

One-Shot Generation of Near-Optimal Topology through Theory-Driven Machine Learning

Ruijin Cang^a, Hope Yao^a, Yi Ren^{a,*}

^a*Mechanical Engineering, Arizona State University, Tempe*

Abstract

We introduce a theory-driven mechanism for learning a neural network model that performs generative topology design in one shot given a problem setting, circumventing the conventional iterative process that computational design tasks usually entail. The proposed mechanism can lead to machines that quickly response to new design requirements based on its knowledge accumulated through past experiences of design generation. Achieving such a mechanism through supervised learning would require an impractically large amount of problem-solution pairs for training, due to the known limitation of deep neural networks in knowledge generalization. To this end, we introduce an interaction between a student (the neural network) and a teacher (the optimality conditions underlying topology optimization): The student learns from existing data and is tested on unseen problems. Deviation of the student's solutions from the optimality conditions is quantified, and used for choosing new data points to learn from. We call this learning mechanism "theory-driven", as it explicitly uses domain-specific theories to guide the learning, thus distinguishing itself from purely data-driven supervised learning. We show through a compliance minimization problem that the proposed learning mechanism leads to topology generation with near-optimal structural compliance, much improved from standard supervised learning under the same computational budget.

Keywords: Topology Optimization, Meta-learning, Active Learning

1. Introduction

This paper is motivated by the observation that experienced human engineers can quickly generate solutions based on accumulated knowledge, while algorithms are only programmed to solve individual problems from scratch, even when the problems are structured similarly. The lack of ability to generalize from experience makes algorithms often too slow to respond to real-world challenges, especially when a stringent time or cost budget is in place. For

*Corresponding author

Email addresses: cruijin@asu.edu (Ruijin Cang), houpuyao@asu.edu (Hope Yao), yiren@asu.edu (Yi Ren)

URL: designinformatics.github.io (Yi Ren)

example, the design of vehicle body-in-white is often done by experienced structure engineers, since topology optimization (TO) on full-scale crash simulation is not yet fast enough to respond to requests from higher-level design tasks, e.g., geometry design with style and aerodynamic considerations, and thus may slow down the entire design process¹.

Research exists in developing deep neural network models that *learn* to create structured solutions in a *one-shot* fashion, circumventing the need of iterations (e.g., in solving systems of equations [1], simulating dynamical systems [2], or searching for optimal solutions [3, 4, 5]). Learning of such models through data, however, is often criticized to have limited generalization capability, especially when highly nonlinear input-output relations or high-dimensional output spaces exist [6, 7, 8]. In the context of TO, this means that the network may create structures with unreasonably poor physical properties when it responds to new problem settings. More concretely, consider a topology with a tiny crack in one of its trusses. This design would be far from optimal if the goal is to lower compliance, yet standard data-driven learning mechanisms do not prevent this from happening, i.e., they don't know that they don't know (physics).

Our goal is to create a learning mechanism that knows what it does not know, and thus can self-improve in an effective way. Specifically, we are curious about how physics-based knowledge, e.g., in the forms of dynamical models, theoretical bounds, and optimality conditions, can be directly injected into the learning of networks that perform one-shot solution generation. We call this type of learning mechanisms "*theory-driven*". This idea is particularly enchanting as such knowledge widely exists in engineering disciplines and is often differentiable, which is important for gradient-based learning algorithms.

As an initial attempt in this direction, this paper will focus on TO, and uses a compliance minimization problem [9] as a running example. To overview, we introduce an interaction between a student (the neural network) and a teacher (the optimality conditions underlying the TO): The student learns from existing data and is tested on unseen problems. Deviation of the student's solutions from the optimality conditions is quantified, and used to choose new data points for the student to learn from. We use this mechanism to learn a solution generator from a distribution of compliance minimization problems with different load settings. We show that the proposed method has significantly better performance in predicting the optimal topologies for unseen loads, than using a static data collection, under the same computational cost for data acquisition.

The proposed method will enable learning of solution generators for engineering problems that need to be solved repetitively with variations in their settings, e.g., optimal design (structures, networks, geometries, etc.) as an inner loop of a larger-scale system design, or real-time optimal control with context-dependent parameters. The generated solutions kick-start iterative solvers with good initial guesses and thus shorten the design (solution searching) processes.

The rest of the paper is structured as follows: In Sec. 2 we review related work at the intersection of generative design and machine learning, and highlight the new contributions

¹The authors learned about this challenge through personal communication with an engineer at a major car manufacturer.

from this paper. We then revisit in Sec. 3 the compliance minimization problem, which will be used as a running example throughout the paper, and explain the need for learning one-shot solution generators. Sec. 4 introduces the proposed learning mechanism, which will then be validated in Sec. 5 against two benchmark methods on the running example. Sec. 6 discusses the connection between the proposed method and recent developments in machine learning, and suggests future directions. Sec. 7 concludes the paper.

2. Related Work

Existing research on generative design answer three types of questions: (1) What is the design *representation*? (2) What is the *goodness* measure of a design? And based on these two, (3) how do we *search* for a good design? Challenges in answering these questions include *high-dimensional or ill-defined design spaces* such as for topologies [10, 11], material microstructures [12, 13, 14], or complex geometries [15, 16], *expensive evaluations* of designs and their sensitivities, e.g., due to model nonlinearity [17, 18], coupled materials or physics [19, 20, 21], or subjective goodness measures [22, 23], or *search inefficiency* due to the absence of sensitivities [24, 25, 26] or the existence of random variables [27].

This paper takes a different angle by focusing on design tasks for which answers to the above three questions exist, yet applying them to real-world design tasks is computationally unaffordable, thus the need for algorithms that learn to *improve* their efficiency through problem solving. Below we review related work, and explain how this paper is similar and different from them.

To start with, the task of improving learning efficiency through experience is known as *meta-learning* [28, 29] (with close connections to transfer learning [30, 31, 32] and life-long learning [33]). While there exists a broad range of problem settings within the literature of meta-learning, the setup commonly involves a student and a learning mechanism that specifies how the student updates its way of learning or solving problems. The student outputs a solution for every input problem, for example, the input can be a labeled dataset and the output a classifier that explains the data, or the input an optimization problem and the output an optimal solution. In the former case, the student updates its way of learning a classifier [34] or its hypothesis space of classifiers [28]; in the latter, it updates its gradient [35] or non-gradient [36] search strategies. These updates are governed by the learning mechanism, the design of which is driven by a goodness measure of the student, e.g., the generalization performance of a classifier or the convergence rate of an optimization solver.

The problem of learning to generate designs can be cast as a meta-learning problem, where the student is the generator that takes in settings of the design problem (e.g., the distribution and magnitude of loads, the material properties, or the boundary conditions for a topology optimization problem) and outputs a design solution, whereas the learning mechanism updates the architecture or parameters of the generator.

It is worth noting two differences between the proposed method and contemporary meta-learning. First, typical mechanisms proposed in meta-learning literature are iterative, e.g., in forms of recurrent neural networks [34, 35, 37]. This choice of model is due to the iterative

nature of problem solving and the need of memory in decision making during the iteration. In contrary, we model the transition from problem settings to solutions using a feedforward neural network, which is one-shot in nature. We made this choice based on the finding that optimal solutions to a distribution of TO problems often form a continuous manifold (see Sec. 3 for details), which suggests that directly learning the manifold through a feedforward network might be achievable, in which case we circumvent the challenges from modeling iterative solvers.

