminum alloys are preferred, and at higher masses, steels are selected.

Next, we consider the 3D transmission tower illustrated in fig. 16 with a $100N$ force, and seek to optimize the truss cross sectional areas and select the optimal material from the database. In this problem, along with the safety constraints

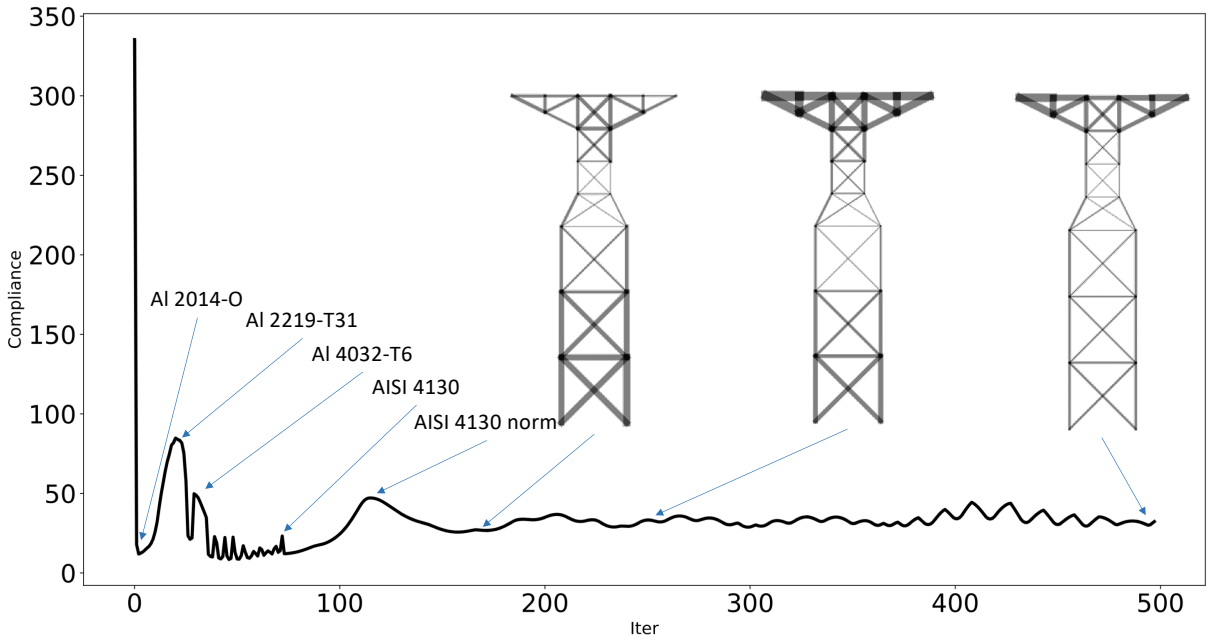

Figure 15: Convergence of compliance for the material and area for loading in fig. 14.

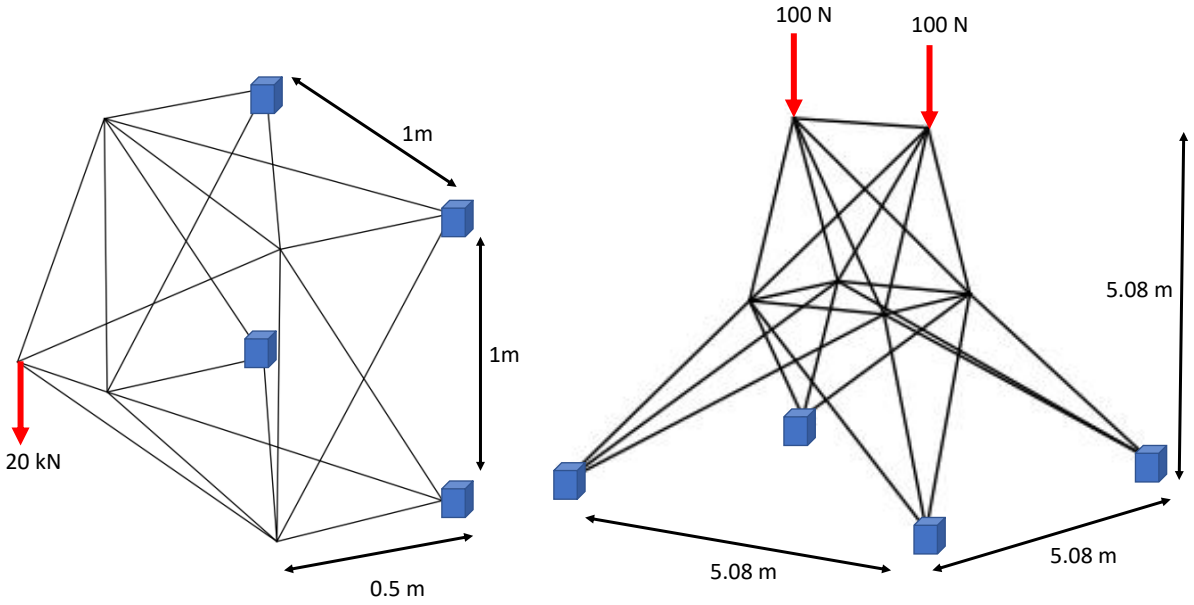

Figure 16: Examples in 3D: a mid noded cantilever and a 25 member transmission tower

