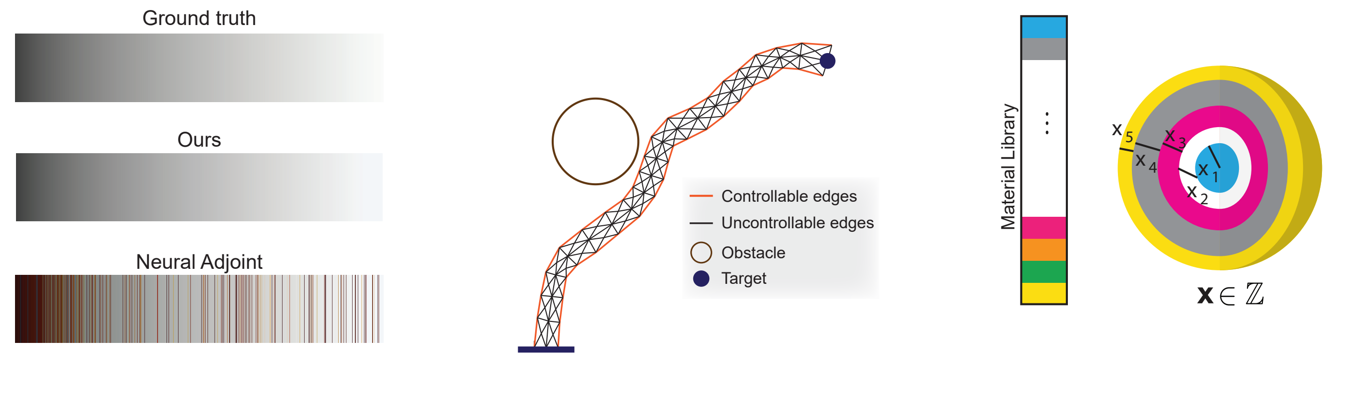
Mixed Integer Neural Inverse Design

NAVID ANSARI, Max Planck Institute for Informatics, Germany
HANS-PETER SEIDEL, Max Planck Institute for Informatics, Germany
VAHID BABAEI, Max Planck Institute for Informatics, Germany



(a) Spectral printing (b) Soft robot inverse kinematics (c) Nano-photonics integer inversion

Fig. 1. We address challenging problems in neural inverse design by taking advantage of the underlying mathematical properties of neural surrogate models (NSMs). For example, in Figure 1a, given a piecewise linear NSM that predicts the spectrum of a color as a function of the input ink ratios, we can find the best combination of ink ratios for reproducing a certain target spectrum. In Figure 1b, we find the optimal control parameters of a soft robot such that it reaches a target location while avoiding an obstacle. Figure 1c depicts an integer-constrained inverse design problem where for a multi-shell nano-spherical scatterer we find the optimal *integer* thicknesses of base materials (potentially from within a large material library) to obtain a desired scattering profile.

In computational design and fabrication, neural networks are becoming important surrogates for bulky forward simulations. A long-standing, intertwined question is that of inverse design: how to compute a design that satisfies a desired target performance? Here, we show that the piecewise linear property, very common in everyday neural networks, allows for an inverse design formulation based on mixed-integer linear programming. Our mixed-integer inverse design uncovers globally optimal or near optimal solutions in a principled manner. Furthermore, our method significantly facilitates emerging, but challenging, combinatorial inverse design tasks, such as material selection. For problems where finding the optimal solution is intractable, we develop an efficient yet near-optimal hybrid approach. Eventually, our method is able to find solutions provably robust to possible fabrication perturbations among multiple designs with similar performances. Our code and data are available at https://gitlab.mpi-klsb.mpg.de/nansari/mixed-integer-neural-inverse-design.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Artificial intelligence**.

Authors' addresses: Navid Ansari, Max Planck Institute for Informatics, Saarbrücken, Germany, nansari@mpi-inf.mpg.de; Hans-Peter Seidel, Max Planck Institute for Informatics, Saarbrücken, Germany, hpseidel@mpi-sb.mpg.de; Vahid Babaei, Max Planck Institute for Informatics, Saarbrücken, Germany, vbabaei@mpi-inf.mpg.de.

0730-0301/2022/7-ART151
https://doi.org/10.1145/3528223.3530083

Additional Key Words and Phrases: Computational design, fabrication, inverse design, mixed-integer programming, neural networks, surrogate models

ACM Reference Format:
Navid Ansari, Hans-Peter Seidel, and Vahid Babaei. 2022. Mixed Integer Neural Inverse Design. *ACM Trans. Graph.* 41, 4, Article 151 (July 2022), 14 pages. https://doi.org/10.1145/3528223.3530083

1 INTRODUCTION

Data-driven prediction of a design's *performance* is an indispensable tool in computational design and fabrication. The amazing success of deep neural networks in computer vision and natural language processing is propelling the development of neural-network based surrogate models, or neural surrogate models (NSMs), in computational design [Jiang et al. 2020]. NSMs either learn and replace computationally expensive physics-based simulations [Kiarashinejad et al. 2020] or are fitted to measured data when accurate simulations are not available [Shi et al. 2018]. In addition to accelerating the computational design pipeline, generality is an implicit but important advantage of learned surrogate models: the same developed machinery can be applied to NSMs learned independently for different applications.

Forward predictions are essential for troubleshooting and analysis in computational design. But, oftentimes, their most important application is in *inverse design*, i.e., the reverse process of mapping functional goals, or performances, into fabricable designs. Although there has been recent progress in inverting neural networks [Ren et al. 2020], there remain many unaddressed challenges. Due to the non-convexity of NSMs [Bunel et al. 2018], none of the current neural network inversion methods is capable of reasoning about the optimality of the obtained solutions. Moreover, for many naturally occurring combinatorial problems in computational design, such as selecting an optimal subset of materials, we still have to resort to stochastic algorithms.

Our main insight in this work is that given a *piecewise linear* neural surrogate model (PL-NSM), the inverse design problem can be formulated as a mixed-integer linear program (MILP). The piecewise linearity assumption is not particularly restrictive: most common neural networks are a composition of linear transformations, such as fully-connected or convolution layers, and piecewise linear activation functions, such as the rectified linear unit (ReLU). A MILP formulation of NSM-based inverse design addresses the challenges mentioned above. The MILP can be solved with measurable optimality as it produces the *gap* between the objective's *relaxed* and *feasible* solutions[1]. For small and medium sized networks, our inverse design objective typically reaches a gap of 0, i.e., finds the globally optimal design. For larger networks, due to the combinatorial complexity of solving MILPs, finding global optima becomes increasingly difficult. Nevertheless, for these networks the solutions are still *near optimal* as the relaxed solution can be computed via linear relaxation of the corresponding MILP, i.e., a linear program. We also show that the objective's feasible solution can be computed using alternative inverse methods thereby accelerating the gap closure via a hybrid of gradient-based and MILP approaches. Furthermore, the MILP can be straightforwardly augmented to solve challenging combinatorial inverse design problems. This is a significant advantage, as a large portion of inverse design problems are combinatorial by nature due to different fabrication requirements. Finally, when the optimization objective for many designs is similar, our method can be used to sort those solutions based on their robustness to different perturbations. The main contributions of this paper are:

- A novel method of inverse design via casting the inversion of piecewise linear NSMs as mixed-integer linear programming.
- Introducing a hybrid approach capable of providing near optimality certificate while inverting large neural networks.
- Proposing a framework to reliably analyze the robustness of the inverse designs.
- Equipping the MILP inverse design with combinatorial constraints and applying it on a range of real-world problems.

We evaluate our proposed approaches through an extensive set of experiments in spectral printing, soft robot inverse kinematics, and photonic design.

2 BACKGROUND AND RELATED WORK

Functional Fabrication. One of the most important missions of computational design and fabrication is to translate functional goals, or performances, into fabricable designs [Bermano et al. 2017]. In the computational fabrication literature, there are many examples trying to find a design for a prescribed performance. Example performances include deformation [Schumacher et al. 2015], color [Sumin et al. 2019], gloss [Matusik et al. 2009], shadow [Mitra and Pauly 2009], relief [Schüller et al. 2014], caustics [Schwartzburg et al. 2014], etc. In order to solve these challenging inverse problems, often, the fabrication process is first modeled in a forward fashion where the performance is predicted from its corresponding design. Then, to solve the original *performance to design* problem, the forward process is inverted using an optimization. Inspired by the similarity among these problems, Chen et al. [2013] propose a framework, called *spec2fab*, that abstracts the functional fabrication process in a general manner. In this work, we focus on functional fabrication problems whose forward modeling can be expressed via a piecewise linear neural network.

Neural Networks and Computational Design. Neural networks can map designs to performances by approximating complex physics simulations [Kiarashinejad et al. 2020]. Moreover, they can operate as purely data-driven simulations when accurate physics-based models are unavailable or difficult to develop [Shi et al. 2018]. In addition to accelerating the computations, neural surrogate models are highly transferable across different applications due to their underlying similarities. Perhaps *photonic design* is the front-runner field in using neural surrogate models for computational design [Jiang et al. 2020]. Also, in computational fabrication we are seeing a surge in the use of NSMs, such as in computational design of cold-bent glass façades [Gavriil et al. 2020], appearance-preserving tactile design [Tymms et al. 2020], or fine art reproduction [Shi et al. 2018]. A particularly relevant related work is the recent ink selection method [Ansari et al. 2020]. In order to benefit from efficient MILP solvers, for the ink selection problem, this work develops a linear, but approximate, forward model that predicts the spectra of different ink combinations. (In general, developing such linear spaces requires deep domain knowledge and specialized measurements.) In a second stage, for spectral reproduction of a given painting, an accurate data-driven forward model based on NSMs is deployed. Here we show that *both* ink selection and spectral reproduction can be performed using a single neural-network forward model without requiring additional, specialized models.

Neural Network Inversion. Recently, there has been a surge in inverse models for neural networks. The first solution coming to mind is to train neural networks in the reverse direction using the training data. This naive approach fails because of the one-to-many nature of the problem: the same performance could lead to different designs causing problems during optimization (e.g., via inconsistent gradients). In order to bypass this challenge, *tandem* networks [Liu et al. 2018; Shi et al. 2018] map performances into designs using a first neural network but, in order to compute a consistent loss, use a pre-trained forward network to map the resulting design into its corresponding performance. Conditional variational auto-encoders [Kingma and Welling 2013] have also been used for the inverse design task [Kiarashinejad et al. 2020]. These networks condition the design on the target performance and yield a distribution of solutions from which multiple samples could be drawn.