Secondly, we note that existing meta-learning tasks are often set in contexts where large data acquisition is affordable. This does not hold in our case, since finite element analysis and design sensitivity analysis are costly yet necessary for training the neural network. This requires us to focus on adaptively choosing data points to improve the generator, thus rendering our approach somewhat more similar to active learning [38, 39], where the goal is to improve data efficiency of learning by querying data based on the learned model. Nonetheless, active learning strategies are usually statistics-based. For example, in the context of classification, new data points are chosen based on uncertainty of their predicted labels [40] or their predicted contribution to the prediction error of the learned model [41]. These methods, however, are not suitable for our case since we care about the physical optimality of the generated topologies, rather than the pixel-wise matching between the generated topologies and the corresponding true optima (e.g., l2-norm or cross-entropy defined on image differences often used as metrics of prediction error). The method we introduce is thus distinctively different from active learning, as we choose data based on the optimality conditions of TO, which are problem-dependent and theory-driven.

Lastly, the proposed learning method is aligned with the recent surge of machine learning techniques with integrated physics knowledge. Among this body of work, [42] proposes to learn a neural network for predicting intrinsic physical properties of objects with the assistance of a physical simulator that computes object interactions based on the predicted properties. [43] developed an encoder that computes object positions and velocities from images of objects, by enforcing these properties to be compliant with common sense. Similarly, [44] proposed a physics-based regularization to learn object trajectories and human movements from videos. Instead of being taught physics, [45] demonstrated a grounded way for machines to acquire an intuition of the physical world through reinforcement learning. Our paper is similar to those learning mechanisms with injected physics constraints, while employing adaptive sampling rather than batch-mode training to avoid expensive simulations. In addition, the use of theory-driven constraints for learning also aligns our method with Q learning, where values are learned to satisfy Bellman’s equation.

3. Problem Statement

3.1. One-shot solution generator

We define a one-shot solution generator as a feedforward neural network $\mathbf{x} = g(\mathbf{s}, \boldsymbol{\theta})$ that computes a solution $\mathbf{x} \in \mathcal{X}$ (e.g., a topology) given problem settings $\mathbf{s} \in \mathcal{S}$ (e.g., loads) and network parameters $\boldsymbol{\theta}$, where \mathcal{X} is a solution space, and \mathcal{S} is a problem space (e.g., a space of locations and orientations of loads). The generator is one-shot in the sense that computing

\mathbf{x} through g is much less expensive than using an iterative algorithm. We also define the generalization performance of the generator (denoted by $F(\boldsymbol{\theta})$) as the expected performance of its solutions over a distribution of problems specified by a probability density function $p(\mathbf{s})$:

$$F(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{s})} f(g(\mathbf{s}, \boldsymbol{\theta}), \mathbf{s}), \quad (1)$$

where $f(\mathbf{x}, \mathbf{s})$ measures the performance of \mathbf{x} under \mathbf{s} .

3.2. The compliance minimization problem

We now review the mechanical compliance minimization problem introduced in [9] to substantiate $f(\mathbf{x}, \mathbf{s})$. In this context, a solution \mathbf{x} is a N -by-1 vector with values between 0 and 1, elements of which control a density vector, denoted as $\boldsymbol{\rho}$, of the corresponding physical elements of a meshed structure through the following relation:

$$\rho_e = \frac{\tanh(\beta/2) + \tanh(\beta(\tilde{x}_e - 0.5))}{2 \tanh(\beta/2)}, \quad (2)$$

where x_e and ρ_e are the elements of \mathbf{x} and $\boldsymbol{\rho}$, respectively, for $e = 1, \dots, N$. The shape parameter β controls the sharpness of the transition to the density ρ_e from a filtered variable \tilde{x}_e , which is a weighted average of neighbours of x_e :

$$\tilde{x}_e = \frac{\sum_{i \in \mathcal{M}_e} \omega_{i,e} x_i}{\sum_{i \in \mathcal{M}_e} \omega_{i,e}}, \quad (3)$$

where \mathcal{M}_e is the set of neighbours of element e , and weights $\omega_{i,e}$ are defined as

$$\omega_{i,e} = 1 - \|z_i - z_e\|_2 / r_e, \quad (4)$$

with z_i the coordinates of meshed element i , and r_e the filter radius. In TO, this filter (Eq. (3)) is used to prevent convergence to impractical checkerboard topologies [9].

The connection from \mathbf{x} to the global stiffness matrix of the topology (denoted as \mathbf{K}) can be established through the density vector $\boldsymbol{\rho}$. Given loads \mathbf{s} and boundary conditions, the displacement \mathbf{u} of the structure can be found by solving $\mathbf{K}\mathbf{u} = \mathbf{s}$, under the assumption that \mathbf{K} is independent of \mathbf{u} (e.g., linear elastic material and small displacement). The compliance minimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{s}) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} \\ \text{subject to} \quad & \mathbf{K} \mathbf{u} = \mathbf{s}, \\ & x_e \in [0, 1], \quad \forall e \in \{1, \dots, N\}, \\ & g_0(\mathbf{x}) = \frac{1}{N} \sum_e \rho_e - \alpha \leq 0, \\ & g_1(\mathbf{x}) = \|\bar{\rho}\|_p - \alpha \leq 0. \end{aligned} \quad (\text{TO})$$

Here the constraint g_0 (g_1) limits the global (local) density of the structure to be lower than a threshold α . $\|\mathbf{x}\|_p = (\frac{1}{N} \sum_e x_e^p)^{1/p}$ is the p -norm defined on \mathbb{R}^N . p is set to 16 following

[9] so that $\|\mathbf{x}\|_p$ approximately computes the maximum of $|\mathbf{x}|$ while being differentiable. The averaged local density $\bar{\rho}_e = (\sum_{i \in \mathcal{N}_e} \rho_i) / (\sum_{i \in \mathcal{N}_e} 1)$ is defined on the neighborhood $\mathcal{N}_e = \{i \mid \|x_i - x_e\|_2 \leq R_e\}$ with radius R_e . Note that R_e for the local density constraint is different from the filter radius r_e . The optimality conditions of (TO) are listed as follows

$$\begin{aligned}
\nabla_{\mathbf{x}} L &:= -\mathbf{u}^T \frac{\partial \mathbf{K}}{\partial \mathbf{x}} \mathbf{u} + \mu_0 \nabla_{\mathbf{x}} g_0 + \mu_1 \nabla_{\mathbf{x}} g_1 + \boldsymbol{\mu}_u^T - \boldsymbol{\mu}_l^T = \mathbf{0}^T \\
x_e &\in [0, 1], \quad \forall e \in \{1, \dots, N\}, \\
g_0 &\leq 0, \quad g_1 \leq 0, \quad \mu_0 g_0 = 0, \quad \mu_1 g_1 = 0, \\
\boldsymbol{\mu}_l^T \mathbf{x} &= 0, \quad \boldsymbol{\mu}_u^T (\mathbf{x} - \mathbf{1}) = 0 \\
\mu_0 &\geq 0, \quad \mu_1 \geq 0, \quad \boldsymbol{\mu}_u \geq \mathbf{0}, \quad \boldsymbol{\mu}_l \geq \mathbf{0},
\end{aligned} \tag{5}$$

where $\nabla_{\mathbf{x}} y(\mathbf{x}, \cdot)$ is the partial derivative of function y with respect to variables \mathbf{x} , and is defined as a row vector. Finding a solution to comply with Eq. (5) can be done through a gradient-based solver, e.g., an augmented Lagrangian algorithm (see Sec. 7). However, we need to note that the computational cost for converging to an optimal solution usually does not scale well. In particular, solving Eq. (5) with a problem size $N = 4800$ requires on average around 5000 finite element analyses (i.e., computing \mathbf{u} from $\mathbf{K}\mathbf{u} = \mathbf{s}$).

