

A Neuro-symbolic Approach to Inverse Design of Thin-layer Metamaterials Under Layout Constraints

Supplementary Material

Abstract

This document reports technical details, experimental settings, and additional experimental results for our submission “A Neuro-symbolic Approach to Inverse Design of Thin-layer Metamaterials Under Layout Constraints”. The code to reproduce experiments will be provided in a public repository upon acceptance.

This document is organized as follows: Section A provides a detailed description of existing inverse design architectures that were used for benchmarking our approach; Section B reports insights on the datasets and details about metrics and hyperparameters for reproducibility; Sections C and D discuss experiment’s hardware, computation time and additional results.

A Benchmark Models

In this section, we review the state-of-the-art architectures that serve as baselines for our benchmarking on the metamaterial inverse design task.

A.1 Neural Adjoint (NA)

It is part of a family of gradient-based inverse design methods [1, 2, 3]. They use the pre-trained network N_F (as a surrogate simulator), which is frozen during the inverse computation phase. In such a phase, the network takes as input an initial randomly sampled guess \bar{x}_0 of trainable weights. The loss function is meant to minimize the distance between \bar{y} and $N_F(\bar{x}_0)$, and it is optimized via backpropagation by directly updating the weights of \bar{x}_0 (keeping N_F frozen). Eventually, the resulting design \bar{x} (such that $F(\bar{x}) \approx \bar{y}$) is given by the values of the weights in \bar{x}_0 after their optimization. A known limitation of this approach is that the search space defined by \bar{x}_0 is often narrow, leading to convergence to suboptimal local minima [4] and hence, making NA particularly sensitive to the initialization of \bar{x}_0 . To mitigate this, [5] proposes an extension that resamples the \bar{x}_0 multiple times. For a given target response \bar{y} , NA repeats T times the optimization of \bar{x}_0 , starting from different random initializations of its weights. In addition, [5] introduces a *boundary loss* \mathcal{L}_{bnd} to constrain the final design \bar{x} to be a normally distributed variable. However, this constraint is tailored for real-valued design spaces and does not directly suit metamaterials, where the design involves a binary layout matrix M representing material assignments to layers. For this reason, in our experiments, we replace \mathcal{L}_{bnd} with a one-hot encoding loss \mathcal{L}_{oh} , specifically designed for metamaterial design as proposed in [6]. This loss encourages valid material assignments by enforcing one-hot encodings across the rows of the real-valued \bar{M} . Figure 1 illustrates how we adapt the general NA framework to the metamaterial inverse design in our experimental setting.

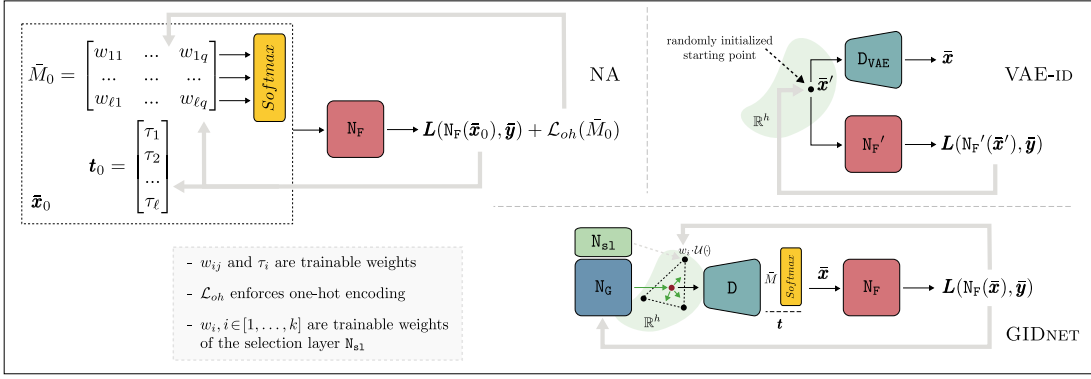


Figure 1: NA, VAE-ID and GIDNET frameworks for the metamaterial design.