[1] *Relaxed solutions* are obtained by partially dropping the integer constraints of the original problem. As a result, in case of a minimization (maximization) the relaxed solution is an estimate guaranteed to be smaller (larger) than the optimal feasible solution. The MILP solver tries in parallel to find better *feasible solutions* and to improve the relaxed solution by progressively dropping fewer integer constraints. When finally all constraints are considered, the relaxed and feasible solutions are equal, $gap = 0$, indicating the optimal solution is found.

An invertible neural network [Ardizzone et al. 2019], based on *real NVP* [Dinh et al. 2017], is another inversion method. In this method, a specialized architecture based on normalizing flows is trained in both forward and inverse directions leading to a bijective mapping between design and performance spaces. Ren et al. [2020] benchmarked these inverse methods and found out that a gradient-based method, via backpropagation with respect to the design variables, results in significantly more accurate solutions. Dubbed as *neural adjoint* (NA), this method uses a boundary loss to punish infeasible designs. It also runs the optimization starting from multiple random initial guesses in search for the best objective value. Our method, by nature, is similar to the NA as both are *optimizations* and not inverse architectures as in [Liu et al. 2018; Shi et al. 2018; Ardizzone et al. 2019]. In Section 4, we evaluate our method against the NA extensively.

Mixed-Integer Programming and Neural Networks. The mathematical optimization problems in which all or some variables are integers are known as mixed-integer programming[2] (MIP) [Floudas 1995]. A technique for solving MIPs with nonlinear, nonconvex functions, dating back to Markowitz and Manne [1957], is to estimate those functions with piecewise linear functions [Belotti et al. 2013]. The resulting approximation, usually via auxiliary binary variables, is a mixed-integer linear programming with more scalable and efficient solvers. The connection between piecewise linearity of some class of neural networks and MILP solvers has only recently been identified [Cheng et al. 2017]. MILP formulation has since been exploited for formal verification of networks against adversarial attacks [Fischetti and Jo 2018]. It is important to note that unlike the classic use of piecewise linear functions for *approximating* non-linear functions, MILP representation is simply a *reformulation* of the already piecewise linear networks without any approximation. We borrow the MILP formulation of piecewise-linear networks, initially appeared in the formal verification literature [Fischetti and Jo 2018; Bunel et al. 2018; Tjeng et al. 2019], and develop a novel neural inverse design framework. To the best of our knowledge, we are the first to introduce the MILP-based *neural inverse design* and extend it to related tasks, such as combinatorial inverse design problems.

3 NEURAL INVERSE DESIGN VIA MIXED-INTEGER LINEAR PROGRAMMING

In this section, we take a forward model expressed as a trained, piecewise linear neural network and invert it using mixed-integer linear programming. In addition to solving typical inverse design problems, we show how extra integer constraints can readily be added to our pipeline allowing for solving challenging combinatorial inverse design problems. Additionally, this formulation can be easily adapted for evaluating the robustness of different designs. Finally, our proposed method can be combined with other inversion methods in order to find more accurate near-optimal designs efficiently.

3.1 Mixed-Integer Formulation

A *feedforward* neural network F_θ is built by a number of function compositions [Montufar et al. 2014]

$$\mathbf{x}^L = F_\theta(\mathbf{x}^0) = f^L \circ g^{L-1} \circ f^{L-1} \circ \ldots \circ g^1 \circ f^1(\mathbf{x}^0) \tag{1}$$

and maps the input $\mathbf{x}^0 \in \mathbb{R}^m$ to the output $\mathbf{x}^L \in \mathbb{R}^n$ (note that the last layer does not undergo an activation). Here, f^l is a linear pre-activation function

$$\begin{gathered} f^l(\mathbf{x}^{l-1}) = \mathbf{W}^l \mathbf{x}^{l-1} + \mathbf{b}^l \\ \forall\, l \in \{1, 2, \cdots, L\} \end{gathered} \tag{2}$$

whose weights $\mathbf{W}^l$ and biases $\mathbf{b}^l$ at all layers (1 to L) make up the network's parameters θ which are computed during the training. In our notation superscripts and subscripts (to appear later) indicate the layers and nodes, respectively. The function g^l is a nonlinear activation function. Throughout all inverse problems in this work we assume the widely used rectified linear unit or ReLU as the activation function. Using other piecewise linear activation functions, such as leaky ReLU or max pooling layers is also straightforward. The ReLU function is defined as

$$\mathbf{x}^l = g^l(f^l(\mathbf{x}^{l-1})) = \max\{\mathbf{0}, \mathbf{W}^l \mathbf{x}^{l-1} + \mathbf{b}^l\}. \tag{3}$$

We adopt a vector-matrix notation for compactness and readability. That is, the max operator in Equation 3 takes a vector input and outputs the component-by-component maxima.

In a general neural inverse problem we search for an input vector $\mathbf{x}^0$ that minimizes a distance, using $\mathcal{L}_1$ norm[3], between the network prediction and a target performance $\mathbf{t}$

$$\underset{\mathbf{x}^0}{\operatorname{argmin}} \left\| F_\theta(\mathbf{x}^0) - \mathbf{t} \right\|_1. \tag{4}$$

This optimization is very challenging as F_θ is a highly non-linear, non-convex function [Bunel et al. 2018]. Nevertheless, we can exploit the piecewise linear structure of neural networks and model their optimization using mixed-integer linear programming. That is, the optimization only involves linear terms and constraints. In doing so, we eliminate the network's non-linearities at the cost of introducing new binary and continuous variables.

We adapt the MILP-based reformulation of ReLU networks [Tjeng et al. 2019] (previously used for formal verification) for solving our central inversion problem, summarized in Equation 4. Given a pretrained network F_θ with a given set of weights $\mathbf{W}^l$ and biases $\mathbf{b}^l$, we encode the inverse problem shown in Equation 4 as a MILP with linear and binary constraints[4]

$$\underset{\mathbf{z}^1, \cdots, \mathbf{z}^{L-1}, \quad \mathbf{x}^0, \cdots, \mathbf{x}^L}{\operatorname{argmin}} \left\| \mathbf{x}^L - \mathbf{t} \right\|_1 \tag{5a}$$

$$\mathbf{x}^l \preceq \mathbf{W}^l \mathbf{x}^{l-1} + \mathbf{b}^l \; - \; \mathbf{l}^l (\mathbf{1} - \mathbf{z}^l) \tag{5b}$$

$$\mathbf{x}^l \succeq \mathbf{W}^l \mathbf{x}^{l-1} + \mathbf{b}^l \tag{5c}$$

$$\mathbf{x}^l \preceq \mathbf{u}^l \odot \mathbf{z}^l \tag{5d}$$

$$\mathbf{x}^l \succeq \mathbf{0} \tag{5e}$$

$$\mathbf{z}^l \in \{0, 1\}^{K^l}. \tag{5f}$$

[2]We recommend the following short and gentle introduction to MIP and its solvers to the less familiar reader: https://www.gurobi.com/resource/mip-basics/

[3]The $\mathcal{L}_1$ norm is more amenable to linearization.

[4]Note the curled inequalities and $\odot$ symbol indicate our continuing use of component-by-component convention.

For the nodes at layer l we introduce a set of continuous ($\mathbf{x}^l$) and binary ($\mathbf{z}^l$) variables. Vectors $\mathbf{l}^l$ and $\mathbf{u}^l$ are the lower and upper bounds to the nodes' pre-activation values $\mathbf{W}^l\mathbf{x}^{l-1} + \mathbf{b}^l$ and are precomputed (see Section 3.3).

While optimizing Equation 5, the solver *branches* on these binary variables and, in the worst case, checks all possible network configurations. It is simple to verify that the constraints replace the role of $\max\{0, .\}$ operation. When $\mathbf{z}^l_k = 1$ (corresponding to neuron k in layer l) the constraints 5b and 5c are binding and thus:

$$\begin{cases} \mathbf{x}^l_k \leq \mathbf{W}^l_{(k,:)}\mathbf{x}^{l-1} + \mathbf{b}^l_k \\ \mathbf{x}^l_k \geq \mathbf{W}^l_{(k,:)}\mathbf{x}^{l-1} + \mathbf{b}^l_k \end{cases} \implies \mathbf{x}^l_k = \mathbf{W}^l_{(k,:)}\mathbf{x}^{l-1} + \mathbf{b}^l_k$$

where $\mathbf{W}^l_{(k,:)}$ is k'th row of matrix $\mathbf{W}$ and represents all the weights which are connecting layer $\mathbf{x}^{l-1}$ to $\mathbf{x}^l_k$. Otherwise, when $\mathbf{z}^l_k = 0$, the constraints in equations 5d and 5e are binding and thus:

$$\begin{cases} \mathbf{x}^l_k \leq 0 \\ \mathbf{x}^l_k \geq 0 \end{cases} \implies \mathbf{x}^l_k = 0$$

Note that while we are mostly interested in the optimized value of $\mathbf{x}^0$, we should optimize all introduced binary and continuous variables to enforce the constraints in Equation 5 and thus have a correct representation of the neural network.

3.2 Combinatorial Inverse Design

The MILP representation of the inverse design problem can take additional integer constraints in a seamless manner. These integer constraints appear in many computational design problems. For example, in a *selection* problem, we are interested in a limited number D of all input design features $\mathbf{x}^0$, which results in an optimal target performance $\mathbf{t}$. We can cast the selection problem as Equations 5a to 5f **in addition** to

$$\sum_{i=1}^{K^0} \mathbf{q}_i \leq D \tag{6a}$$

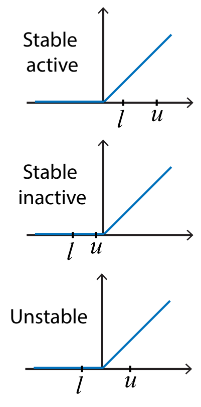
$$\mathbf{q} \in \{0,1\}^{K^0} \tag{6b}$$

$$0 \leq \mathbf{x}^0_i \leq \mathbf{q}_i, \quad \forall\, i \in \left\{1, 2, \cdots, K^0\right\} \tag{6c}$$

where the vector of inputs to the neural network $\mathbf{x}^0$ is of size K^0 and normalized between 0 and 1, and $\mathbf{q}$ is our introduced *selection variable*, a binary vector of same size (different from previously defined binary variables $\mathbf{z}^l$). The inequality constraints 6a and 6c ensure that at most D entries of $\mathbf{x}^0$ take non-zero values and thus used for estimating $\mathbf{t}$. Indices of these entries match the indices of non-zero elements in $\mathbf{q}$ and point to the selected elements. Other combinatorial inverse design problems can be formulated similarly by adding proper constraints and integer variables. In Sections 4.3 and 4.4 we show how this formulation is applied to real-world inverse problems.