of buckling and yielding (with a factor of safety of 4), we impose a cost constraint. The final compliance values (J^*) and the selected optimal material, for various cost constraints, are summarized in table 11. The truss cross-sections are illustrated in fig. 18 (cross sectional values not included for brevity). From fig. 18 we observe that at lower target costs, Steel alloys are preferred, and at higher costs, Aluminum alloys are selected.

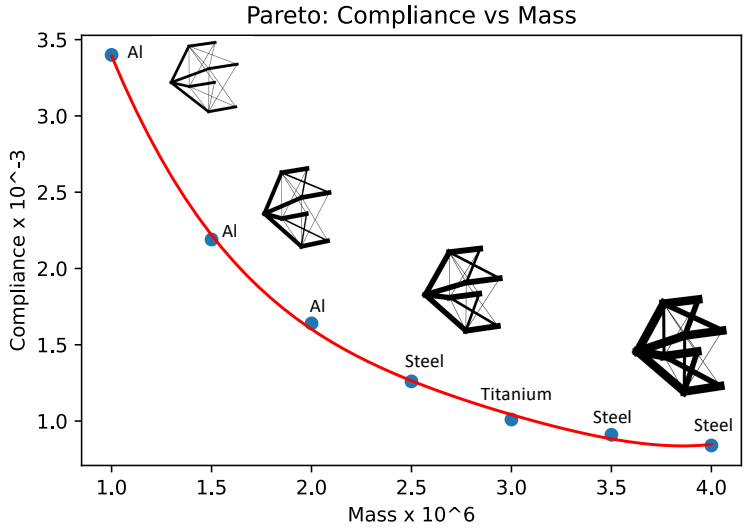

Figure 17: Pareto designs under mass constraint: optimized cross sectional areas and best material choices are depicted.

Target mass	**Final compliance J^***	**Closest material**
1e6	3.4e-3	Al 6063
1.5e6	2.19e-3	Al 6061
2e6	1.64e-3	Al 6061
2.5e6	1.26e-3	Steel 2018
3e6	1.01e-3	Ti-6Al-1Mo-1V
3.5e6	9.10e-4	Steel 2018
4e6	8.36e-4	Steel 2018

Table 10: Final compliance and closest material for the 3D cantilever problem under mass constraint.

Target cost	**Final Compliance J^***	**Closest material**
2e4	8.15e-6	1345 Steel
4e4	6.06e-6	1345 Steel
5e4	4.86e-6	1345 Steel
8e4	3.09e-6	1350 Steel
1e5	2.45e-6	6063 Al
1.2e5	2.03e-6	356 Al
1.5e5	1.72e-6	6063 Al
2e5	1.45e-6	6063 Al

Table 11: Final compliance and closest material for the 3D transmission tower problem under cost constraint

5 Conclusion

Engineers today have access to over $150,000$ materials [35], and this number is growing as new materials are being discovered [36] [37]. This presents a significant challenge to design engineers. In this paper, using variational auto encoders (VAEs), we proposed a generic method to simultaneously select the optimal material, while optimizing the geometry. The proposed method was demonstrated using trusses.

There are several limitations to the proposed method. A heuristic confidence-metric was used, as a final step, to snap from the continuous latent space to the nearest material. While this is a well known method in integer optimization, and was found to be effective in our study, it might lead to discrepancies in performance, as was noted in the numerical experiments. On the other hand, since the proposed method suggests an optimal set of material properties, it could drive material innovation in targeted applications. A very small material database with about 100 materials, with limited number of attributes, was considered here. Larger databases [38] with additional attributes needs to be explored and