A.2 VAE-ID

Originally proposed for molecule inverse design in [7], this architecture jointly trains a variational autoencoder [8] and a variant N_F' of a surrogate model to guide optimization during the inverse computation. In particular, the variational autoencoder is an encoder–decoder pair (E_{VAE}, D_{VAE}). The encoder E_{VAE} maps an input x to the parameters of a multivariate Gaussian distribution, defining a continuous latent representation of dimension h . A latent vector $x' \in \mathbb{R}^h$ is then sampled from this distribution, typically using the reparameterization trick [8]. The decoder $D_{VAE} : \mathbb{R}^h \rightarrow \mathbb{R}^{\ell \times (q+1)}$ generates a candidate design from x' . On the other hand, $N_F' : \mathbb{R}^h \rightarrow \mathbb{R}^m$ maps the latent representation of the input x' to the property y . During training VAE-ID aims to jointly optimize E_{VAE} , D_{VAE} , and N_F' on reconstruction and property prediction—to let the latent space be a continuous representation conditioned by the properties.

By leveraging such representation, in the inverse computation phase VAE-ID initially samples a random point (or T points in the case of resampling strategy) from the latent design space. Such starting point is then provided in input to D_{VAE} to generates a candidate design \bar{x} . The loss function is meant to minimize the distance between \bar{y} and $N_F'(\bar{x})$. To this aim, both D_{VAE} and N_F' are frozen, and the design is optimized by directly moving \bar{x}' to explore the latent space. At the end of this optimization, the final latent design is eventually decoded. Interestingly, no constraints are enforced during the exploration process; instead, the validity of the final designs is assessed only ex-post.

In the application to metamaterials design, no modification is needed to enforce onehot encoding of the generated metamaterial. Indeed, while in the other architectures, this is done to let suitably defined inputs to N_F , in this case, N_F' is trained to work with continuous representations coming from the latent design space.

A.3 GIDNET

It is a recently proposed approach to inverse design proposed in [6]. During the training phase, GIDNET constructs a latent space using an autoencoder composed of an encoder–decoder pair (E, D). The encoder E maps an input x to its latent representation $x' \in \mathbb{R}^h$, while the decoder D attempts to reconstruct the original input. When required, the decoder is further trained to enforce categorical structure in the reconstructed design. To this end, the authors introduce a custom loss function \mathcal{L}_{oh} , which penalizes continuous outputs that deviate from a one-hot encoding.

During the inverse computation phase, GIDNET employs a dedicated mechanism known as *Selection Layer* to identify a suitable region of the latent space to explore. This is achieved by selecting a set of k candidate designs—typically the k nearest neighbors to the target response \bar{y} in the training dataset—and computing a linear combination of their latent representations. Each candidate is weighted by a trainable parameter in the layer N_{s1} , allowing the model to flexibly explore the latent space around a meaningful region. From

this initialization point, normally distributed random noise is added and passed through the generator N_G , which perturbs the point in multiple directions within the latent space to produce a diverse set of candidate solutions. These latent candidates are then decoded via D into the original design space and subsequently evaluated by the surrogate model N_F . The loss function—designed to minimize the distance between the predicted response $N_F(\bar{x})$ and the target response \bar{y} , guiding the exploration. During this process, D and N_F are kept frozen, while the parameters of the generator N_G and the k weights in N_{s1} are updated to learn meaningful perturbations that improve design quality in the latent space.

To ensure a fair comparison with other methods, such as NA and VAE-ID, which permit resampling of initialization points, we adapt GIDNET by modifying its initialization strategy. Specifically, instead of selecting the k nearest neighbors to the target response \bar{y} , we uniformly sample k latent vectors within the bounds of the training set’s distribution in the latent space. Such points are then combined to define a region of the latent space from which the exploration is initialized, as shown in Figure 1 (for $k = 3$).

Notably, GIDNET has demonstrated superior performance in several real-valued inverse design problems, as well as in the inverse design of metamaterials, making it the state-of-the-art architecture in this domain [6].