3.3. Learning a solution generator

With the above setup, the problem of learning a one-shot solution generator can be formulated as follows:

$$\begin{aligned}
\min_{\boldsymbol{\theta}} \quad & F(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{s})} f(\mathbf{x}, \mathbf{s}) \\
\text{subject to} \quad & \mathbf{x} = g(\mathbf{s}, \boldsymbol{\theta}), \\
& \mathbf{K}\mathbf{u} = \mathbf{s}, \\
& f(\mathbf{x}, \mathbf{s}) = \frac{1}{2} \mathbf{u}^T \mathbf{K}\mathbf{u} \\
& g_0(\mathbf{x}) = \frac{1}{N} \sum_e \rho_e - \alpha \leq 0, \\
& g_1(\mathbf{x}) = \|\bar{\rho}\|_p - \alpha \leq 0.
\end{aligned} \tag{P}$$

We will force network outputs to be within $(0, 1)^N$ by attaching sigmoid activations to its output layer.

4. Learning with a Physics-based Criterion

(P) can be solved by matching the input-output pairs of the generator to a dataset $\mathcal{D} = \{\mathbf{x}_i^D, \mathbf{s}_i^D\}_{i=1}^n$ where \mathbf{x}_i^D are optimal for \mathbf{s}_i^D . This leads to the data-driven learning formulated as follows:

$$\min_{\boldsymbol{\theta}} \sum_{\mathcal{D}} \|g(\mathbf{s}_i^D, \boldsymbol{\theta}) - \mathbf{x}_i^D\|_2^2. \tag{P1}$$

As we reviewed in the last section, collecting \mathcal{D} can be costly due to the iterative nature of solving the topology optimization problem (TO). On the other hand, checking the compliance of an arbitrary topology \mathbf{x} to the optimality conditions (Eq. (5)) only requires solving $\mathbf{K}\mathbf{u} = \mathbf{s}$ once. This finding indicates that the optimality conditions may offer affordable means to identify new data point that will most effectively improve the generalization performance of g . Specifically, we define the deviation of solution \mathbf{x} from the optimality conditions as

$$d(\mathbf{x}, \boldsymbol{\mu}) = \|\nabla_{\mathbf{x}}L\|_2^2 + w_0g_0^2 + w_1g_1^2 \quad (6)$$

where the algorithmic parameters w_0 and w_1 weight the penalties on constraints g_0 and g_1 , respectively. One issue in evaluating d is that we do not know the values of the Lagrangian multipliers (which are denoted by $\boldsymbol{\mu}^T = [\mu_0, \mu_1, \boldsymbol{\mu}_l^T, \boldsymbol{\mu}_u^T]$) before solving the problem. To this end, we propose to find $\boldsymbol{\mu}^*$ that minimizes the deviation of $\nabla_{\mathbf{x}}L$ from $\mathbf{0}$ subject to their constraints from (P):

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \|\nabla_{\mathbf{x}}L\|_2^2 \\ \text{subject to} \quad & \boldsymbol{\mu} \geq 0, \boldsymbol{\mu}^T \mathbf{g} = 0. \end{aligned} \quad (\text{P2})$$

By solving (P2) for all \mathbf{s} in a validation set \mathcal{S}_v , we can then choose a new training data point $(\mathbf{s}^*, \mathbf{x}^*)$ for which the minimal deviation $d(g(\mathbf{s}^*, \boldsymbol{\theta}), \boldsymbol{\mu}^*)$ is the largest among \mathcal{S}_v . It is worth noting that (P2) is a $(2N + 2)$ -dimensional quadratic programming problem and can be solved efficiently using standard solvers (e.g., sequential quadratic programming).

The learning algorithm can now be summarized in Alg. 1. Details on setting the initial training set \mathcal{S}_0 , the validation set \mathcal{S}_v , the computational budget B , and the budget lower bound b will be introduced along the case studies in Sec. 5.

5. Case Studies

This section presents two case studies where we demonstrate the superior learning efficiency of the proposed algorithm in comparison with two benchmark mechanisms. The first benchmark uses a static dataset ($\mathcal{D}_{\text{static}}$) for training, and is denoted as *Benchmark I*. The second benchmark, denoted as *Benchmark II*, is similar to the proposed approach, but chooses data points using a different heuristic. For all three learning mechanisms (Benchmarks I, II, and ours), the topology optimization problem is solved by an Augmented Lagrangian algorithm, details of which is deferred to the appendix.

5.1. The heuristic of Benchmark II

In Benchmark II, we evaluate the performance of the generator g by measuring the difference between the compliance produced by g and the predicted compliance based on the training data \mathcal{D} :

$$d_h(g(\mathbf{s}_i, \boldsymbol{\theta}^*)) = |f(g(\mathbf{s}_i, \boldsymbol{\theta}^*), \mathbf{s}_i) - \hat{f}(\mathbf{s}_i)|, \quad (7)$$

where \mathbf{s}_i comes from the validation set \mathcal{S}_v , and \hat{f} is an ordinary least square model that fits to $\{(\mathbf{s}_i^*, f(\mathbf{x}_i^*, \mathbf{s}_i^*)) \mid (\mathbf{x}_i^*, \mathbf{s}_i^*) \in \mathcal{D}\}$. In this study, polynomial models are used for curve fitting. The validation data point with the highest value of d_h is chosen, and its true optimal topology is then computed and used to improve the generator.

Algorithm 1: Theory-driven learning

input : Problem distribution $p(\mathbf{s})$
output: Learned model parameters $\boldsymbol{\theta}^*$

- 1 Draw initial problem set \mathcal{S}_0 from $p(\mathbf{s})$;
- 2 Find optimal \mathbf{x}_i for each \mathbf{s}_i in \mathcal{S}_0 ;
- 3 Initialize dataset $\mathcal{D} = \mathcal{D}_0 = \{(\mathbf{x}_i, \mathbf{s}_i)\}_{i=1}^{|\mathcal{S}_0|}$;
- 4 Draw validation problems \mathcal{S}_v from $p(\mathbf{s})$;
- 5 Set computation budget $B = B_0$ and budget lower bound b ;
- 6 **while** $B > b$ **do**
- 7 Update $\boldsymbol{\theta}^*$ by solving (P1) based on \mathcal{D} ;
- 8 Calculate $\boldsymbol{\mu}_i^*$ and $d(g(\mathbf{s}_i, \boldsymbol{\theta}^*), \boldsymbol{\mu}_i^*)$ for all \mathbf{s}_i in \mathcal{S}_v based on Eq. (6);
- 9 Find $\mathbf{s}^* \in \mathcal{S}_v$ with the highest d value;
- 10 Derive \mathbf{x}^* for \mathbf{s}^* by solving (TO);
- 11 Record δB as the number of $\mathbf{K}\mathbf{u} = \mathbf{s}$ solved in solving (TO) and computing Eq. (6);
- 12 Update the budget $B = B - \delta B$;
- 13 $\mathcal{D} = \mathcal{D} + (\mathbf{x}^*, \mathbf{s}^*)$, $\mathcal{S}_v = \mathcal{S}_v - \mathbf{s}^*$;
- 14 **end**

5.2. Study setups

The topology optimization problems to be solved follow (TO). Two cases are created to demonstrate the scalability of the proposed method. In Case 1, the input \mathbf{s} represents a single load applied to a 2D structure represented by a 40-by-120 mesh, see Fig. 1a. In this case, the input to the generator is the angle of the load, uniformly distributed in between 0 and π , i.e., $p(\mathbf{s}) = 1/\pi$. In Case 2, \mathbf{s} encodes (1) the x- and y-coordinates of the loading, which is drawn uniformly from all nodes in the highlighted area in Fig. 1b, and (2) the direction of the point load uniformly drawn from 0 to 2π .