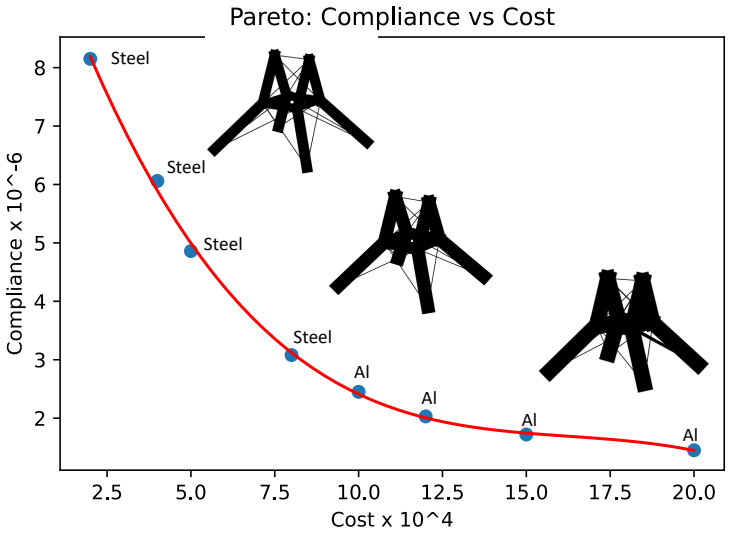

Figure 18: Pareto designs under cost constraint: optimized cross sectional areas and best material choices are depicted.

scalability of method remains to be evaluated. In passing, we note that VAEs have shown promise in other applications, with datasets larger than 50,000 entries [39, 40]. The method could further benefit from better tuning of the networks, optimal choice of the NN architecture [41], and reducing the reconstruction error of VAEs [42]. Another promising direction for discrete selection within continuous frameworks is differentiable programming [29, 43] and learnable lookup tables [44]. Comparison of the proposed method against such methods will be considered in the future.

Despite these limitations, we believe that the method holds promise. For example, it could be extended to topology optimization with material selection, specifically using the NN framework proposed in [45]. Furthermore, inclusion of a diverse set of material attributes, such as thermo-elastic properties [46], auxetic properties [47], coating properties [48], global warming indices [13] can also be considered. Inclusion of uncertainty in material properties is also of significant interest. Finally, the method can be extended to the selection of discrete components in assemblies such as springs, bolts, etc., and to the selection of discrete microstructures [49] in multiscale designs.

Compliance with ethical standards

The authors declare that they have no conflict of interest.

Reproduction of Results

The Python code pertinent to this paper is available at https://github.com/UW-ERSL/MaTruss. The material data was sourced from SolidWorks.

Acknowledgements

The authors would like to thank the support of National Science Foundation through grant CMMI 1561899.

References

[1] Rudy Eggert. *Engineering design*. Pearson/Prentice Hall, 2005.

[2] G. I. N. Rozvany, Bendsoe M.P., and U. Kirsch. Layout optimization of structures. *Applied Mechanics Reviews*, 48(2):41–119, 02 1995.

[3] W Achtziger. Truss topology optimization including bar properties different for tension and compression. *Structural optimization*, 12(1):63–74, 1996.

[4] Sourav Rakshit and GK Ananthasuresh. Simultaneous material selection and geometry design of statically determinate trusses using continuous optimization. *Structural and Multidisciplinary Optimization*, 35(1):55–68, 2008.

[5] Michael F Ashby and David CEBON. Materials selection in mechanical design. *Le Journal de Physique IV*, 3(C7):C7–1, 1993.

[6] Michael F Ashby and Kara Johnson. *Materials and design: the art and science of material selection in product design*. Butterworth-Heinemann, 2013.

[7] MF Ashby. Multi-objective optimization in material design and selection. *Acta materialia*, 48(1):359–369, 2000.

[8] A Jahan, MY Ismail, SM Sapuan, and F Mustapha. Material screening and choosing methods–a review. *Materials & Design*, 31(2):696–705, 2010.

[9] R Venkata Rao. A material selection model using graph theory and matrix approach. *Materials Science and Engineering: A*, 431(1-2):248–255, 2006.

[10] Chang-Chun Zhou, Guo-Fu Yin, and Xiao-Bing Hu. Multi-objective optimization of material selection for sustainable products: artificial neural networks and genetic algorithm approach. *Materials & Design*, 30(4):1209–1215, 2009.

[11] GK Ananthasuresh and MF Ashby. Concurrent design and material selection for trusses. 2003.

[12] Mathias Stolpe and Krister Svanberg. A stress-constrained truss-topology and material-selection problem that can be solved by linear programming. *Structural and Multidisciplinary Optimization*, 27(1):126–129, 2004.

[13] Ernest Ching and Josephine V Carstensen. Truss topology optimization of timber–steel structures for reduced embodied carbon design. *Engineering Structures*, page 113540, 2021.

[14] Satadru Roy, William A Crossley, and Samarth Jain. A hybrid approach for solving constrained multi-objective mixed-discrete nonlinear programming engineering problems. 2021.

[15] JS Arora, MW Huang, and CC Hsieh. Methods for optimization of nonlinear problems with discrete variables: a review. *Structural optimization*, 8(2):69–85, 1994.