B Experimental Setting

B.1 Datasets

In our experiments, we use two state-of-the-art datasets for the inverse design of metamaterials. Both datasets consist of metamaterial-response pairs where each metamaterial x_i structure is associated with an optical response y_i , obtained via the *transfer matrix method* (TMM) [9]. However, they differ in the number of material layers, and the dimensionality of the optical response:

- $\mathcal{D}_{\ell=5}$ proposed in [10], in this dataset structures are made of 5 layers, and the materials set of 5 choices is defined as $\mathcal{M} = \{Ag, Al_2O_3, ITO, Ni, TiO_2\}$. Each layer thickness is defined in the range $[1, 60]$ nm. The input space is therefore $\mathbb{R}^{5 \times (5+1)}$. Each structure is associated with reflectance and transmittance spectra, for different polarizations, incident angles for 200 equally spaced points over the range $[450, 950]$ nm (with values in $[0, 1]$). The output space is $\mathbb{R}^{2 \times 2 \times 3 \times 200}$. In our experiments on this dataset, we used 219500 examples as training-set and 500 examples as test-set.
- $\mathcal{D}_{\ell=10}$: proposed in [11], in this dataset structures are made of 10 layers¹, and the materials set of 7 choices² is defined as $\mathcal{M} = \{ZnO, AlN, Al_2O_3, MgF_2, SiO_2, TiO_2, SiC\}$. Each layer thickness is defined in the range $[0, 1]$. The input space is therefore $\mathbb{R}^{10 \times (7+1)}$. The response y is a 2001-dimensional real-valued vector representing the average spectral reflectivity averaged over two polarizations, for different incident angles across 2001 equally-spaced wavelengths, in range $[0.3, 20]$ μ m. In our experiments, we used 44300 examples of the dataset as training-set and 100 examples as test-set.

B.2 Architectures Instantiation

For our experiments, we instantiated three state-of-the-art output-dependent architectures: NA, VAE-ID, and GIDNET. These architectures were evaluated on the two metamaterial datasets for the inverse design task, comparing the performance of their original (*baseline*) implementations against their variant incorporating the L^s term during the inverse computation phase. Architectures’ components instantiations follow.

¹Technically, these metamaterials consist of 11 layers, but the final layer is always *Ag* with a thickness of 0.1 μ m and is not part of the design space.

²*Ag* appears exclusively in the final layer, hence it is excluded from the set of available materials.

Surrogate models All inverse design methods considered rely on a surrogate simulator model to evaluate and optimize candidate solutions during the inverse computation phase. We hence trained two surrogate, N_F and N'_F , on each of the datasets $\mathcal{D}_{\ell=5}$, $\mathcal{D}_{\ell=10}$. The training objective for all the surrogates was to minimize the mean squared error loss function (MSE) between the predicted and the ground-truth spectral responses. To ensure a fair comparison across methods, we use the same surrogate model architecture—with the same number of network parameters—for all the experiments.

For $\mathcal{D}_{\ell=5}$, we adopt the same N_F architecture—matching the number and configuration of neural network layers—used in [6]. The only exception is for VAE-ID, which operates in a latent space; in this case, the input layer of N'_F is adjusted to match the latent dimensionality h . Model selection for N_F was performed on $\mathcal{D}_{\ell=5}$ via grid search over the following hyperparameter space: learning rate $lr \in \{0.001, 0.005, 0.01, 0.05\}$, number of training epochs $e \in \{50, 100, 150, 200\}$, and batch size $bs \in \{256, 512, 1024\}$. The goal was to identify the configuration yielding the best predictive performance, measured in terms of MSE. The best-performing configuration was $lr = 0.005$, $e = 150$, and $bs = 1024$. The hyperparameter search was therefore aimed at identifying the configuration that achieved the lowest predictive MSE. The best-performing configuration was $lr = 0.005$, $e = 150$, and $bs = 1024$. A learning rate scheduling strategy (ReduceLROnPlateau) with a patience of 10 epochs was applied in all training runs. The final N_F models trained on $\mathcal{D}_{\ell=5}$ achieved an MSE of 0.0003 on the test set, with spectral responses y_i normalized to the range $[0, 1]$.

For $\mathcal{D}_{\ell=10}$ we configured N_F as feed forward neural network of 3 fully connected subsequent layers of 80, 420, 640, 2001, 2001 neurons respectively. Again for VAE-ID, the input layer of N'_F is adjusted to match the latent dimensionality h . Model selection for N_F was performed on $\mathcal{D}_{\ell=10}$ via grid search over the same hyperparameter space (and ReduceLROnPlateau strategy) defined above for $\ell = 5$. The configuration yielding the best MSE was $lr = 0.005$, $e = 200$, $bs = 256$. The final N_F models trained on $\mathcal{D}_{\ell=10}$ achieved an MSE of 0.0028 on the test set, with spectral responses y_i originally defined in the range $[0, 1]$.