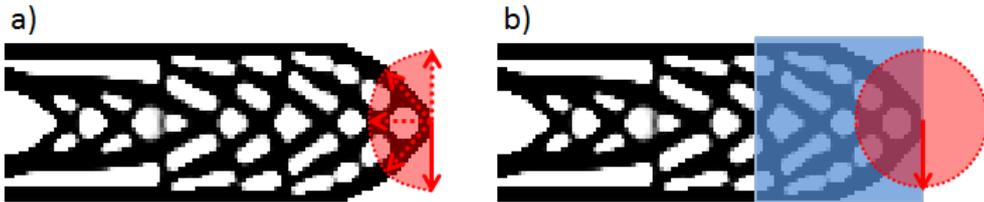


Figure 1: Sample solutions for (a) Case 1, where a point load is applied to the middle node at the tip of the cantilever beam, with directions uniformly distributed in $[0, \pi]$, and (b) Case 2, where the point load is applied to a node in the highlighted square area which occupies one-third of the design space, with its direction drawn from $[0, 2\pi]$

To compare the three learning mechanisms, each is executed 10 times to account for the randomness in the sampling of the initial dataset \mathcal{D}_0 (and $\mathcal{D}_{\text{static}}$ in the case of Benchmark

I) and the validation set at each iteration. The generalization performance is measured by

$$F(\boldsymbol{\theta}^*) = \sum_{\mathbf{s}_i \in \mathcal{S}_t} f(g(\mathbf{s}_i, \boldsymbol{\theta}^*), \mathbf{s}_i), \quad (8)$$

where \mathcal{S}_t is a separate test set drawn from $p(\mathbf{s})$. For Case 1, the sample sizes are $|\mathcal{S}_0| = 5$, $|\mathcal{S}_v| = 100$, $|\mathcal{S}_t| = 100$, and $|\mathcal{D}_{\text{static}}| = 16$. For Case 2, since the problem space is much larger, the sample sizes are set to $|\mathcal{S}_0| = 1000$, $|\mathcal{S}_v| = 6000$, $|\mathcal{S}_t| = 1000$, and $|\mathcal{D}_{\text{static}}| = 7000$. In addition, since validating a large number of inputs becomes expensive, we uniformly sample 100 validation points from $|\mathcal{S}_v|$ to perform active learning in each iteration. All previously sampled points will be removed from \mathcal{S}_v . For both cases, the computational budget B_0 is set as $b_{\min}|\mathcal{D}_{\text{static}}|$, where b_{\min} is the minimal solution cost among all problems sampled for Benchmark I. The budget lower bound b is set to the maximal cost among the same set of problems. This setting ensures that the adaptive methods (Benchmark II and the proposed method) will always use less computational resource than the static method for topology optimization, thus creating a comparison in favor of the latter. For the following results, we set $w_0 = w_1 = 1$ for the proposed method. A full parametric study on these hyper-parameters has not been conducted, yet the effectiveness of the current setting is validated (Sec. 5.4).

5.3. Architectures of the solution generators

The architectures of the solution generators are summarized in Fig. 2. For Case 1, we use a two-dimensional input to represent the x- and y-components of the point load. For Case 2, we use a three-dimensional input to represent the x, y location and the orientation of the point load. The choice of these input representations are based on empirical tests of the generalization performance of the learned models. In particular, we found in Case 2 that using a sparse load representation as network inputs will lead to much worse generation performance for all learning mechanisms.

5.4. Results

Case 1 results. For Case 1, Fig. 3a compares the compliance of the topologies generated by all three learning mechanisms with the ground truth for all test inputs; Fig. 3b reports the corresponding compliance gaps produced by these mechanisms. The generalization performance of a learning mechanism is measured by the mean and the standard deviation of the average compliance gaps across all test inputs. The result shows that the proposed mechanism outperforms the benchmarks at predicting optimal topologies for unseen loading conditions in a low-dimensional case. To further demonstrate the difference between the three mechanisms, we visualize and compare generations for four test loads in Fig. 1. The loading directions are marked in the last row of the figure, which are 0.3π , 0.4π , 0.45π , 0.55π respectively. The resultant compliance values are shown at the bottom of each topology.

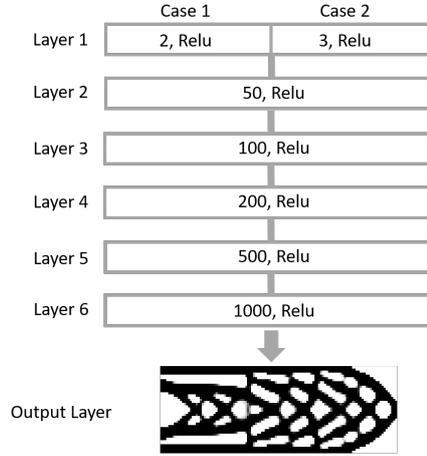


Figure 2: Architectures of the solution generators

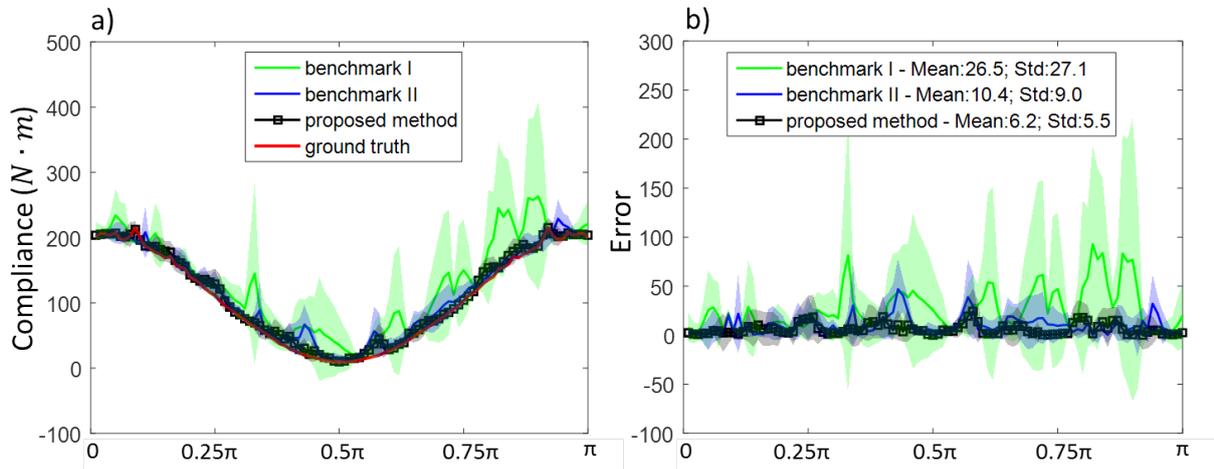


Figure 3: (a) Comparison between the compliance of the ground truth topologies and those from the generated solutions from all three learning mechanisms. 10 independent experiments are conducted for each learning mechanism. The means and the standard deviations are reported. (b) The absolute compliance gaps.