[16] Joaquim RRA Martins and Andrew Ning. *Engineering design optimization*. Cambridge University Press, 2021.

[17] J. Lee and S. Leyffer. *Mixed Integer Nonlinear Programming*. The IMA Volumes in Mathematics and its Applications. Springer New York, 2011.

[18] Matthias Köppe. On the complexity of nonlinear mixed-integer optimization. In *Mixed Integer Nonlinear Programming*, pages 533–557. Springer, 2012.

[19] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[20] Liwei Wang, Yu-Chin Chan, Faez Ahmed, Zhao Liu, Ping Zhu, and Wei Chen. Deep generative modeling for mechanistic-based learning and design of metamaterial systems. *Computer Methods in Applied Mechanics and Engineering*, 372:113377, 2020.

[21] Xiang Li, Shaowu Ning, Zhanli Liu, Ziming Yan, Chengcheng Luo, and Zhuo Zhuang. Designing phononic crystal with anticipated band gap through a deep learning based data-driven method. *Computer Methods in Applied Mechanics and Engineering*, 361:112737, 2020.

[22] Tinghao Guo, Danny J Lohan, Ruijin Cang, Max Yi Ren, and James T Allison. An indirect design representation for topology optimization using variational autoencoder and style transfer. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0804, 2018.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[24] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[25] Dassault Systèmes. Solidworks. `http://www.solidworks.com`, May 2021.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[27] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, dec 2015.

[28] Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. Match: Differentiable material graphs for procedural material capture. *ACM Trans. Graph.*, 39(6):1–15, December 2020.

[29] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.

[30] Krishnan Suresh. *Design Optimization using MATLAB and SOLIDWORKS*. Cambridge University Press, 2021.

[31] Hoel Kervadec, Jose Dolz, Jing Yuan, Christian Desrosiers, Eric Granger, and I Ben Ayed. Constrained deep networks: Lagrangian optimization via log-barrier extensions. *CoRR, abs/1904.04205*, 2(3):4, 2019.

[32] Larry J Segerlind. Applied finite element analysis. 1984.

[33] Aaditya Chandrasekhar, Saketh Sridhara, and Krishnan Suresh. Auto: a framework for automatic differentiation in topology optimization. *Structural and Multidisciplinary Optimization*, 64(6):4355–4365, Dec 2021.

[34] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[35] Michael F. Ashby. Chapter 1 - introduction. In Michael F. Ashby, editor, *Materials Selection in Mechanical Design (Fourth Edition)*, pages 1–13. Butterworth-Heinemann, Oxford, fourth edition edition, 2011.

[36] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):1–7, 2016.

[37] Xiou Ge, Richard T Goodwin, Jeremy R Gregory, Randolph E Kirchain, Joana Maria, and Lav R Varshney. Accelerated discovery of sustainable building materials. *arXiv preprint arXiv:1905.08222*, 2019.

[38] Granta Design Limited, UK. CES EduPack software.

[39] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[40] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

[41] Frank Hung-Fat Leung, Hak-Keung Lam, Sai-Ho Ling, and Peter Kwong-Shun Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural networks*, 14(1):79–88, 2003.

[42] Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141. IEEE, 2017.

[43] Xi Peng, Ivor W Tsang, Joey Tianyi Zhou, and Hongyuan Zhu. k-meansnet: When k-means meets differentiable programming. *arXiv preprint arXiv:1808.07292*, 2018.

[44] Longguang Wang, Xiaoyu Dong, Yingqian Wang, Li Liu, Wei An, and Yulan Guo. Learnable lookup table for neural network quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12423–12433, 2022.

[45] Aaditya Chandrasekhar and Krishnan Suresh. Tounn: Topology optimization using neural networks. *Structural and Multidisciplinary Optimization*, 63(3):1135–1149, Mar 2021.

[46] Oliver Giraldo-Londoño, Lucia Mirabella, Livio Dalloro, and Glaucio H Paulino. Multi-material thermomechanical topology optimization with applications to additive manufacturing: Design of main composite part and its support structure. *Computer Methods in Applied Mechanics and Engineering*, 363:112812, 2020.

[47] Koshi Takenaka. Negative thermal expansion materials: technological key for control of thermal expansion. *Science and technology of advanced materials*, 13(1), feb 2012.

[48] Anders Clausen, Niels Aage, and Ole Sigmund. Topology optimization of coated structures and material interface problems. *Computer Methods in Applied Mechanics and Engineering*, 290:524–541, 2015.

[49] Yu-Chin Chan, Daicong Da, Liwei Wang, and Wei Chen. Remixing functionally graded structures: Data-driven topology optimization with multiclass shape blending. *arXiv preprint arXiv:2112.00648*, 2021.