Autoencoders The VAE-ID and GIDNET methods require a pretrained autoencoder (variational in the case of VAE-ID) to reconstruct the latent representations of metamaterials. To ensure a fair comparison across methods, we employ the same Encoder–Decoder architecture—with an identical number of network parameters—for each experiment on dataset $\mathcal{D}_{\ell=i}$, where $i \in 5, 10$. For both datasets, the architecture follows the parametric configuration proposed in [6]. According to this configuration, the dimensionality of the latent space (defined as $\ell \times 3$) results in $h = 15$ for $\mathcal{D}_{\ell=5}$ and $h = 30$ for $\mathcal{D}_{\ell=10}$. Since VAE-ID relies on a variational autoencoder, its architecture was modified to include two additional linear layers that map the Encoder’s output to the mean and log-variance parameters of the latent Gaussian distribution, following the original formulation of the variational autoencoder [8]. Both the GIDNET autoencoder and VAE-ID variational autoencoder were trained to minimize the reconstruction error, using a composite loss function consisting of column-wise categorical cross-entropy for the materials matrix M and MSE for the thickness vector \mathbf{t} . In the case of GIDNET, we additionally include the one-hot regularization term introduced by the authors in [6], while for VAE-ID, we incorporate the Kullback–Leibler divergence term as defined in the original variational framework. For both datasets and methods, we performed grid-search on a hyperparameter space defined by: $lr \in \{0.001, 0.005, 0.01, 0.05\}$, $e \in \{50, 100, 150\}$, and $bs \in \{128, 256, 512, 1024\}$. For the autoencoder on $\mathcal{D}_{\ell=5}$, the best-performing configuration was $lr = 0.001$, $e = 150$, and $bs = 1024$, achieving a material assignment accuracy of 1.000 (i.e., the average proportion of correctly assigned materials per layer) and a thickness reconstruction MSE of 1.41×10^{-4} on the test set. For the variational autoencoder on $\mathcal{D}_{\ell=5}$, the optimal configuration was $lr = 0.001$, $e = 150$, and $bs = 256$, with a reconstruction MSE of 0.035. On $\mathcal{D}_{\ell=10}$, the best AE configuration remained the same ($lr = 0.001$, $e = 150$, $bs = 1024$), achieving an accuracy of 1 and a reconstruction MSE of 1.08×10^{-3} on the test set. For the VAE on $\mathcal{D}_{\ell=10}$, the best setup was $lr = 0.001$, $e = 100$, and $bs = 256$, resulting in a reconstruction MSE of

Table 1: Execution times (in seconds) for the inverse computation of a single metamaterial, $e = 200$ and $T = 1$

	$\mathcal{D}_{\ell=5}$						$\mathcal{D}_{\ell=10}$					
	NA		VAE-ID		GIDNET		NA		VAE-ID		GIDNET	
	baseline	with L^s	baseline	with L^s	baseline	with L^s	baseline	with L^s	baseline	with L^s	baseline	with L^s
UA	3.42	15.97	3.05	13.93	6.02	16.65	1.38	261.85	1.17	245.37	4.01	263.23
NA	3.68	14.28	3.98	14.47	6.03	14.06	1.15	45.42	1.23	44.04	3.63	46.40
PAL2	3.37	14.98	3.29	14.29	5.64	16.27	1.24	26.43	1.13	27.06	3.56	27.08
PAL3	—	—	—	—	—	—	1.23	55.95	1.13	56.53	3.57	58.45
PAL4	—	—	—	—	—	—	1.24	186.59	1.13	187.61	3.57	204.47
P2	3.44	13.53	3.38	13.67	5.63	12.72	1.24	34.34	1.14	35.35	3.61	35.83
P3	3.14	13.93	3.27	13.91	5.27	14.37	1.21	65.09	1.12	65.68	3.57	68.33
P4	—	—	—	—	—	—	1.24	109.33	1.13	110.18	3.59	113.00

4×10^{-4} on the test set.

Other components GIDNET uses two additional components: a *Selection Layer* N_{s1} and a generator N_G to explore the latent space. Both components’ configurations are taken from the best results in the original work [6]. N_G is implemented for both datasets as a fully connected neural network of 2 layers with $6 \cdot \ell$ and $3 \cdot \ell$ neurons. The dimensionality of N_{s1} is $k = 30$.