3.3 Bound Precomputation

We precompute as tight as possible lower $\mathbf{l}^l$ and upper $\mathbf{u}^l$ bounds to the pre-activation values $\mathbf{W}^l\mathbf{x}^{l-1} + \mathbf{b}^l$. There are two main, interrelated advantages in bound tightening. First, it improves the solve time of the problem by strengthening its formulation [Vielma 2015]. Second, tighter bounds can lead to more *stable* ReLUs. Stable ReLUs are those that operate on nodes whose bounds lie completely within either positive or negative domain (see the inset). When the bounds lie within the positive domain (stably active), the value of such a node is always a linear combination of preceding nodes and there is no need to introduce new optimization variables for it. When the bounds lie within the negative domain (stably inactive), the value of such a node is always zero and therefore the corresponding variables are dropped. Otherwise, we have *unstable* ReLUs for which we must define binary and continuous variables.



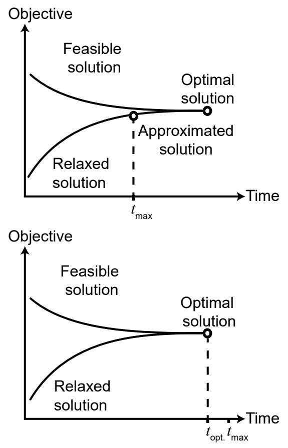
The procedure for bound precomputation is similar to our original inverse problem where we calculate the minimum (maximum) of each node of the neural network using the same mixed-integer formulation (Equation 5). Except that instead of minimizing (maximizing) the distance to the target $\mathbf{t}$, we minimize (maximize) the value of each individual node k in layer l:

$$\underset{\mathbf{z}^1, \cdots, \mathbf{z}^{l-1}, \mathbf{z}^l_k, \quad \mathbf{x}^0, \cdots, \mathbf{x}^{l-1}, \mathbf{x}^l_k}{\operatorname{argmin}} \quad \mathbf{x}^l_k \tag{7}$$

$$\Big(\underset{\mathbf{z}^1, \cdots, \mathbf{z}^{l-1}, \mathbf{z}^l_k, \quad \mathbf{x}^0, \cdots, \mathbf{x}^{l-1}, \mathbf{x}^l_k}{\operatorname{argmax}} \quad \mathbf{x}^l_k \Big). \tag{8}$$

This optimization is still subjected to constraints 5b to 5f for the considered *node* and all preceding layers.

Our bound tightening algorithm is based on [Fischetti and Jo 2018] but extended to include the design constraints. Since designs ($\mathbf{x^0}$ in our notation) usually come with their own constraints, we observe that it is highly beneficial to enforce those constraints when precomputing the nodes' bounds as they lead to tighter bounds. The bound precomputation can be very expensive for larger networks as it should be performed on each node separately. The computation is especially heavy within last layers of the network. In practice, we set a time limit ($t_{\max}$) for the solver during this computation. If we stop the optimization prematurely, the relaxed solution of the optimization is the node's bound. As depicted in the inset figures, relaxed solutions are conservative estimation of the optimal solutions and guarantee that in case of a minimization (maximization) there are no smaller (larger) solutions than this estimation. The feasible solution, however, is the solution that is found for the original MILP problem thus far and there could



be smaller (larger) values if the minimization (maximization) continues. It is important to note that using feasible solution as upper and lower bounds in an early-stopped optimization, results in overestimating the lower bound and underestimating the upper bound values. This will lead to calculating incorrectly tighter bounds and overestimating the number of stable ReLUs, which results in suboptimal solutions. Algorithm 1 in Appendix A presents the extended bound precomputation step by step.

3.4 Combination of Gradient-Based Optimizations and MILP

For non-combinatorial inverse design problems, gradient-based optimizations are an attractive choice given that the network's gradient information can be efficiently computed via automatic differentiation. The neural adjoint (NA) method [Ren et al. 2020], for example, relies on a gradient descent approach based on the backpropagation algorithm [Hecht-Nielsen 1992] for inverting neural networks. The process is very similar to network training except instead of network's parameters its input is optimized. Despite its good scalability with network's size, this (and any other) neural inverse method is unable to reason about the optimality of the obtained solutions.

Using our method, once the neural inverse design is formulated via MILP, we immediately obtain a relaxed solution to the objective. Typical MILP solvers search recursively for both feasible and relaxed solutions of the objective and try to close their gap as quickly as possible [Klotz and Newman 2013]. Therefore, any feasible solution is a near optimal solution because we know how far it is from the (conservative) relaxed solution at any moment. Given the obtained solutions via NA are all feasible to the MILP objective, we can solve for the relaxed solution of the objective via MILP and its feasible solution via NA. In practice we run NA and the MILP on the same network simultaneously and track the optimality gap. We can then stop the process by monitoring the gap. Note that, for larger neural networks (in our experience approximately larger than 4 layers each 150 neuron wide) it is not tractable to insist on a zero gap. As we show in Section 4.5, NA solutions (in case of minimization) reduce the objective's feasible solutions significantly more quickly resulting in a tighter objective's bound in a less amount of time.

3.5 Design Robustness

Inverse designs are typically one-to-many problems where for a given performance there are multiple acceptable designs. It is therefore interesting to study other attributes during inverse design. An important attribute is the robustness of designs to possible perturbations during fabrication [Sigmund 2009]. We define the robustness of a computed, candidate design $\hat{\mathbf{x}}^0$ as the maximum deviation of its performance from a desired target performance $\mathbf{t}$ when the candidate design is perturbed by a small positive number ϵ at each dimension. In other words, we look for the worst performance of a design when it is allowed to roam inside a hypercube around it. The mixed integer formulation allows us to find the *provably* worst performance. We write this problem as

$$\begin{gathered}\underset{\mathbf{z}^1,\cdots,\mathbf{z}^{L-1},\quad \mathbf{x}^0,\cdots,\mathbf{x}^L}{\operatorname{argmax}} \left\|\mathbf{x}^L-\mathbf{t}\right\|_1 \\ \hat{\mathbf{x}}^0_i-\epsilon \leqslant \mathbf{x}^0_i \leqslant \hat{\mathbf{x}}^0_i+\epsilon, \quad \forall\, i \in \left\{1,2,\cdots,K^0\right\}.\end{gathered} \tag{9}$$

Once again this optimization is subjected to constraints 5b to 5f. Note that the candidate design $\hat{\mathbf{x}}^0$ need not necessarily come from MILP-based inversion. In our case, in Section 4.6, we use the neural adjoint (NA) method for computing candidate designs. In general, robustness computation is more efficient than typical MILP-based inversion as the design is usually perturbed within a small neighborhood. This, on top of bound precomputation, leads to further reduction of unstable ReLUs. As a result, when dealing with robustness computation, the scalability becomes problematic only for significantly larger networks.

4 EVALUATION

In this section, we demonstrate the potential of our proposed method. For our analyses and experiments, we focus on several real-life applications in three different areas of computational design and control: neural spectral printing [Shi et al. 2018; Ansari et al. 2020], inverse kinematic of soft robots [Xue et al. 2020; Sun et al. 2021], and photonic design [Peurifoy et al. 2018; Nadell et al. 2019]. We solve all MILPs using Gurobi, a state-of-the-art solver [Gurobi Optimization 2018]. In order to better relate the experiments to the theory (developed in Section 3), in Table 1, we summarize the setup of each experiment in connection with Equations 5 and 6. All bound precomputation and MILPs, unless otherwise mentioned, are solved on a CPU cluster with 256 cores. This does not mean that employing all cores is always desirable when solving a MILP. In practice, we find that using more than 30 cores does not help with the speed up. On the other hand, the nodes' bound precomputation is trivially parallelizable for the nodes belonging to the same layer. The time limit $t_{\max}$ for bound precomputation is set to 150 seconds. We collect the timing of different computations, and the configuration of each NSM in Table 2. In all experiments, the reported error is the objective value of our optimization (based on the $\mathcal{L}_1$ norm) for a set of unseen, target performances.

4.1 Neural Spectral Separation

We begin by studying a neural inverse problem in spectral printing. Spectral printing ensures that printed items are visually close to the originals, independent of the color of the light source under which they are observed. In this experiment we used two different neural networks as the input to our method. The 4-ink network is a trained PL-NSM with 4 hidden layers each having 100 neurons and ReLU activation functions [Ansari et al. 2020]. The 44-ink network is a trained PL-NSM with 2 hidden layers each having 50 neurons and ReLU activation functions. The 4-ink network, a surrogate for a forward spectral prediction model, has a 4D design space made of CMYK (cyan, magenta, yellow and black) ink ratios and outputs a 31D spectrum. Here the inverse design problem, known as *spectral separation*, concerns finding the ink ratios for a target spectrum. A particularly challenging target for spectral separation is the perfect gray ramp introduced by Ansari et al. [2020]. This gradient is formed by 901 dark to light ideal gray spectra which have equal reflectivity across all visible wavelengths (Figure 2).

We compare our MILP-based inversion with both the method of tandem, previously used for the exactly same problem [Shi et al. 2018; Ansari et al. 2020], and the neural adjoint (NA) [Ren et al.

Table 1. The setup of each experiment in connection with Equations 5 and 6.