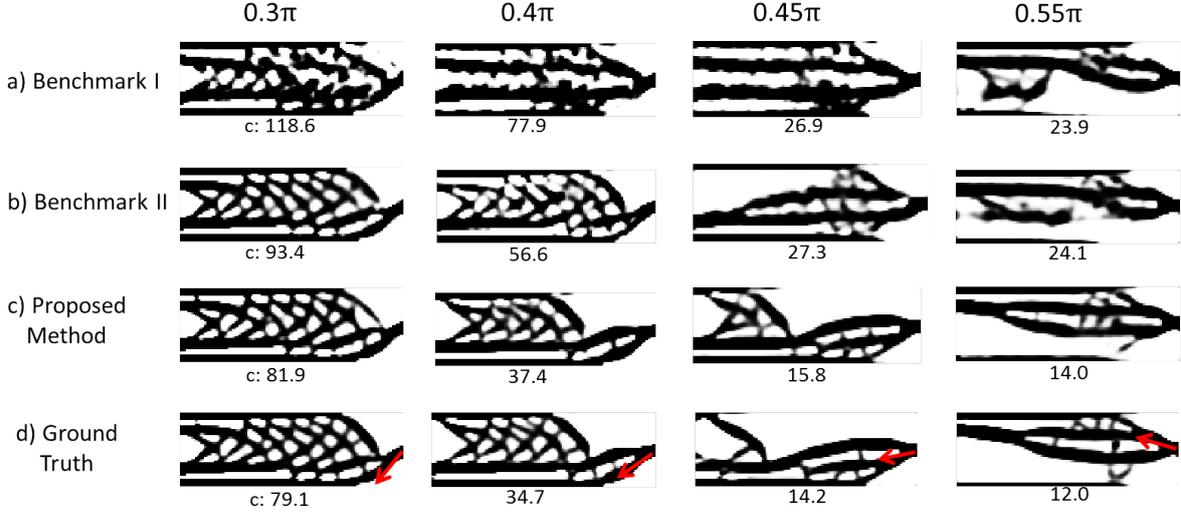


Figure 4: Case 1 topologies predicted by (a) Benchmark I, (b) Benchmark II, (c) the proposed method. (d) Ground truth computed using the Augmented Lagrangian solver. Compliance values are shown at the bottom of each topology. Loading directions are marked as the arrows.

Challenges in Case 2. Case 2 examines the performance of the proposed mechanism under a higher-dimensional and larger input space. We notice that the learning becomes significantly more challenging in this case. (see Sec 6). Specifically, for all three algorithms under the same budget (which is equivalent to solving 7000 TO problems), there exist inputs for which the generated topologies have significantly larger compliance than the ground truth. We mark test data points with a compliance gap of over 1000 as failed designs. For Benchmark I, II and the proposed method, the mean failure rates over the entire test are 14%, 0.8%, and 0.6% respectively. Some failed generations are shown in Fig. 5 along side the corresponding ground truth.

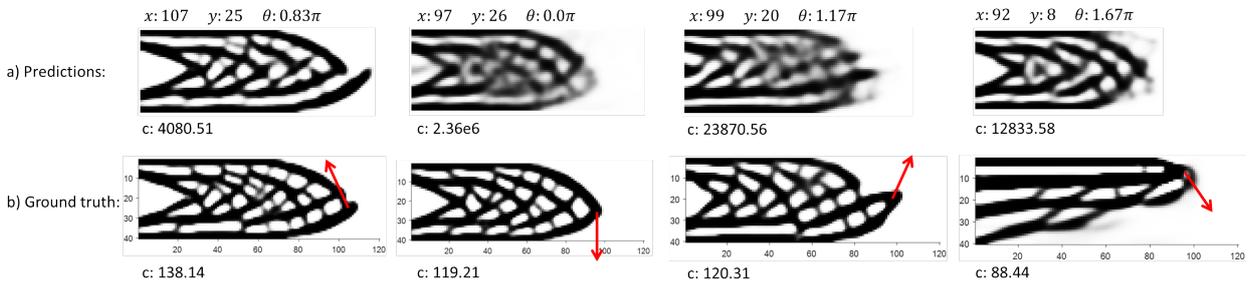


Figure 5: Example of failed generations with significant larger compliance than the corresponding ground truth. Loading conditions are marked at the top and compliance at the bottom. Some generations are close to the ground truth in the pixel space but miss critical elements, such as the first three. Some of the generations are simply off, such as the last one.

Table 1: Comparisons on the mean count of failures from 1000 test data points and the median compliance gaps for Case 2

	Failure rate: mean (std)	Median compliance gap: mean (std)
Benchmark I	5.12% (1.2%)	22.12 (3.34)
Benchmark II	0.64% (0.42%)	6.64 (0.87)
Propose method	0.16% (0.09%)	5.2 (0.6)

An augmented learning objective. To alleviate this issue, we introduce design sensitivity $\nabla_{\mathbf{x}}f$ at the optimal solution as a weighting factor of the learning loss in (P1), based on the insight that structural elements with higher sensitivity contribute more to the compliance and thus should have higher priority during model training. Concretely, the training problem at each iteration of the active learning process is now formulated as

$$\min_{\boldsymbol{\theta}} \sum_{(\mathbf{s}_i, \mathbf{x}_i) \in \mathcal{D}} \|g(\mathbf{s}_i, \boldsymbol{\theta}) - \mathbf{x}_i\|_{\boldsymbol{\Lambda}_i}^2, \quad (9)$$

where $\boldsymbol{\Lambda}_i = \text{diag}([\lambda_{i,1}, \dots, \lambda_{i,N}])$ is a diagonal weighting matrix, $\lambda_{i,e} = \frac{\nabla_{\mathbf{x}}f - \min(\nabla_{\mathbf{x}}f)}{\max(\nabla_{\mathbf{x}}f) - \min(\nabla_{\mathbf{x}}f)}$, and $\|\mathbf{x}\|_{\boldsymbol{\Lambda}}^2 = \mathbf{x}^T \boldsymbol{\Lambda} \mathbf{x}$. It is important to note that $\boldsymbol{\Lambda}_i$ is a byproduct of computing \mathbf{x}_i and does not cost extra budget.

Case 2 results. By introducing this augmentation, the mean rate of failures drops to 5.12%, 0.64%, and 0.16% for Benchmark I, II, and the proposed method, respectively. Since the compliance distributions are far from normal due to the failed designs, we report the means and standard deviations of the median compliance gaps from all experiments and learning mechanisms instead. These results are summarized in Table 1. As a demonstration, we compare in Fig. 6 the generations from all mechanisms under four test settings. The proposed method has the closest compliance to the ground truth.

Validity of hyper-parameter settings. For both case studies we set the weights for the local and global constraints (w_0 and w_1 in Eq. 6) to be $w_0 = 1$ and $w_1 = 1$. To validate this setting, we monitor the violation to the constraints by solutions generated for all test inputs. For a proper measure of constraint satisfaction, test cases where constraints are satisfied will be ignored in the calculation of the mean violation. It is noted that ground truth topologies have non-zero violation due to the non-zero error thresholds set in the augmented Lagrangian algorithm. In Case 1 (Case 2), the mean violation to the global volume fraction constraint is $g_0 = 4.89\%$ (3.0%) for the ground truth and $g_0 = 8.76\%$ (1.6%) for the generated solutions. In both cases, no violation to the local constraints are observed for either the ground truth or the generated solutions. This result indicates that the setting of the hyper-parameters leads to the learning of a generator with reasonable compliance to the global and local volume fraction constraints. A parametric study is yet needed to fully characterize the tradeoff between the learning of effective topologies and that of constraint compliance by tuning the hyper-parameters.