B.3 Metrics

For the evaluation of our approach we used three metrics, namely, spectral root mean squared error (srmse) and one-hot as defined in [6], and valid percentage of materials.

With the latter metrics, we evaluate the percentage of valid materials over the set of T initialization points. Let α be the number of samples in the test set. We recall that inverse design is performed T times for each element in the test set. Let \mathcal{V}_i^ϕ be the set of valid materials generated for the i -th element in the test set for a constraint ϕ .

For a given constraint ϕ :

$$\text{valid}_\phi(\%) = \frac{1}{\alpha} \sum_i^\alpha \frac{|\mathcal{V}_i^\phi|}{T} \times 100$$

with α number of samples in the test set, and \mathcal{V}_i^ϕ set of valid materials satisfying ϕ , out of the T initialization points for the i -th sample in the test set.

C Hardware and Timing

We conducted comparative experiments on the time requirement in the same Python environment on Ubuntu 22.04.01, over four 12-core Intel(R) Xeon(R) Gold 5118 CPUs (2.30GHz), 504GB RAM, and two NVIDIA Tesla V100 GPUs (16 GB each). The results are reported in Table 1. It is worth noticing that, while VAE-ID and NA optimize multiple starting points in parallel, GIDNET optimizes the same points sequentially. This causes the inverse design runtime of GIDNET to increment linearly with the number of starting points.

The code was developed in Python 3.12.9 and relies on key libraries such as PyTorch (version 2.6.0). A comprehensive list of all packages and their exact versions is provided in the `requirements.txt` file. All the experiments are fully reproducible and random seeds have been properly defined for this purposes in the code. Detailed instructions to reproduce the experiments can be found in the `README.md` file within the code repository, which is included as supplementary material and will be made publicly available upon acceptance.