		Variables of Equations 5 and/or 6				
Experiment name	Experiment mode	$\mathbf{t}$ (Target)	$\mathbf{x}^0$ (Input)	$\mathbf{x}^L$ (Output)	Design constraints	Variable of interest
Neural Spectral Separation	4-ink network	Perfect gray spectrum ($\mathbf{t} \in \mathbb{R}^{31}$)	Ink ratios ($\mathbf{x}^0 \in \mathbb{R}^4$)	Gray spectrum ($\mathbf{x}^L \in \mathbb{R}^{31}$)	$0 \leq \mathbf{x}^0 \leq 1$	$\mathbf{x}^0$
	44-ink network	Perfect gray spectrum ($\mathbf{t} \in \mathbb{R}^{31}$)	Ink ratios ($\mathbf{x}^0 \in \mathbb{R}^{44}$)	Gray spectrum ($\mathbf{x}^L \in \mathbb{R}^{31}$)	$0 \leq \mathbf{x}^0 \leq 1$	$\mathbf{x}^0$
Soft Robot Inverse Kinematics	-	Target location ($\mathbf{t} \in \mathbb{R}^2$)	Stretch and contraction ($\mathbf{x}^0 \in \mathbb{R}^{40}$)	Location of all vertices ($\mathbf{x}^L \in \mathbb{R}^{206}$)	Equations 11, 12, and 13	$\mathbf{x}^{\mathbf{0}}$
Material Selection	Selection	Matrix of 6 spectra ($\mathbf{t} \in \mathbb{R}^{6\times31}$)	Matrix of 6 area coverage ($\mathbf{x}^0 \in \mathbb{R}^{6\times44}$)	Matrix of 6 spectra ($\mathbf{x}^L \in \mathbb{R}^{6\times31}$)	Equation 6, $D = 2$, $\mathbf{q} \in \{0, 1\}^{44}$	$\mathbf{q}$
	Inversion	Painting's color spectrum ($\mathbf{t} \in \mathbb{R}^{31}$)	Ink ratios ($\mathbf{x}^0 \in \mathbb{R}^{44}$)	Color spectrum ($\mathbf{x}^L \in \mathbb{R}^{31}$)	$0 \leq \mathbf{x}^0 \leq \mathbf{q}$	$\mathbf{x}^0$
Nano-Photonics	Inversion (rounded)	Scattering cross section ($\mathbf{t} \in \mathbb{R}^{200}$)	Spherical shell thickness ($\mathbf{x}^0 \in \mathbb{R}^4$)	Scattering cross section ($\mathbf{x}^L \in \mathbb{R}^{200}$)	$0 \leq 10\mathbf{x}^0 \leq 70$	$Round(\mathbf{x}^0)$
	Integer-constrained inversion	Scattering cross section ($\mathbf{t} \in \mathbb{R}^{200}$)	Spherical shell thickness ($\mathbf{x}^0 \in \mathbb{Z}^4$)	Scattering cross section ($\mathbf{x}^L \in \mathbb{R}^{200}$)	$0 \leq 10\mathbf{x}^0 \leq 70$	$\mathbf{x}^0$
Contoning	Inversion (rounded)	Color spectrum ($\mathbf{t} \in \mathbb{R}^{31}$)	Ink layer thickness ($\mathbf{x}^0 \in \mathbb{R}^{11}$)	Color spectrum ($\mathbf{x}^L \in \mathbb{R}^{31}$)	$0 \leq \mathbf{x}^{\mathbf{0}} \leq 30$, $\sum_1^{11} x_i^0 = 30$	$Round(\mathbf{x}^0)$
	Integer-constrained inversion	Color spectrum ($\mathbf{t} \in \mathbb{R}^{31}$)	Ink layer thickness ($\mathbf{x}^0 \in \mathbb{Z}^{11}$)	Color spectrum ($\mathbf{x}^L \in \mathbb{R}^{31}$)	$0 \leq \mathbf{x}^{\mathbf{0}} \leq 30$, $\sum_1^{11} x_i^0 = 30$	$\mathbf{x}^0$
MILP-NA combination	-	Perfect gray spectrum ($\mathbf{t} \in \mathbb{R}^{31}$)	Ink ratios ($\mathbf{x}^0 \in \mathbb{R}^8$)	Gray spectrum ($\mathbf{x}^L \in \mathbb{R}^{31}$)	$0 \leq \mathbf{x}^0 \leq 1$	$\mathbf{x}^0$, Optimality gap
Robustness Analysis	-	Metasurface spectrum ($\mathbf{t} \in \mathbb{R}^{300}$)	4×cylinder height and radius ($\mathbf{x}^{\mathbf{0}} \in \mathbb{R}^8$)	Metasurface spectrum ($\mathbf{x}^L \in \mathbb{R}^{300}$)	$0 \leq \mathbf{x}^{\mathbf{0}} \leq 1$, $\hat{\mathbf{x}}_i^0 - 10^{-3} \leqslant \mathbf{x}_i^0 \leqslant \hat{\mathbf{x}}_i^0 + 10^{-3}$	$\operatorname{argmax} \left\lVert \mathbf{x}^L - \mathbf{t} \right\rVert_1$

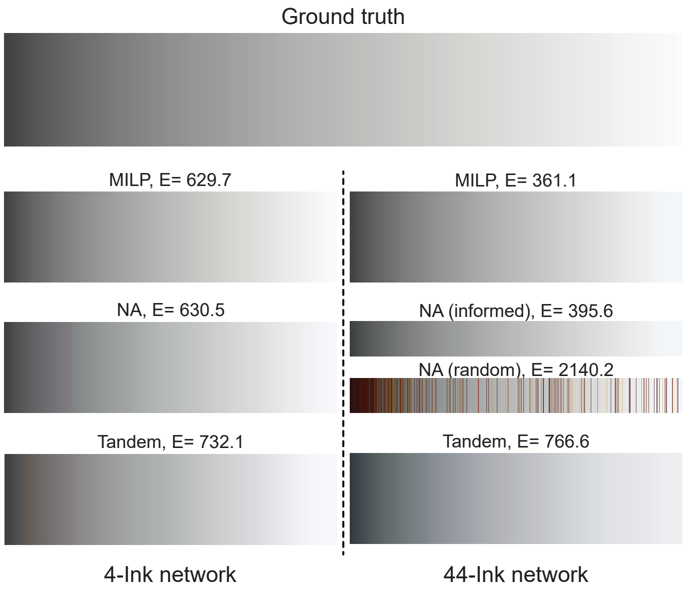


Fig. 2. Different neural inverse methods for spectral separation of a perfect gray ramp. The error (E) is the sum of objective for all 901 gray spectra. We split the NA's solution in the middle to show both random, and domain-knowledge informed initializations.

2020] as it has been shown to significantly outperform other neural inverse methods in literature. Figure 2 visualizes the accuracy of different inverse methods for spectral separation. Our method performs 901 separate optimizations (i.e., Equation 5) in order to find the corresponding ink ratios. For NA, we run the 901 optimizations, each 50 times with different random initialization. We allow up to 2000 iterations of Adam [Kingma and Ba 2014] and terminate the optimization if the solution does not improve within a threshold in 10 consecutive iterations. In the tandem method we query the learned inverse method once using a batch of 901 gray spectra as input.

Looking at the spectral separation accuracies in Figure 2, using the 4-ink PL-NSM, both our method and NA perform very well surpassing the tandem method significantly. In fact, with a gap of 0 between the feasible and relaxed solution of the objective, our method finds the *global* optima for all 901 instances. This indicates that the method of NA has also performed remarkably well as its error is only slightly worse than our method. For a better comparison, we re-run an identical set of experiments using a new PL-NSM with 44 inks as input, i.e., a 44D design space. For this network, our method again finds the global optima for all spectral targets. This time, however, NA struggles to find acceptable solutions for many targets and produces an erroneous reproduction. This indicates that NA's performance drops for higher dimensional design spaces likely due to random initialization.

In a second experiment on NA, instead of random initialization, we initialize the optimization with an *informed* guess. That is, in a crude estimation, we assume that the average spectra of all input inks, i.e., when each ink's contribution is 1/44, is a 50% gray spectrum. Therefore, for reproducing the pure black, i.e., the darkest gray (100%), each ink is initialized with 2/44 ratio, and so on.

With this informed initialization drawn from domain knowledge, the accuracy of NA increases substantially but still trails MILP's performance. This experiment reveals a significant advantage of our method where unlike NA, due to global optimality, increasing the design space dimensionality does not affect the accuracy. Being insensitive to the initialization is another major advantage as using domain knowledge for informed initial guesses is not always feasible.

Using the MILP approach, the optimization of a single gray spectrum takes on average around 256 and 840 seconds for the 4- and 44-ink PL-NSM, respectively. We spend also around 331 and 3.5 seconds on a one-time precomputation of upper and lower bounds of the network nodes. The NA method, on a Titan X GPU, takes on average 62 and 99 seconds for the 4- and 44-ink PL-NSM, respectively. The fastest method is tandem, spending less than 1 second for all spectra, as querying neural network is extremely efficient, though at the cost of significant accuracy loss.

4.2 Soft Robot Inverse Kinematics

Soft robotics is dedicated to studying robots made with flexible materials, with applications in minimally invasive surgeries [Majidi 2014], prosthetics [Polygerinos et al. 2015] and many more. To control the movement of soft robots accurately, an efficient inverse kinematics approach is required. In this section, using our MILP approach, we solve the *neural* inverse kinematics problem for soft robots, introduced by Sun et al. [2021]. The problem involves controlling the soft robot such that it reaches a certain target location. Following [Sun et al. 2021], we consider a snake-like robot made of 103 vertices connected with flexible edges and a fixed bottom (Figure 1b). Among these edges, 40 side edges (colored in Figure 1b) are controllable whose stretches and contractions form our design space (40D). This problem is typically formulated as a PDE-constrained optimization where the design parameters are the boundary conditions, and the solution of the optimization is the position of all the vertices (final shape) [Xue et al. 2020]. As PDE-constrained optimizations are computationally costly, resorting to NSMs is an attractive alternative.

We train a PL-NSM with two hidden layers, each having 128 neurons and ReLU activation functions. To create the training data we solve a PDE-constrained optimization for 50,000 randomly generated stretches and contractions of controllable edges [Xue et al. 2020]. The model's input is the stretches and contractions of 40 controllable edges of the soft robot. The output is the (x, y) coordinates of all 103 vertices, hence, our performance space has 206 dimensions. For the inversion, we would like to optimize the input such that the center of the soft robot's tip reaches the target location $\mathbf{t}$, i.e.,

$$\underset{\mathbf{z}^1,\cdots,\mathbf{z}^{L-1},\quad \mathbf{x}^0,\cdots,\mathbf{x}^L}{\operatorname{argmin}} \left\| \mathbf{x}_i^L - \mathbf{t} \right\|_1, \quad i \in [123,\ 124], \tag{10}$$

where, in our setup, indices 123 and 124 represent the (x, y) location of the robot's tip. Note that Equation 10 is the neural inverse problem objective function (Equation 5a) with a slight modification and is still under the constraints 5b to 5f.