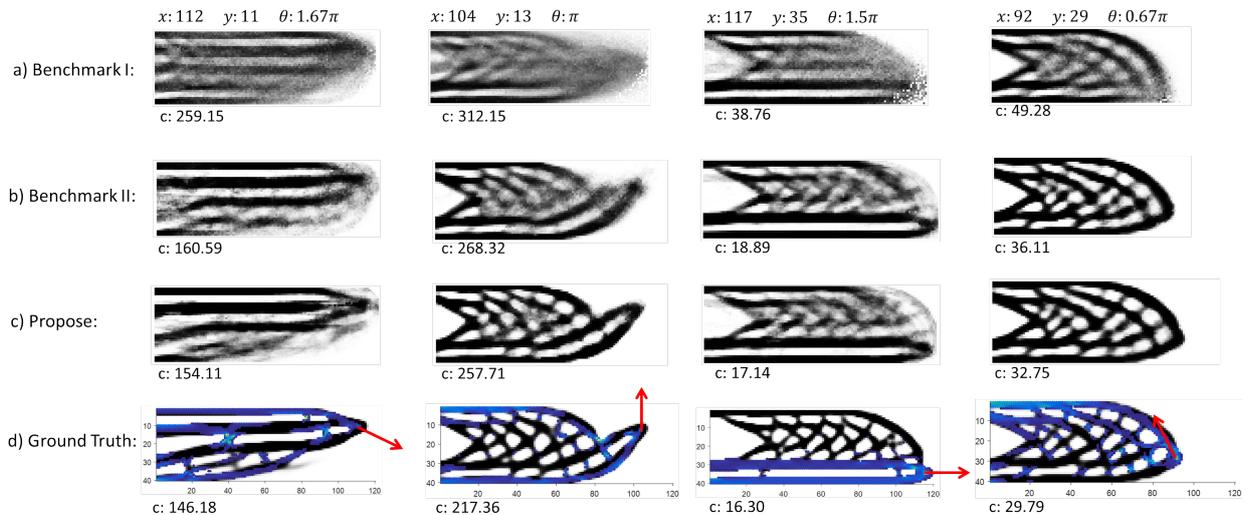


Figure 6: Case 2 optimal topologies predicted by (a) Benchmark I, (b) Benchmark II, and (c) the proposed method. (d) The ground truth topologies with design sensitivity information: cold (warm) color indicates lower (higher) design sensitivity. The point loads are marked as arrows with detailed input values reported at the top of the figure. The compliance values are shown at the bottom of each topology. Best viewed online.

Comparison on solution generation speed. Lastly, we shall note that the computation time required for generating solutions through the learned model is negligible (in the order of 10^{-2} seconds per topology on Intel Xeon CPU E5-1620 @ 3.50GHz) compared with that through a TO solver (in the order of 10^2 to 10^3 seconds per topology). It is also worth mentioning that the run-time cost of the model does not include the data acquisition cost for training the model, and the above comparison only makes sense for cases where the expected amount of computation (for all TO problems to be solved) is much larger than what is required for training the network. We acknowledge, nonetheless, that quantifying the total computational cost of a design task is challenging (e.g., due to unknown designer preferences), and theoretical upper bounds on the data size for neural networks are yet to be established. However, addressing these issues is beyond the scope of this paper.

6. Discussion

We now discuss the remaining issues related to learning manifolds of optimal solutions using neural networks.

6.1. Curse of dimensionality and potential solutions

So far we assumed that the pre-defined network architecture is able to capture the underlying manifold in high-dimensional spaces. As we see from the two cases, increasing the latent dimension of the manifold from one to three already raised the data demand significantly, indicating that the manifold of optimal topologies underlying Case 2 is much more

“complex” than that of Case 1. Indeed, from Narayanan and Mitter [46], the sampling complexity of learning a manifold (to within a specified tolerance) is exponential on the intrinsic dimension (the dimension of the input), linear on the intrinsic volume (the size of the input space), and polynomial on the curvature of the manifold.

This leads to two legitimate concerns. The first regards model sufficiency: pre-defining the network architecture of the generator could be a stab in the dark when a new manifold is to be learned, as we do not know whether the network is sufficiently flexible to fit to the manifold. The second regards data sufficiency: real-world design problems may have solution manifolds that are too data-demanding for active learning alone to handle.

Lei et al. [47] recently proposed an approach to the first concern in the context of piecewise linear networks (with ReLU activations). It is shown that both the manifold and the network complexity can be measured by the number of polyhedral cells induced respectively by the geometry of the manifold and the architecture of the network. Further studies following these complexity measures may lead to protocols for determining the network architecture of the generator before learning of generator weights takes place, essentially by computing an upper bound on the number of planes needed to locally and linearly approximate the underlying manifold. An efficient algorithm for doing so, however, is yet to be developed.

The second concern, however, is more critical to the application of the proposed method. One potential solution is based on the insight that the governing equations for training can be derived at arbitrary spatial resolution of the structure. While the intrinsic dimension and volume do not change across resolutions, we hypothesize that lowering the resolution will reduce the manifold curvature, and thus reduce the sampling complexity. In the context of TO, with lower resolution of the structure (i.e., less elements), we expect an easier learning problem. Based on this hypothesis, it is possible that a hierarchical network architecture can be learned progressively to alleviate the curse of dimensionality: At each spatial resolution level, we learn a generator that predicts the transition from coarse solutions to the ones that satisfy the optimality conditions at this level. The coarse solutions are proposed by the generator learned from the lower-level resolution. The investigation of this approach will be reported in a separate paper. It should be reiterated that despite the inevitable scalability challenge, the value of learning models through domain-specific theories is clearly demonstrated in this study.

6.2. From theory-driven data selection to full theory-driven learning

Another direction to explore is based on the note that the proposed method does not solve the learning problem (P) directly. Rather, we collect true solutions and fit a generator (i.e., a neural network) to it, with the hope that by intelligently collecting true solutions on fly, the fit model will effectively converge to the true solution manifold governed by the optimality conditions. An interesting question is whether solving (P) directly through a gradient-based method can be achieved and will be more effective than the presented method. If we consider the presented learning mechanism as theory-driven data selection, then solving (P) directly will be full theory-driven learning. More concretely, in each iteration of the learning, we would need to solve a batch of TO problems *partially*, i.e., finding feasible topologies that

reduce the violation to the optimality conditions, and use the resultant changes in topologies to update the solution generator. The key difference between the presented method and a full theory-driven learning mechanism is that while the former guarantees optimality of known solutions as long as the network is flexible enough to fit through these solutions, the latter does not have such a guarantee at any time during the training; instead, it requires less cost per iteration (since it does not completely solve TO problems), and may afford more iterations (batches of TO problems). Fig. 7 visualizes the difference between the two using a simple 2D illustration, where the circle is the unknown solution manifold, the curve represents a solution generator, and the dots are sampled solutions.

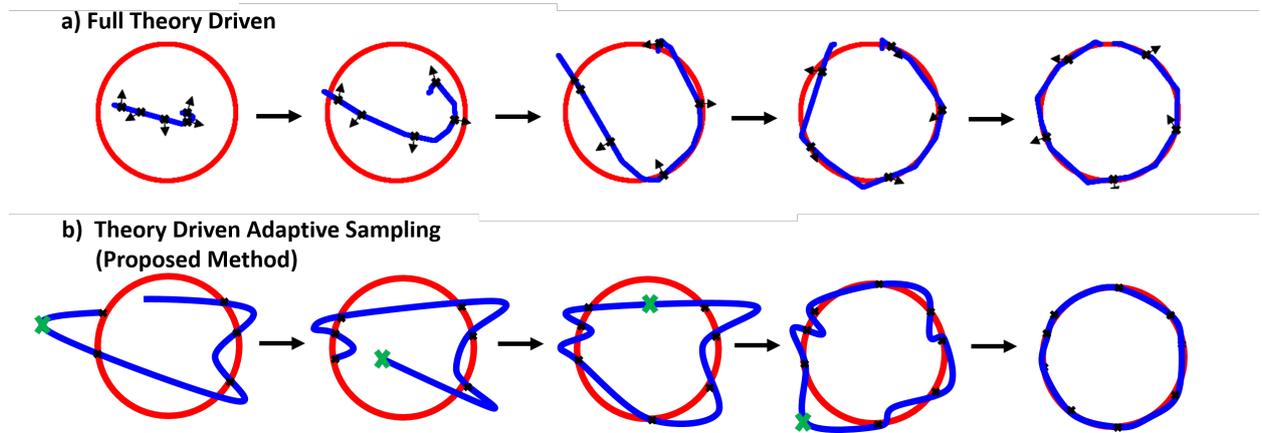


Figure 7: (a) Full theory-driven learning: In each iteration, random inputs are picked to compute the improvement directions of the model towards the ground truth. (b) Theory-driven adaptive sampling (this paper): Sample inputs are selected based on their violation to the optimality conditions. The corresponding ground truth (the crosses) are revealed, which informs the improvement directions of the model. Best viewed online.