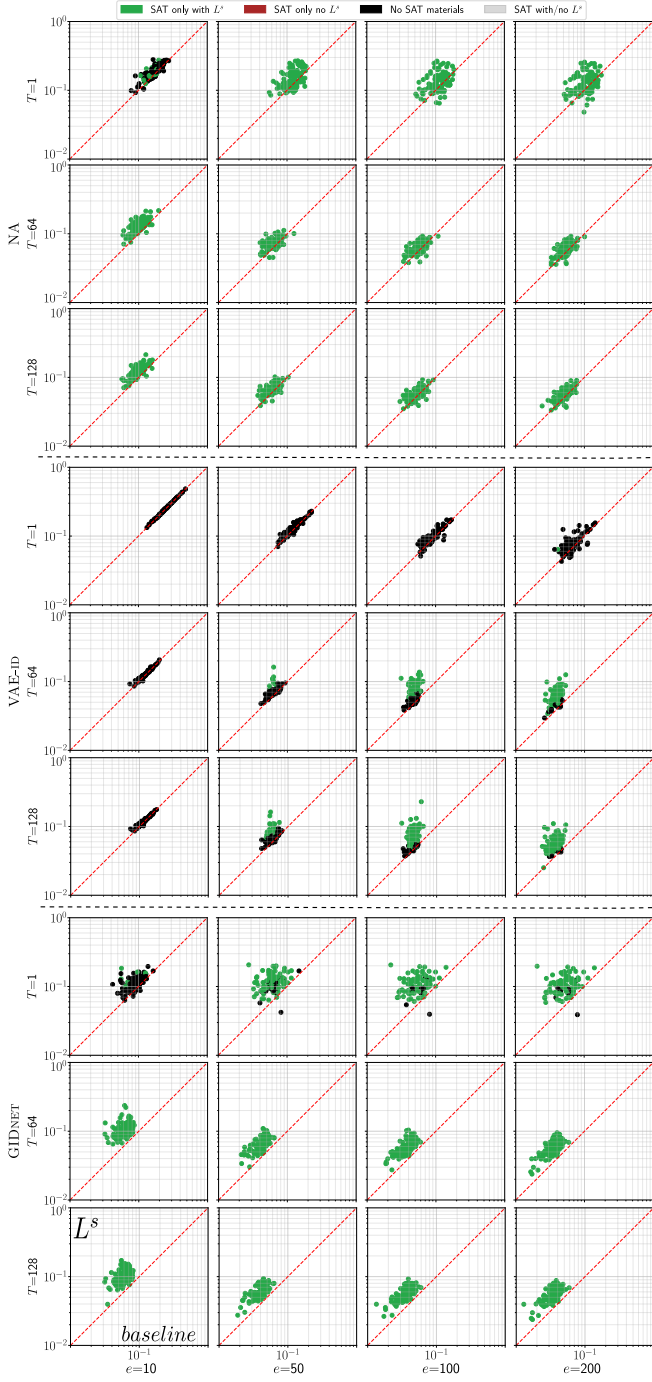


Figure 2: Results for the P4 layout constraint on $\mathcal{D}_{\ell=10}$.

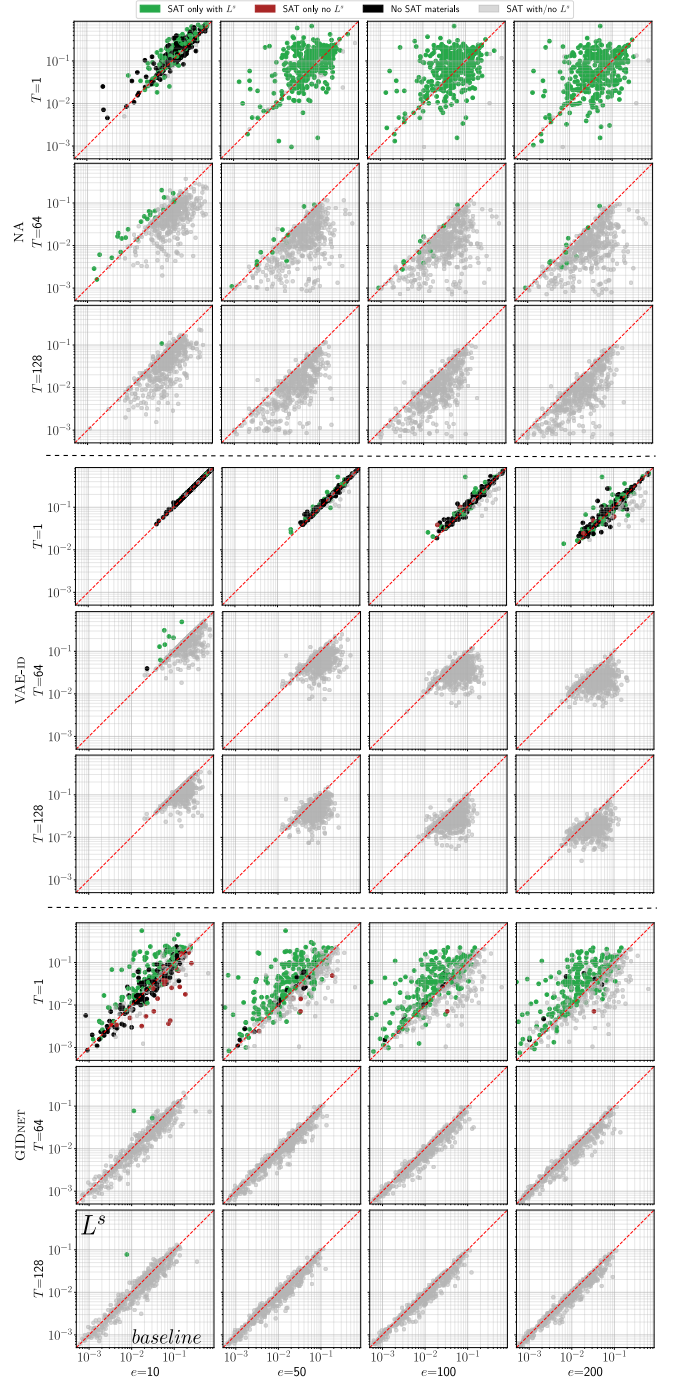


Figure 3: Results for the UA layout constraint on $\mathcal{D}_{\ell=5}$.

D Additional results

In the following, we show additional results considering a set of different constraints and different datasets with respect to the ones reported in the main paper.

Figures 2 and 3 replicate the scatterplot layout introduced in the main paper, comparing the SRMSE of materials found by the SL-augmented and baseline methods. The red diagonal marks equal performance on the two approaches.