To enforce valid designs, we limit the contractions and stretches (ζ) in the input,

$$\zeta_{\mathbf{min}} \leq \mathbf{x}^0 \leq \zeta_{\mathbf{max}}. \tag{11}$$

The minimum ($\zeta_{\mathbf{min}}$) and maximum ($\zeta_{\mathbf{max}}$) contractions and stretches are among soft robot properties. Following Sun et al. [2021] we set $\zeta_{\mathbf{min}}$ and $\zeta_{\mathbf{max}}$ to -0.2 and 0.2, respectively. Moreover, to avoid non-physical changes, we use a similar term to the one proposed by Sun et al. [2021] which controls the smoothness of the contractions and stretches in consecutive edges:

$$\begin{gathered}\left|(\mathbf{x}_{i+1}^0 - \mathbf{x}_i^0) - (\mathbf{x}_i^0 - \mathbf{x}_{i-1}^0)\right| \leq l_d, \\ i \in (1, n),\ i \neq \frac{n}{2},\ i \neq \frac{n}{2} + 1.\end{gathered} \tag{12}$$

Here l_d is the limit for the deformation and should be determined based on the mechanical properties of the soft robot (flexibility, stress tolerance, etc.). In this experiment, we assume a material that can tolerate a predefined amount of deformation $l_d = 0.2$. Indices $i \in [1, n/2]$ and $i \in [n/2 + 1, n]$ correspond to the edges on the robot's left-hand side and right-hand side, respectively. Moreover, $i = n/2$ and $i = n/2 + 1$ are the end-nodes from two different sides and excluded from Equation 12 as we apply the smoothness terms on each side separately.

Finally, to have a more challenging setup, we seek robot paths that avoid a circular obstacle with radius r. Unlike Sun et al. [2021], we use an $\mathcal{L}_1$ norm for avoiding the obstacle. Using $\mathcal{L}_1$ norm, the vertices avoid the auxiliary, *square* obstacle and thus the chance of intersection between the *edges* and the primary, *circular* obstacle will be reduced significantly. To avoid the obstacle, we define the following constraint

$$\sqrt{2}r \leq \left\| \mathbf{x}_i^L - \mathbf{o} \right\|_1 \tag{13}$$

where $\mathbf{o}$ is the location of the center of the obstacle, r is the obstacle radius, and $\sqrt{2}r$ is half the diagonal of the square encompassing the obstacle. In this experiment we set $r = 0.9$. Note that only a subset of vertices are exposed to the obstacle (highlighted via blue dots in Figure 3), and if they do not collide with the obstacle, the rest of the vertices remain intact. In Equation 13, index i represents the set of such exposed vertices.

We solve this inverse problem using our MILP approach for 1000 randomly sampled target locations. We spend 307 seconds on a one-time network's bound pre-computation. In average, the solve time for each sample is 9.2 seconds. All solutions avoid the obstacle and reach $\mathcal{L}_1$ error of 0. Figure 3 shows 4 randomly chosen solution samples of the MILP optimization. In the first three instances, it seems that the soft robot collides with the $\mathcal{L}_1$ square but a closer look reveals that the vertices, as expected (and guaranteed), are intact. Note that, following the previous work, here we define the constraints based only on vertices, but designing linear constraints for edges to avoid the obstacle is also conceivable. As mentioned earlier, using the $\mathcal{L}_1$ error reduces the chance of the intersection between the edges and the circular obstacle.

We also solve the neural inverse kinematics problem using the NA method. We use the optimization objectives introduced in [Sun et al. 2021] with slight changes such that the losses are comparable with the MILP method (Appendix B). We observe that in the NA-based inversion, all samples avoid the obstacle too, but the average $\mathcal{L}_1$ error

Table 2. Network configuration and calculation time for the experiments of section 4.

Experiment's name	Experiment's mode	Hidden layers configuration	MILP computation time (s)	Bound precomputation time (s)	Other methods (s)
Neural Spectral Separation	4-ink network	100, 100, 100, 100	256	331	NA=62, Tandem<1
	44-ink network	50, 50	840	3.5	NA=99, Tandem<1
Soft Robot Inverse Kinematics	-	128, 128	9.2	307	NA=84.5
Material Selection	Selection	50, 50	5×10^3	3.5	-
	Inversion	50, 50	1.8	3.5	-
Nano-Photonics	-	100, 50, 100	44	4.8	-
Contoning	-	50, 50, 50	40	8	-
Robustness Analysis	-	500, 500, 500, 500	64	1.31×10^4	-

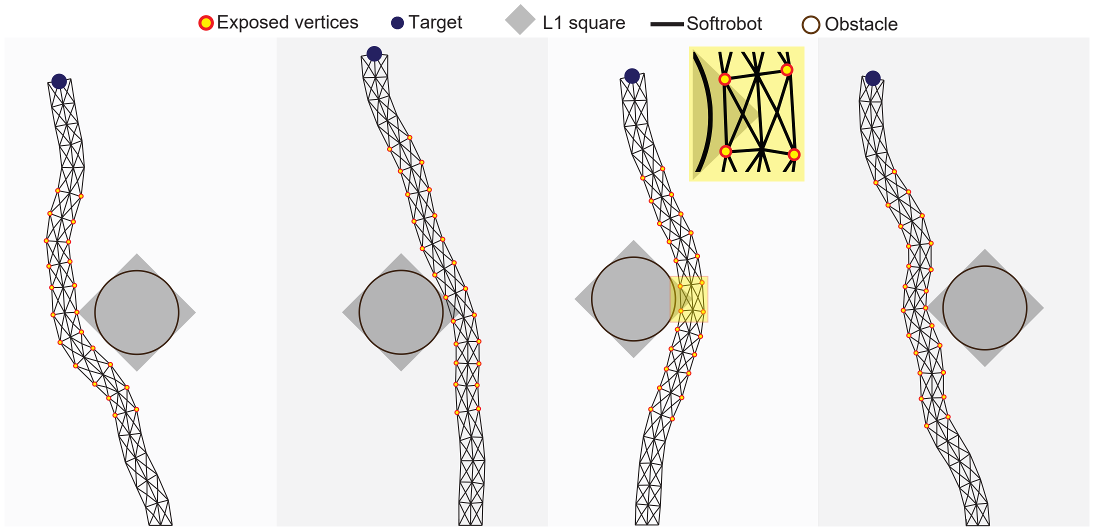


Fig. 3. Soft robot inverse kinematics with MILP. Using the $\mathcal{L}_1$ norm square prevents the collision between robot's *vertices* and the square while significantly reducing the chance of the collision between the *edges* and the (primary) circular obstacle. The highlighted vertices in orange-yellow are those considered in the no-collision term.

is 0.0308 with the average calculation time of 84.5 seconds. Unlike our method, in NA different constraints such as obstacle avoidance and physical deformations are incorporated as *soft* constraints in the energy term with weight hyperparameters to be tuned. This reveals other important advantages of the MILP formulation. First, different constraints can be added to the MILP simply and are guaranteed to be satisfied. Second, no hyperparameter tuning is required. The latter is especially beneficial in real-world problems where there might be many constraints that need to be incorporated in the objective function. By introducing more constraints, it becomes significantly harder to tune the corresponding weights.

4.3 Material Selection

Although digital fabrication technologies, such as multi-material 3D printers, have a limited number of channels, there is a vast array of materials that can fill those channels. Consequently, the question of which subset of materials is optimal for a given task (also known as material selection) is becoming a recurrent question [Ansari et al. 2020; Piovarči et al. 2020; Nindel et al. 2021].

Here we reproduce the results of the duotone reproduction experiment from Ansari et al. [2020] via our approach. The effect of the ink selection is highly visible in duotone experiment and the smallest mistake will stand out prominently. Similar to us, Ansari et al. [2020] employ a MILP formulation for the ink selection. However, they need to develop a custom *linear* forward model that requires deep domain knowledge and specialized measurements. Interestingly, for the actual spectral separation, they train NSMs for the selected inks. Here we show that both spectral separation and the ink selection can be performed via purely data-driven NSMs.

In our duotone reproduction setup, following Ansari et al. [2020], given a spectral image we look for the best pair of inks leading to optimal spectral reproduction from within an ink library of 44 inks. The input is the spectral image of a limited palette watercolor painting from Ansari et al. [2020] shown in Figure 4a. We adopt the PL-NSM (Section 4.1) that predicts the spectrum of a set of 44 library inks. As printed data for training a 44-ink network is not provided, we simulate such data using the Neugebauer model [Yule 1967], an analytical spectral prediction model. The Neugebauer primaries are computed using the multiplication of library ink transmittances.

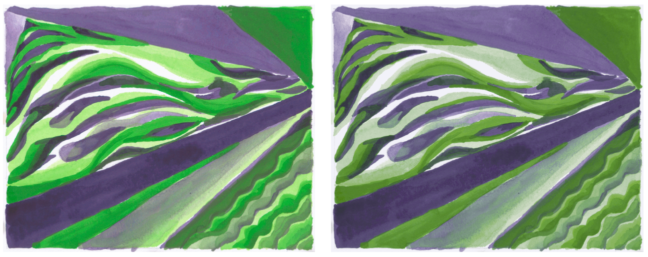


Fig. 4. Given a PL-NSM that predicts the spectrum of a set of 44 inks, our method finds the optimal pair of inks that results in the best spectral reproduction of the input image.

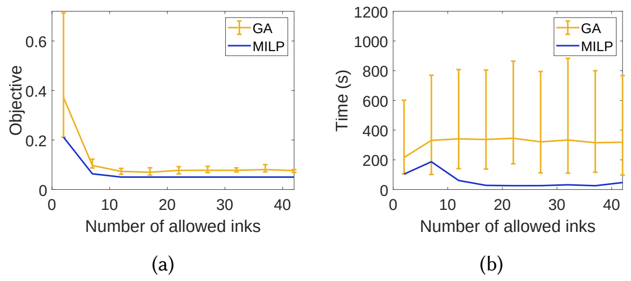


Fig. 5. Comparing our MILP and a GA approach for an ink selection problem. We repeat GA 10 times and report the average, maximum and minimum values.