6.3. Interpretability of the generators

One additional challenge we face is the interpretation of the learned generator. Given the fact that the generators perform reasonably well in the two cases, one would like to visualize what they learn, e.g., local structural features that serve as puzzle pieces and lead to low-compliance structures when assembled. However, an investigation into the learned generators shows that such interpretable knowledge are not evident from the visualization of the network parameters. One potential reason is that the arbitrarily chosen network architecture may introduce confounding hidden nodes that decompose interpretable features that could have existed. One solution could be to impose regularization on all network parameters to be learned. Yet this will introduce more hyper-parameter tuning as a result.

6.4. Value of learning manifolds of optimal designs to designers

Last but not least, we shall come back to discuss and reiterate why and when learning manifolds of optimal designs has a value to designers. As we discussed in the introduction, there are cases where a large number of topology optimization problems (or other forms

of optimization problems) need to be solved with only parametric differences. Such cases include when the TO is nested in a larger-scale system-level optimization, or when problem settings of the TO are required to be explored by human designers. In addition, designers may solve problems without knowing that similar ones have been previously solved by others. The proposed learning mechanism would not only allow a machine to accumulate and learn from solutions to similar problems, but also *effectively* practice by itself to reinforce its intuition at quickly solving common sets of problems, and thus may reduce the computational and energy cost of design tasks of growing complexity.

On the other hand, the learning itself requires solving more problems during adaptive sampling. Hence, it would be ideal to understand whether the expected total cost of solving a set of problems in the future surpasses the potential cost of learning. If so, the learning mechanism would have a value. The comparison between these two costs, however, will not be straight forward, as we do not know the learning cost beforehand for a certain performance threshold of the resultant generator. Therefore, a performance bound of the generator along the sample size will need to be developed to guide the decision on whether the learning has a practical value or not.

7. Conclusion

We were motivated by the lack of knowledge accumulation capability of existing computational solvers for design problems. This drawback of machines hampers them from quickly creating good solutions in response to changing requirements in real-world design processes. Our solution to this end was to create a solution generator that adaptively learns from true solutions from a distribution of problems and predicts solutions to other unseen problems from the same distribution. The generator was modeled by a feedforward neural network, and thus produces solutions in one-shot, as opposed to through iterations as in conventional approaches. Our key contribution was the introduction of problem-specific optimality conditions as a tractable validation measure to enable more effective learning. We highlight that computing the violation of a generated solution to the optimality conditions requires only a single finite element analysis, while comparing the generated solution with the ground truth would require an entire topology optimization process, which requires thousands of finite element analysis for the problems we studied. We showed through two case studies that the proposed learning algorithm achieves significantly better generalization performance than the benchmarks under the same computational cost. While more sophisticated topology optimization settings should be tested, the proposed method is generally applicable to tasks of learning distributions of optimal solutions, provided that the optimality conditions can be derived and computed at low cost. Source codes for reproducing results from the paper are available [here](#).

Appendix

Topology optimization through augmented Lagrangian

The topology optimization problem of (TO) is solved using Alg. 2. Note that other algorithms, such as the method of moving asymptotes [48], are also applicable.

Algorithm 2: Topology optimization through Augmented Lagrangian (AL)

```
input : Problem parameters  $\alpha, \beta, p, r_e, R_e, \beta_t, \mathbf{s}$ 
output: Design variable  $\mathbf{x}$ 
1 Set algorithmic parameters  $\epsilon_{al} = 1, \epsilon_{opt} = 10^{-3}$ , initial guess  $\mathbf{x} = \alpha \mathbf{1}$ ;
2 Pre-compute neighbourhood  $\mathcal{M}_e$  and  $\mathcal{N}_e$ , and filter weights  $\omega_{i,j}$ ;
3 while  $\beta < \beta_t$  do
    /* gradually change the problem */
4     Set AL parameters:  $r_0 = 1, r_1 = 1, \mu_0 = \mu_1 = 0, \eta_0 = \eta_1 = 0.1, \epsilon = 1, \delta \mathbf{x} = 10^6 \mathbf{1}$ ;
5     Compute  $\tilde{\mathbf{x}}, \boldsymbol{\rho}, \bar{\boldsymbol{\rho}}, \mathbf{K}, \mathbf{u}, f, g_0$ , and  $g_1$ ;
6     while  $\max |\delta \mathbf{x}| > \epsilon_{al}$  or  $g_0 > 0$  or  $g_1 > 0$  do
7         /* solve the constrained problem */
8         Set  $\delta \mathbf{x} = 10^6 \mathbf{1}$ ;
9         while  $\max |\delta \mathbf{x}| > \epsilon$  do
10            /* solve the unconstrained problem */
11            Set learning rate  $a = 10^{-3}$ ;
12            Compute  $\nabla_{\mathbf{x}} f, \nabla_{\mathbf{x}} g_0, \nabla_{\mathbf{x}} g_1$ , and
13             $\delta \mathbf{x} = \nabla_{\mathbf{x}} f + (\mu_0 + 2g_0/r_0)\nabla_{\mathbf{x}} g_0 (g_0 > 0) + (\mu_1 + 2g_1/r_1)\nabla_{\mathbf{x}} g_1 (g_1 > 0)$ ;
14            while 1 do
15                /* line search */
16                Set  $\Delta \mathbf{x} = -a\delta \mathbf{x}$ , clip each element of  $\Delta \mathbf{x}$  to  $[-0.1, 0.1]$ ;
17                Set  $\mathbf{x}' = \mathbf{x} + \Delta \mathbf{x}$ ;
18                Compute  $\boldsymbol{\rho}', \bar{\boldsymbol{\rho}}', \mathbf{K}', \mathbf{u}', f', g'_0$ , and  $g'_1$  based on  $\mathbf{x}'$ ;
19                Compute  $L = f + \mu_0 g_0 + 0.5g_0^2/r_0 + \mu_1 g_1 + 0.5g_1^2/r_1$ ;
20                Compute  $L' = f' + \mu_0 g'_0 + 0.5g_0'^2/r_0 + \mu_1 g'_1 + 0.5g_1'^2/r_1$ ;
21                if  $L' - L > 0$  then
22                    /* if learning rate is too high */
23                    Set  $a = 0.5a$ ;
24                else
25                     $\mathbf{x} = \mathbf{x}'$ ;
26                end
27            end
28        end
29        /* update augmented Lagrangian parameters */
30        for  $i = 0, 1$  do
31            if  $g_i < \eta_i$  then
32                Set  $\mu_i = \mu_i + 2g_i/r_i, \eta_i = 0.5\eta_i$ ;
33            else
34                Set  $r_i = 0.5r_i$ ;
35            end
36        end
37    end
38    Set  $\beta = 2\beta$ ;
39 end
```