The former figure shows results for the P4 layout constraint on $\mathcal{D}_{\ell=10}$. All solutions are obtained via SL-augmented optimization at the price of higher SRMSE compared to the baseline, which, nevertheless, produced invalid material. As materials satisfying the P4 layout constraint are absent from $\mathcal{D}_{\ell=10}$, this highlights the contribution of the Semantic Loss in scenarios where the constraint is not represented in the training data. Figure 3 shows results for the UA layout constraint on $\mathcal{D}_{\ell=5}$. In all the methods, as T and e increase, we can notice that both approaches lead to the discovery of valid materials. Indeed, such valid materials are already well represented in the training set (see Table 2 in the main paper), which contains a notable amount of material that satisfies the UA layout constraint. Thus, it is also probable for baseline methods to produce valid metamaterials. However, the Semantic Loss improves the exploration process, leading to materials with a lower SRMSE than their baseline counterpart, as we can observe from the mass of gray points below the bisector.

Figure 4 presents a series of histograms for the layout constraints PAL2, PAL3 and PAL4 on $\mathcal{D}_{\ell=10}$. In each plot, the number of valid materials found with Semantic Loss optimization is shown in blue, while the baseline (without Semantic Loss) appears in orange. Starting from the first constraint, PAL2, we can observe that the data reflects the previously observed results, where SL-augmented architectures are able to find valid materials in the early stages of exploration, whereas baseline models need more search time and starting points. We also notice how, on average, the mean of the distribution for the SL-augmented models is shifted to the left, towards lower SRMSE values compared to the baseline. When transitioning to the intermediate constraint, PAL3, the difference between the two approaches becomes more pronounced. The SL-augmented models still achieve high numbers of valid materials (also early in the process), in contrast, the baseline performance starts to drop as the design space narrows. The latter layout constraint, PAL4, is the one least represented in the original data (0% constraint satisfaction in both the training and test split of $\mathcal{D}_{\ell=10}$). Nonetheless, the SL-augmented architectures achieve excellent results while the baseline models struggle to find valid solutions. From this disparity, we can draw two conclusions. First, Semantic Loss drives the exploration process of the models towards regions of the design space that satisfy the target layout even when no such examples exist in the training data. Second, the gap between the number of valid materials found by the SL-augmented and baseline widens as the constraint becomes more stringent.