Since we are looking for a pair of inks that performs well on the whole image, following Ansari et al. [2020], we sample 6 spectra from the input image. We use the method explained in Section 3.2, relying on both Equations 5 and 6. As we use 6 sampled spectra from the input image, we optimize for the 6 targets **t** simultaneously via using a sum in Equation 5a. In fact one can see this problem as solving 6 copies of the network simultaneously for 6 different target spectra all of which must satisfy Equation 6. This means that the variables are almost 6x more than solving for a single target (see Table 1).

Our MILP-based ink selection finds the ground truth inks with a gap of 0, i.e., provably the optimal pair of inks for reproduction of the given input (Figure 4a). Having obtained the two optimal inks, in order to reproduce the input image, we could calibrate a new NSM using the reliable, printed data of those two inks. More interesting is to use the same 44-ink network, this time in a spectral separation configuration, in order to simulate how the pair of optimal inks reproduce the painting (Figure 4b). As we see in Figure 4, the reproduction is of high quality. The quality can be still improved if we calibrate the network with printed rather than simulated data. But the remarkable fact of this experiment remains the globally optimal ink selection on a neural network. The time for solving the ink selection problem is 4998s. The time for the spectral separation is, on average, 1.8 seconds for each spectra, and manageable as there are only 5483 unique colors in the scene. Finally, the one-time precomputation of nodes' bounds took 3.5 seconds.

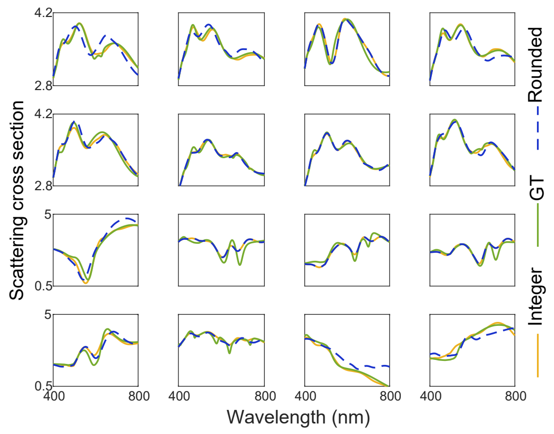
4.3.1 Comparison with Genetic Algorithm. When selecting 2 out of 44 inks, it might be tempting to try a brute-force approach where the objective for each possible pair of inks is computed and then the pair with the best objective is selected. There are however two major caveats. First, although the number of two-ink combinations in a set of 44 inks is reasonable, selecting a larger number of inks via a brute force approach is infeasible. For example, selecting 10 inks amounts to around 2×10^9 combinations, which means we need to perform this number of optimizations to find the objective for each combination. Second, in the absence of a MILP approach, the objective values may not be optimal.

More appropriate solutions to such combinatorial problems are based on stochastic methods, such as genetic algorithm (GA) or simulated annealing. In this section we compare our proposed method to GA for a selection problem. Genetic algorithm searches the combinatorial space stochastically via their well-known heuristics and, in general, prefer combinations with best objectives. For computing the GA's objective, we use the interior point method. In Figure 5 we perform ink selection for a single target spectrum each time allowing for a different number of inks. At each step we repeat the experiment 10 times to capture the variance in GA solutions, and show the average, maximum and minimum values. As shown in Figure 5, the MILP approach always yields the optimal solutions, outperforming GA in both time and accuracy. Note that MILP is considerably faster even though minimum GA computation time on the plot seems to be smaller at around 8 inks. This is because MILP computation time should be compared to the GA's multiple run times until GA converges to a desired solution (due to stochasticity). In this experiment these two methods were evaluated on the same hardware (Intel Xeon CPU E5).

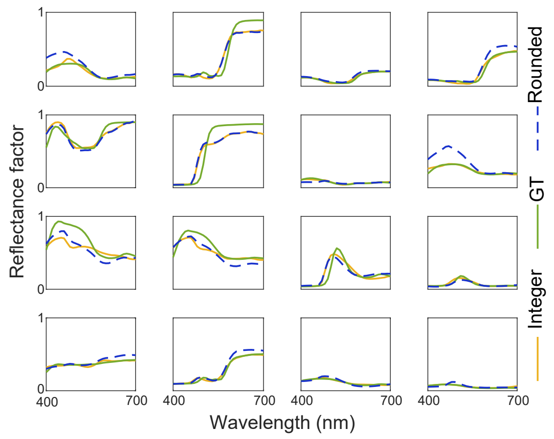
4.4 Integer-Constrained Inverse Design

Apart from material selection, a significant portion of inverse design problems are combinatorial by nature due to the fabrication constraints. For example, *metamaterials* are usually made of juxtaposition of a set of materials (including the void) in 2D or 3D arrays [Bertoldi et al. 2017]. Current approaches assign continuous material properties (such as permittivity) to the elements of these arrays and quantize these values before fabrication. Unfortunately, the quantization step can significantly undo the optimized performance [Zhu et al. 2020]. When the forward model is expressed via a PL-NSM, our combinatorial inverse design formulation can seamlessly take such integer constraints into account. Here, we demonstrate two examples of integer-constrained inverse designs.

4.4.1 Nano-Photonics. In this experiment (Figure 1c), we consider the light scattering from a multilayer dielectric spherical nanoparticle [Peurifoy et al. 2018]. We obtain a different spectral scattering cross section by changing the thickness of the material of each shell. Similar to spectral printing, here we also look for optimal ratios (thicknesses) of the materials which result in a desired spectrum. In order to imitate possible fabrication constraints, we slightly twist the experiment by limiting the materials to take a predefined set of integer thicknesses (from 0 to 70 nm at 10 nm intervals). We train a PL-NSM with 3 layers of 100, 50, and 100 neurons following



(a)



(b)

Fig. 6. (a) Optimizing for integer shell thickness of a photonic nano-sphere, and (b) integer 3D printed layer thickness in contoning. In both experiments, one can ignore the integer constraints and solve for continuous solutions to be rounded to nearest integers. Our method allows for directly optimizing the desired integer values with significant accuracy gain over the rounded solution.

Peurifoy et al. [2018] which maps the combination of 4 materials into the resulting spectrum.

We test our methods on 16 target scattering cross sections shown in Figure 6a and, for all targets, reach the globally optimal solution with average objective of 19.47. For comparison, we also perform the same inverse design (via MILP) on the targets but without enforcing integer thicknesses. After rounding the obtained optimal but continuous thicknesses to the nearest allowed integers, the error increases significantly (30.30). For this particular problem, our MILP-based integer-constrained inversion takes on average 44 seconds. We also spend 4.8 seconds on the one-time bound precomputation of the network nodes.

4.4.2 Contoning. In color reproduction for 3D printing via *contoning* [Babaei et al. 2017], inks with different thicknesses can be superposed. Contoning avoids potential artifacts of the alternative halftoning techniques that rely on spatial multiplexing of materials on the surface. While contoning works well when ink concentrations are low, with highly concentrated inks the quantization artifacts start to appear as the thickness can only be controlled via tuning a discrete number of layers. In an experiment similar to the previous one, we show how our method obtains the best discrete arrangement of different ink layers for reproducing a given spectrum.

Following the setup of Shi et al. [2018], we want to reproduce a target spectrum by superposing 30 layers of 11 different inks. We train a PL-NSM with 3 layers of 50 neurons which maps the layer layouts to the spectrum. Our printer can print 30 layers of 11 different inks (see Table 1 for design constraints). We have performed this experiment with two different settings, in our first attempt (similar to [Shi et al. 2018]) we set the $\mathbf{x^0} \in \mathbb{R}^{11}$. Since the smallest amount of ink that our fabrication device can deposit is a single layer, we have to round the elements of $\mathbf{x^0}$ to the nearest integer neighbors. This rounding step introduces error to our designs. In the second setup, by defining $\mathbf{x^0} \in \mathbb{Z}^{11}$ in our MILP formulation we directly solved the integer inverse problem and found the optimal integer design. In Figure 6b, we plot 16 target and reproduced spectra. We also show the resulting spectra obtained by rounding the optimal continuous layer thicknesses to the closest integers. The average error in this experiment is 1.14 and 1.72 for optimal integer and rounded solutions, respectively. Our MILP-based integer-constrained spectral separation takes on average 40 seconds. The bound precomputation for the considered network takes 8 seconds.

4.5 Combination of MILP and NA

One of the greatest advantages of using MILP for inverse design is its optimality or near optimality guarantees. Despite this advantage, the MILP approach does not scale with larger networks. On the other hand, gradient-based local optimizers, such as NA [Ren et al. 2020] are very efficient even for large networks. As discussed in Section 3.4, NA solutions are feasible solutions to MILP's objective. Therefore, we can produce feasible solutions via NA and continue using MILP for improving the relaxed solution. In Figure 7, we show this approach on 4 randomly target spectra in a spectral separation problem. We use a larger PL-NSM consisting of 4 layers of 150 neurons, mapping an 8D ink ratio input to spectra. In all the experiments, NA improves the feasible solution significantly faster than MILP and by comparing NA solutions with the MILP's relaxed solution we can reach smaller optimality gaps (distance between the yellow and the green lines instead of the distance between blue and green lines) in a considerably less amount of time. Although we use NA for computing feasible solutions, any other methods capable of yielding feasible solutions efficiently, could be used. Finally, we would like to remind that this technique is only suitable for non-combinatorial inverse design problems where all solutions are trivially feasible.

4.6 Robustness Analysis

For analyzing the robustness of different designs (Section 3.5), we use the recently proposed all-dielectric metasurface (ADM) setup [Ren et al. 2020; Nadell et al. 2019]. ADMs are surfaces with nanostructures which possess unique properties like high temperature

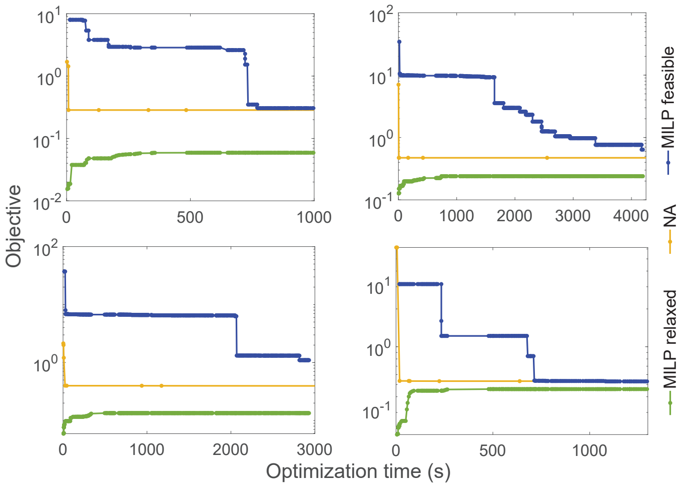


Fig. 7. A NA-MILP hybrid approach. NA closes the optimality gap significantly faster than MILP's own objective's feasible solutions.