References

- [1] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, Accelerating Eulerian Fluid Simulation With Convolutional Networks, ArXiv e-prints [arXiv:1607.03597](https://arxiv.org/abs/1607.03597).
- [2] M. Chu, N. Thuerey, Data-driven synthesis of smoke flows with cnn-based feature descriptors, *ACM Transactions on Graphics (TOG)* 36 (4) (2017) 69.
- [3] R. Giryes, Y. C. Eldar, A. Bronstein, G. Sapiro, Tradeoffs between convergence speed and reconstruction accuracy in inverse problems, *IEEE Transactions on Signal Processing*.
- [4] I. Sosnovik, I. Oseledets, Neural networks for topology optimization, arXiv preprint [arXiv:1709.09578](https://arxiv.org/abs/1709.09578).
- [5] E. Ulu, R. Zhang, L. B. Kara, A data-driven investigation and estimation of optimal topologies under variable loading configurations, *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 4 (2) (2016) 61–72.
- [6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint [arXiv:1312.6199](https://arxiv.org/abs/1312.6199).
- [7] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, arXiv preprint [arXiv:1607.02533](https://arxiv.org/abs/1607.02533).
- [8] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, P. Abbeel, Adversarial attacks on neural network policies, arXiv preprint [arXiv:1702.02284](https://arxiv.org/abs/1702.02284).
- [9] J. Wu, N. Aage, R. Westermann, O. Sigmund, Infill optimization for additive manufacturing approaching bone-like porous structures, *IEEE transactions on visualization and computer graphics* 24 (2) (2018) 1127–1140.
- [10] M. P. Bendsøe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, *Computer methods in applied mechanics and engineering* 71 (2) (1988) 197–224.
- [11] T. Guo, D. J. Lohan, R. Cang, M. Y. Ren, J. 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, 2018, p. 0804.
- [12] R. Cang, Y. Xu, S. Chen, Y. Liu, Y. Jiao, M. Y. Ren, Microstructure representation and reconstruction of heterogeneous materials via deep belief network for computational material design, *Journal of Mechanical Design* 139 (7) (2017) 071404.
- [13] R. Cang, H. Li, H. Yao, Y. Jiao, Y. Ren, Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model, *Computational Materials Science* 150 (2018) 212 – 221.
- [14] R. Bostanabad, Y. Zhang, X. Li, T. Kearney, L. C. Brinson, D. W. Apley, W. K. Liu, W. Chen, Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques, *Progress in Materials Science* 95 (2018) 1 – 41.
- [15] G. Stiny, Introduction to shape and shape grammars, *Environment and planning B: planning and design* 7 (3) (1980) 343–351.
- [16] S.-W. Hsiao, C.-H. Chen, A semantic and shape grammar based approach for product design, *Design studies* 18 (3) (1997) 275–296.
- [17] G. Allaire, F. Jouve, A.-M. Toader, Structural optimization using sensitivity analysis and a level-set method, *Journal of computational physics* 194 (1) (2004) 363–393.
- [18] D. Jung, H. C. Gea, Topology optimization of nonlinear structures, *Finite Elements in Analysis and Design* 40 (11) (2004) 1417–1427.
- [19] O. Sigmund, Design of multiphysics actuators using topology optimization—part ii: Two-material structures, *Computer methods in applied mechanics and engineering* 190 (49-50) (2001) 6605–6627.
- [20] M. Y. Wang, X. Wang, “color” level sets: a multi-phase method for structural topology optimization with multiple materials, *Computer Methods in Applied Mechanics and Engineering* 193 (6-8) (2004) 469–496.
- [21] B. Zhu, M. Skouras, D. Chen, W. Matusik, Two-scale topology optimization with microstructures, *ACM Transactions on Graphics (TOG)* 36 (5) (2017) 164.
- [22] Y. Ren, P. Y. Papalambros, A design preference elicitation query as an optimization process, *Journal of Mechanical Design* 133 (11) (2011) 111004.

- [23] G. Orbay, L. Fu, L. B. Kara, Deciphering the influence of product shape on consumer judgments through geometric abstraction, *Journal of Mechanical Design* 137 (8) (2015) 081103.
- [24] Y. M. Xie, G. P. Steven, A simple evolutionary procedure for structural optimization, *Computers & structures* 49 (5) (1993) 885–896.
- [25] O. Querin, G. Steven, Y. Xie, Evolutionary structural optimisation (eso) using a bidirectional algorithm, *Engineering computations* 15 (8) (1998) 1031–1048.
- [26] R. V. Rao, V. J. Sivasani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, *Computer-Aided Design* 43 (3) (2011) 303–315.
- [27] G. Kharmanda, N. Olhoff, A. Mohamed, M. Lemaire, Reliability-based topology optimization, *Structural and Multidisciplinary Optimization* 26 (5) (2004) 295–307.
- [28] R. Vilalta, Y. Drissi, A perspective view and survey of meta-learning, *Artificial Intelligence Review* 18 (2) (2002) 77–95.
- [29] S. Hochreiter, A. S. Younger, P. R. Conwell, Learning to learn using gradient descent, in: *International Conference on Artificial Neural Networks*, Springer, 2001, pp. 87–94.
- [30] R. Caruana, Learning many related tasks at the same time with backpropagation, in: *Advances in neural information processing systems*, 1995, pp. 657–664.
- [31] S. J. Pan, Q. Yang, A survey on transfer learning, *IEEE Transactions on knowledge and data engineering* 22 (10) (2010) 1345–1359.
- [32] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, Domain-adversarial training of neural networks, *The Journal of Machine Learning Research* 17 (1) (2016) 2096–2030.
- [33] S. Thrun, Is learning the n-th thing any easier than learning the first?, in: *Advances in neural information processing systems*, 1996, pp. 640–646.
- [34] S. Ravi, H. Larochelle, Optimization as a model for few-shot learning.
- [35] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, N. de Freitas, Learning to learn by gradient descent by gradient descent, in: *Advances in Neural Information Processing Systems*, 2016, pp. 3981–3989.
- [36] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, N. de Freitas, Learning to learn without gradient descent by gradient descent, arXiv preprint arXiv:1611.03824.
- [37] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, M. Botvinick, Learning to reinforcement learn, arXiv preprint arXiv:1611.05763.
- [38] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, *Journal of machine learning research* 2 (Nov) (2001) 45–66.
- [39] B. Settles, Active learning, *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6 (1) (2012) 1–114.
- [40] D. D. Lewis, J. Catlett, Heterogeneous uncertainty sampling for supervised learning, in: *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 148–156.
- [41] N. Roy, A. McCallum, Toward optimal active learning through monte carlo estimation of error reduction, *ICML*, Williamstown (2001) 441–448.
- [42] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, W. T. Freeman, Physics 101: Learning physical object properties from unlabeled videos., in: *BMVC*, Vol. 2, 2016, p. 7.
- [43] R. Jonschkowski, R. Hafner, J. Scholz, M. Riedmiller, Pves: Position-velocity encoders for unsupervised learning of structured state representations, arXiv preprint arXiv:1705.09805.
- [44] R. Stewart, S. Ermon, Label-free supervision of neural networks with physics and domain knowledge., in: *AAAI*, 2017, pp. 2576–2582.
- [45] M. Denil, P. Agrawal, T. D. Kulkarni, T. Erez, P. Battaglia, N. de Freitas, Learning to perform physics experiments via deep reinforcement learning, arXiv preprint arXiv:1611.01843.
- [46] H. Narayanan, S. Mitter, Sample complexity of testing the manifold hypothesis, in: *Advances in Neural Information Processing Systems*, 2010, pp. 1786–1794.
- [47] N. Lei, Z. Luo, S.-T. Yau, D. X. Gu, Geometric understanding of deep learning, arXiv preprint arXiv:1805.10451.

- [48] K. Svanberg, The method of moving asymptotes a new method for structural optimization, *International journal for numerical methods in engineering* 24 (2) (1987) 359–373.