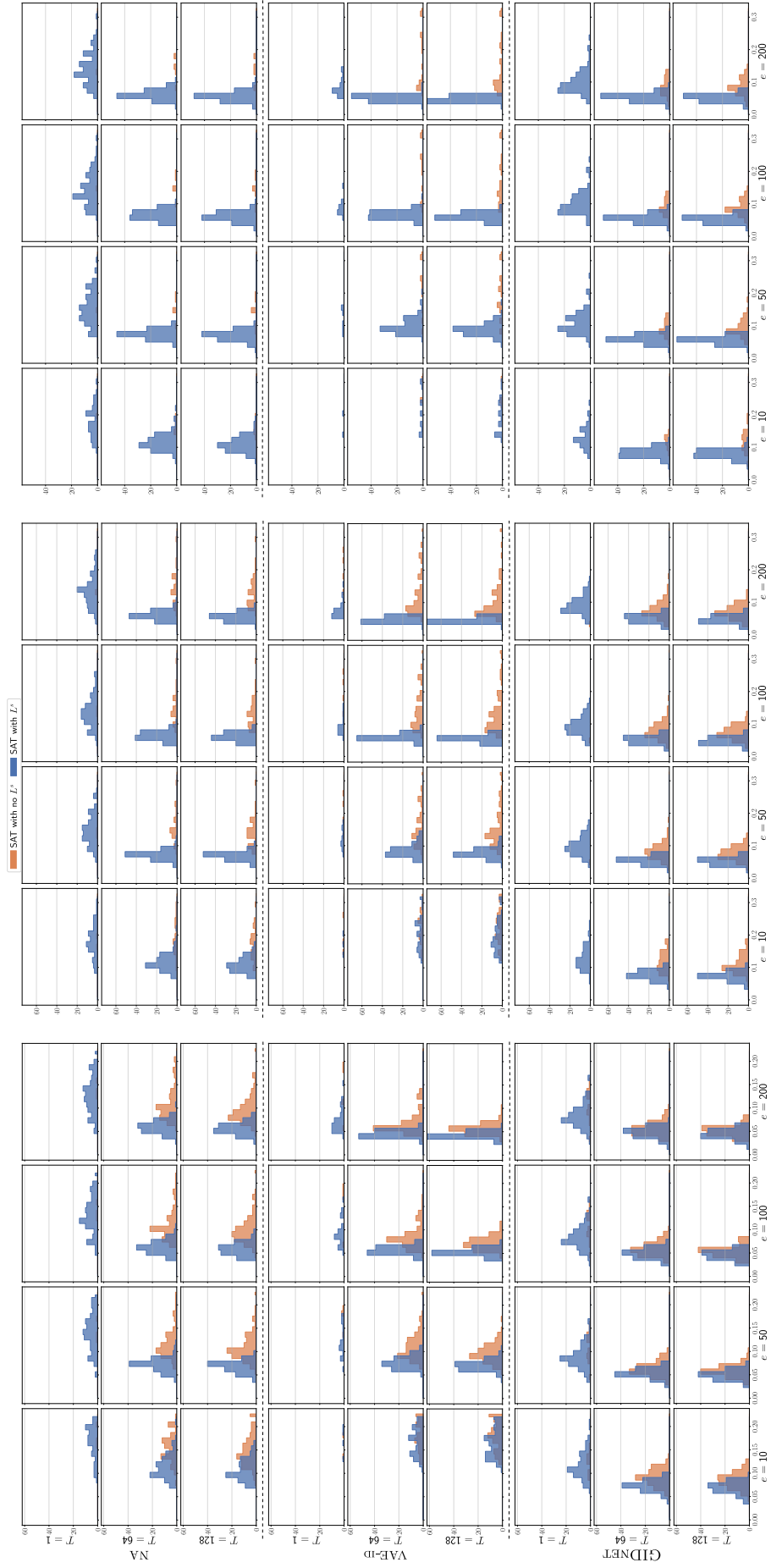


Figure 4: Side-by-side results for the PAL2, PAL3, and PAL4 layout constraints on $\mathcal{D}_{\ell=10}$.

References

- [1] A.H. Zaabab, Qi-Jun Zhang, and M. Nakhla. A neural network modeling approach to circuit optimization and statistical design. *IEEE Transactions on Microwave Theory and Techniques*, 43(6):1349–1358, 1995.
- [2] John Peurifoy, Yichen Shen, Li Jing, Yi Yang, Fidel Cano-Renteria, Brendan G. DeLacy, John D. Joannopoulos, Max Tegmark, and Marin Soljačić. Nanophotonic particle simulation and inverse design using artificial neural networks. *Science Advances*, 4(6):eaar4206, 2018.
- [3] Takashi Asano and Susumu Noda. Optimization of photonic crystal nanocavities based on deep learning. *Optics Express*, 26(25):32704–32717, 2018.
- [4] Jiaqi Jiang, Mingkun Chen, and Jonathan A. Fan. Deep neural networks for the evaluation and design of photonic devices. *Nature Reviews Materials*, 2020.
- [5] Simiao Ren, Willie Padilla, and Jordan Malof. Benchmarking deep inverse models over time, and the neural-adjoint method. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 38–48. Curran Associates, Inc., 2020.
- [6] Carlo Adornetto and Gianluigi Greco. Gidnets: Generative neural networks for solving inverse design problems via latent space exploration. In *IJCAI*, pages 3404–3413, 2023.
- [7] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018. PMID: 29532027.
- [8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [9] John Chilwell and Ian Hodgkinson. Thin-films field-transfer matrix theory of planar multilayer waveguides and reflection from prism-loaded waveguides. *Journal of the Optical Society of America, A*, 1(7):742–753, 1984.
- [10] Andrew Lininger, Michael Hinczewski, and Giuseppe Strangi. General inverse design of layered thin-film materials with convolutional neural networks. *ACS Photonics*, 8(12):3641–3650, 2021.
- [11] Jia-Qi Yang, Yucheng Xu, Jia-Lei Shen, Kebin Fan, De-Chuan Zhan, and Yang Yang. Idtoolkit: A toolkit for benchmarking and developing inverse design algorithms in nanophotonics. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2930–2940, 2023.