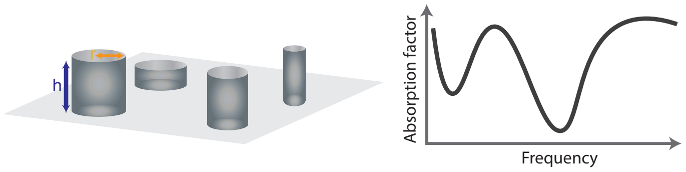


Fig. 8. Schematic representation of an AMD metasurface. By adjusting the sub-wavelength nano-structures (cylinders' heights and radii), we can modulate the spectral absorption profile (right).

resistance, zero ohmic loss, and low thermal conductivity. The properties of AMDs can be modulated by adjusting the structures and arrangement of these nano-structures. In this experiment, the spectral absorption of a particular AMD metasurface can be controlled via changing the heights and radii of nano-cylinders on its surface (Figure 8). As there exist 4 nano-cylinders, the design space can be expressed by 8 parameters. The resulting absorption spectrum is sampled at 300 points.

We train a convolutional PL-NSM that maps 8D designs to 300D spectra. The network is made of 4 fully connected layers each with 500 neurons with ReLU activation functions and batch normalization, followed by 3 deconvolution and 1 convolutional layer. In our experiment, targeting two different spectra, we find a number of corresponding designs via NA which have the best objectives. In Figure 9, we show the (sorted) objectives and corresponding robustness for each solution. While, in Figure 9a, all objectives are comparable, there is one design (solution 19) that has a significantly higher robustness (indicated by a very small circle). This solution is the design of choice for this target spectrum. Moreover, in Figure 9b, sample 1 and 2 give very good accuracy and robustness at the same time and are clearly the most preferred designs. In this experiment, we spend on average 64 seconds on robustness calculation of each design and 3.64 hours on a one-time bound precomputation. We also set ϵ (Equation 9) to 10^{-3}. While here we use a simple perturbation

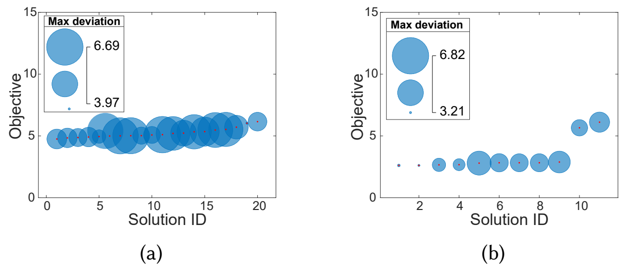


Fig. 9. Robustness analysis of two target spectra of the metasurface design experiment. The size of the circle shows maximum deviation from the objective (Equation 9) and has an inverse relationship with robustness.

model for evaluating the robustness, more sophisticated perturbations, such as erosion and dilation of designs [Sigmund 2009] are possible and left for future work.

4.7 Scalability of MILP-Based Neural Inverse Design

While MILPs are known to be NP-hard problems [Bunel et al. 2020], it is interesting to study their scalability in our context. We choose the neural spectral separation experiment (Section 4.1) as a case study for our scalability analysis. First, in order to see the effect of network's depth on the solve time, we train 4 different PL-NSMs that perform forward spectral prediction for 8 inks. The trained networks have from 6 to 12 hidden layers, each with 50 neurons. In Figure 10a, we show the solve time for each of these networks averaged for 10 target spectra. Similarly, in Figure 10b, we study the effect of network's width, by solving the same spectral separation problem performed on 4 PL-NSMs with a single hidden layer of 100 to 400 neurons. Here also the reported time is the average for 10 different target spectra. Once again, in all experiments, we continue the optimization to reach a duality gap of 0 and thus global optima.

We observe that increasing the depth and width of the network increases the solve time, as expected in MILP problems. This is because each new node (with unstable ReLU) in the network gives rise to an additional binary and continuous variable, as well as new linear and integer constraints, in Equation 5. Note that Figure 10 is indicating the trend of the MILP for a fixed set of target performances and solver settings. MILP solve time heavily depends on the solver's hyperparameters and the layout of the problem (Appendix C). As a result, the absolute time reported in Figure 10 is not necessarily valid for any arbitrary experiment. In practice we have noticed that constrained problems scale better and typically converge faster. For instance, the all-dielectric metasurface (ADM) network is by no means invertible using our method. However, we managed to solve it for the robustness problem by heavily restricting its input domain.

Finally, it is worth mentioning that as long as the network's size is manageable, the underlying complexity of the problem does not affect the optimization complexity.

5 LIMITATIONS AND FUTURE WORK

Due to the NP-hardness of solving MILPs, this method is not suitable for large networks when searching for globally optimal designs. We nevertheless solved some real-life inverse design problems [Ansari

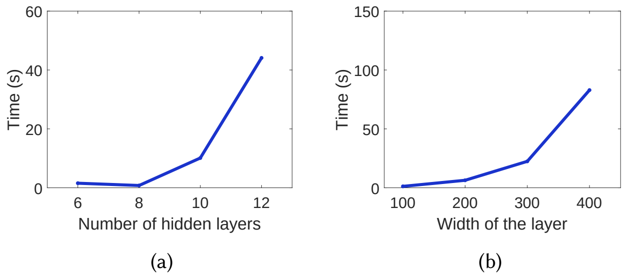


Fig. 10. Scalability of our MILP-based neural inverse design with the depth and width of the network.

et al. 2020; Shi et al. 2018; Peurifoy et al. 2018] using this tool throughout this paper. Looking at Figure 10a, we have found the globally optimal solution through inverting a neural network with 12 hidden layers each having 50 nodes in less than one minute. Thanks to the immense expressive power of neural networks [Hornik et al. 1989], such an architecture is capable of accurately replacing many complex simulations. In fact, in this paper we trained forward models from the literature [Ansari et al. 2020; Shi et al. 2018] with smaller networks. In all these training, we used the same data with the same dimensionality of design and performance spaces and obtained nearly the same training error. In circumstances where using larger networks is necessary, the relaxed solution provided by the MILP solver help making informed decision on early stopping of the optimization. In such cases, the improvement of the feasible solution can also be accelerated via alternative solvers. An interesting direction for future work is to develop a solver customized to the type of inverse problems we deal with. We believe that neural networks with their recursive compositions are amenable to tailored heuristics beyond those found in one-size-fits-all solvers [Gurobi Optimization 2018]. Ironically, machine learning is believed to be a potentially adept tool for discovering such heuristics [Khalil et al. 2017]. Finally, in addition to an accurate neural inversion method, the prerequisite for a perfect neural inverse design pipeline is an accurately trained NSM. While in our evaluations we ensure our trained NSMs are highly accurate, the focus of this work is on accurate optimization whose quality is measurable by the objective value of the optimization. A neural inversion method robust to training imperfections is highly interesting and is left for future work.

6 CONCLUSION

Neural networks are becoming first class citizens when it comes to data-driven modeling in computational design and fabrication. The black box reputation of neural networks has hindered applying them in domains requiring interpretability. While this may to some extent be true during their training, once trained, neural networks are rather well-behaved mathematical functions. In this work we showed that leveraging the underlying mathematical structure of neural surrogate models leads to a tool with many attractive properties. We believe our work paves the way for making sense of data-driven design processes in a more systematic manner.

ACKNOWLEDGMENTS

This work benefited greatly from early discussions with Omid Alizade-Mousavi on the technicalities of MILP reformulation. We are grateful to Nima Vahidi-Ferdowsi and Soroush Nikjou for organizing the code and data, and Hossein Hayati for his help with the nano-photonics experiment. We thank Azadeh Asadi for creating the painting in Figure 4, and Paul Swoboda and anonymous reviewers for their useful feedback.

REFERENCES

Navid Ansari, Omid Alizadeh-Mousavi, Hans-Peter Seidel, and Vahid Babaei. 2020. Mixed integer ink selection for spectral reproduction. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.
Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. 2019. Analyzing Inverse Problems with Invertible Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJed6j0cKX
Vahid Babaei, Kiril Vidimče, Michael Foshey, Alexandre Kaspar, Piotr Didyk, and Wojciech Matusik. 2017. Color contoning for 3D printing. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–15.
Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. 2013. Mixed-integer nonlinear optimization. *Acta Numerica* 22 (2013), 1–131.
Amit H Bermano, Thomas Funkhouser, and Szymon Rusinkiewicz. 2017. State of the art in methods and representations for fabrication-aware design. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 509–535.
Katia Bertoldi, Vincenzo Vitelli, Johan Christensen, and Martin Van Hecke. 2017. Flexible mechanical metamaterials. *Nature Reviews Materials* 2, 11 (2017), 1–11.
Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Pushmeet Kohli, P Torr, and P Mudigonda. 2020. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research* 21, 2020 (2020).
Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. 2018. A Unified View of Piecewise Linear Neural Network Verification. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc., 4790–4799. https://proceedings.neurips.cc/paper/2018/file/be53d253d6bc3258a8160556dda3e9b2-Paper.pdf
Desai Chen, David IW Levin, Piotr Didyk, Pitchaya Sitthi-Amorn, and Wojciech Matusik. 2013. Spec2Fab: a reducer-tuner model for translating specifications to 3D prints. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 135.
Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. 2017. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 251–268.
Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using Real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=HkpbnH9lx
Matteo Fischetti and Jason Jo. 2018. Deep neural networks and mixed integer linear optimization. *Constraints* 23, 3 (2018), 296–309.
Christodoulos A Floudas. 1995. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press.
Konstantinos Gavriil, Ruslan Guseinov, Jesús Pérez, Davide Pellis, Paul Henderson, Florian Rist, Helmut Pottmann, and Bernd Bickel. 2020. Computational design of cold bent glass façades. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.
LLC Gurobi Optimization. 2018. Gurobi Optimizer Reference Manual. http://www.gurobi.com
Robert Hecht-Nielsen. 1992. Theory of the backpropagation neural network. In *Neural networks for perception*. Elsevier, 65–93.
Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.
Jiaqi Jiang, Mingkun Chen, and Jonathan A Fan. 2020. Deep neural networks for the evaluation and design of photonic devices. *Nature Reviews Materials* (2020), 1–22.
Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning Combinatorial Optimization Algorithms over Graphs. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/d9896106ca98d3d05b8cbdf4fd8b13a1-Paper.pdf
Yashar Kiarashinejad, Sajjad Abdollahramezani, and Ali Adibi. 2020. Deep learning approach based on dimensionality reduction for designing electromagnetic nanostructures. *npj Computational Materials* 6, 1 (2020), 1–12.
Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
Ed Klotz and Alexandra M Newman. 2013. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science* 18, 1-2 (2013), 18–32.
Dianjing Liu, Yixuan Tan, Erfan Khoram, and Zongfu Yu. 2018. Training deep neural networks for the inverse design of nanophotonic structures. *ACS Photonics* 5, 4 (2018), 1365–1369.
Johan Lofberg. 2004. YALMIP: A toolbox for modeling and optimization in MATLAB. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*. IEEE, 284–289.
Carmel Majidi. 2014. Soft robotics: a perspective—current trends and prospects for the future. *Soft robotics* 1, 1 (2014), 5–11.
Harry M Markowitz and Alan S Manne. 1957. On the solution of discrete programming problems. *Econometrica: journal of the Econometric Society* (1957), 84–110.
Wojciech Matusik, Boris Ajdin, Jinwei Gu, Jason Lawrence, Hendrik P. A. Lensch, Fabio Pellacini, and Szymon Rusinkiewicz. 2009. Printing Spatially-varying Reflectance. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 128:1–128:9.
Niloy J Mitra and Mark Pauly. 2009. Shadow art. *ACM Transactions on Graphics* 28, CONF (2009), 156–1.
Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. 2014. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*. 2924–2932.
Christian C Nadell, Bohao Huang, Jordan M Malof, and Willie J Padilla. 2019. Deep learning for accelerated all-dielectric metasurface design. *Optics express* 27, 20 (2019), 27523–27535.
Thomas Nindel, Tomáš Iser, Tobias Rittig, Alexander Wilkie, and Jaroslav Křivánek. 2021. A Gradient-Based Framework for 3D Print Appearance Optimization. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).
John Peurifoy, Yichen Shen, Li Jing, Yi Yang, Fidel Cano-Renteria, Brendan G DeLacy, John D Joannopoulos, Max Tegmark, and Marin Soljačić. 2018. Nanophotonic particle simulation and inverse design using artificial neural networks. *Science advances* 4, 6 (2018), eaar4206.
Michal Piovarči, Michael Foshey, Vahid Babaei, Szymon Rusinkiewicz, Wojciech Matusik, and Piotr Didyk. 2020. Towards spatially varying gloss reproduction for 3D printing. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–13.
Panagiotis Polygerinos, Zheng Wang, Kevin C Galloway, Robert J Wood, and Conor J Walsh. 2015. Soft robotic glove for combined assistance and at-home rehabilitation. *Robotics and Autonomous Systems* 73 (2015), 135–143.
Simiao Ren, Willie Padilla, and Jordan Malof. 2020. Benchmarking Deep Inverse Models over time, and the Neural-Adjoint method. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 38–48. https://proceedings.neurips.cc/paper/2020/file/007ff380ee5ac49ffc34442f5c2a2b86-Paper.pdf
Christian Schüller, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Appearance-mimicking surfaces. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–10.
Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. 2015. Microstructures to control elasticity in 3D printing. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 136.
Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, and Mark Pauly. 2014. High-contrast computational caustic design. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.
Liang Shi, Vahid Babaei, Changil Kim, Michael Foshey, Yuanming Hu, Pitchaya Sitthi-Amorn, Szymon Rusinkiewicz, and Wojciech Matusik. 2018. Deep multispectral painting reproduction via multi-layer, custom-ink printing. *ACM Trans. Graph.* 37, 6 (Dec. 2018), 271:1–271:15.
Ole Sigmund. 2009. Manufacturing tolerant topology optimization. *Acta Mechanica Sinica* 25, 2 (2009), 227–239.
Denis Sumin, Tobias Rittig, Vahid Babaei, Thomas Nindel, Alexander Wilkie, Piotr Didyk, Bernd Bickel, Jaroslav Křivánek, Karol Myszkowski, and Tim Weyrich. 2019. Geometry-aware scattering compensation for 3D printing. *ACM Trans. Graph.* 38, 4 (2019).
Xingyuan Sun, Tianju Xue, Szymon M Rusinkiewicz, and Ryan P Adams. 2021. Amortized Synthesis of Constrained Configurations Using a Differentiable Surrogate. *arXiv preprint arXiv:2106.09019* (2021).
Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. 2019. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HyGIdiRqtm
Chelsea Tymms, Siqi Wang, and Denis Zorin. 2020. Appearance-preserving tactile optimization. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.
Juan Pablo Vielma. 2015. Mixed integer linear programming formulation techniques. *Siam Review* 57, 1 (2015), 3–57.
Tianju Xue, Alex Beatson, Sigrid Adriaenssens, and Ryan Adams. 2020. Amortized finite element analysis for fast PDE-constrained optimization. In *International Conference on Machine Learning*. PMLR, 10638–10647.
John AC Yule. 1967. *Principles of color reproduction: applied to photomechanical reproduction, color photography, and the ink, paper, and other related industries.* Wiley New York.
Ziwei Zhu, Utsav D Dave, Michal Lipson, and Changxi Zheng. 2020. Inverse geometric design of fabrication-robust nanophotonic waveguides. In *2020 Conference on Lasers and Electro-Optics (CLEO)*. IEEE, 1–2.

A BOUND PRECOMPUTATION ALGORITHM

ALGORITHM 1: Nodes' Lower and Upper Bound Precomputation.

```
Input
$F_\theta$   // Trained neural network
$t_{\max}$   // Time limit for optimization
Design constraints   // e.g., fabrication constraints
Output
Constraints   // The set of all constraints including
   the upper and lower bounds
begin
    Constraints ⟵ Design constraints
    for $l$ ⟵ 1 to $L$ do
        // Layers
        Optimizer ⟵ Constraints   // Updating the
           optimizer with new constraints after
           proceeding to the next layer
        for $k$ ⟵ 1 to $K$ do
            // Nodes at layer $l$
            Start Timer
            Optimizer ⟵ Obj (Equation 7)
            while Optimizer do
                if $Timer \geq t_{max}$ or $gap == 0$ then
                    $\mathbf{l}_k^l = MILP_{relaxed}$   // Relaxed solution
                       determines the bound
                    Constraints ⟵ $\mathbf{l}_k^l$
                    Break   // Reaching the time limit or
                       finding the optimal solution stops
                       the optimization
                end
            end
            Start Timer
            Optimizer ⟵ Obj (Equation 8)
            while Optimizer do
                if $Timer \geq t_{max}$ or $gap == 0$ then
                    $\mathbf{u}_k^l = MILP_{relaxed}$
                    Constraints ⟵ $\mathbf{u}_k^l$
                    Break
                end
            end
        end
    end
end
```

B SOFT ROBOT INVERSE KINEMATICS OBJECTIVE FOR NEURAL ADJOINT METHOD

We have also solved the inverse kinematics soft robot problem using the method of NA. We use the objective function introduced in [Sun et al. 2021] with slight changes such that we can compare the errors with the results of the MILP method. The objective function is made of three terms:

$$\mathcal{L}_g(\boldsymbol{\theta}, \boldsymbol{u}) := \left\| \mathbf{x}_i^L - \mathbf{t} \right\|_1 + \lambda_1 \cdot \mathcal{B}\left(\mathbf{x}^L, \mathbf{o}\right) + \lambda_2 \cdot \mathcal{R}(\mathbf{x}^0) \quad i \in [123,\ 124], \tag{14}$$

$$\mathcal{B}\left(\mathbf{x}^L, \mathbf{o}\right) := \sum_{i=1}^{m} \left(\max\left(r \cdot \sqrt{2} + \Delta r - \left\| \mathbf{x}_i^L - \mathbf{o} \right\|_1, 0 \right) \right)^2, \tag{15}$$

$$\mathcal{R}(\mathbf{x}^0) := \sum_{\substack{1<i<n, i \neq n/2, \\ i \neq n/2+1}} \left(\frac{\mathbf{x}_{i+1}^0 - \mathbf{x}_i^0}{2} - \frac{\mathbf{x}_i^0 - \mathbf{x}_{i-1}^0}{2} \right)^2. \tag{16}$$

The $\mathcal{L}_1$ norm in Equation 14 is responsible for decreasing the distance between the tip of the soft robot and the target point. Equation 15 punishes the possible collision of the soft robot with the obstacle and Equation 16 encourages the smoothness of the soft robot. Finally, λ_1 and λ_2 are hyperparameters for balancing the objective, which we tune to 0.9 and 0.1, respectively.

C IMPLEMENTATION DETAILS

In this work neural networks are trained using Pytorch library on TITAN X GPU. The weights and biases of the network are then imported to MATLAB. We can use them to formulate our inversion problem using Yalmip [Lofberg 2004], and solve it using Gurobi[Gurobi Optimization 2018]. Yalmip is a MATLAB toolbox for rapid prototyping of optimization problems, and Gurobi is a state of the art MILP solver.

Mixed integer neural inverse design requires MILP representation (Equation 5) of the neural network and the constraints of the specific problem at hand (Table 1, column 6). First, we need to compute the upper and lower bound of each node using Algorithm 1. Having computed the bounds, we can start the optimization and calculate our optimal design (Table 1, column 7).

It is possible that the neural network is too large to be solved efficiently using mixed integer neural inverse design. However, the size of the network is not the only factor behind the scalability of our method. In practice, for some problems in which the size of the network is on the verge of tractability, changing the target performance influences the solve time significantly. Although the CPU type and number of cores could affect the efficiency of the solver, the effect is not consistent. Surprisingly, factors like the order of the constraints can change the MILP solve time. This sensitivity comes from numerous heuristics (and consequently a huge number of hyperparameters) running under the hood of Gurobi. Unfortunately, there is not a single combination that works best for either all the problems or all the targets for one problem. To overcome the effect of these uncertainties and to keep the hyperparameters manageable, we kept the recommended settings of Gurobi and reported the average time on a batch of data.