IDToolkit: A Toolkit for Benchmarking and Developing Inverse Design Algorithms in Nanophotonics Appendix

Jia-Qi Yang
yangjq@lamda.nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China

Yucheng Xu
Research Institute of Superconductor Electronics (RISE), School of Electronic Science and Engineering,
Nanjing University
Nanjing, China

Jia-Lei Shen
191300043@smail.nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China

Kebin Fan
kebin.fan@nju.edu.cn
Research Institute of Superconductor Electronics (RISE), School of Electronic Science and Engineering,
Nanjing University
Nanjing, China

De-Chuan Zhan*
zhandc@nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China

Yang Yang*
yyang@njust.edu.cn
Nanjing University of Science and Technology
Nanjing, China

ACM Reference Format:
Jia-Qi Yang, Yucheng Xu, Jia-Lei Shen, Kebin Fan, De-Chuan Zhan, and Yang Yang. 2023. IDToolkit: A Toolkit for Benchmarking and Developing Inverse Design Algorithms in Nanophotonics Appendix. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3580305.3599385

1 PHYSICAL UNDERSTANDING OF THE PROBLEMS

1.1 Multi-layer Optical Thin Films (MOTFs)

Multilayer optical thin films (MOTFs) are simple structures with many layers of optical films stacked together. They have been widely used in many optical applications to achieve specific optical properties, such as antireflective coating (ARC), distributed Bragg reflector (DBR), energy harvesting, and radiative cooling. The radiative cooling process normally involves the energy transfer from a hot body to a cold body via radiation process. As a daytime radiative cooler, it is required to radiate the heat through the transparency window in the range of 8-13 μm while to reflect most of the incident solar energy.[7, 26, 30] Its ideal absorption spectra are shown in the inset of Figure. 1. To optimize an MOTF cooler with ideal properties, we implement a simulator with 7 different dispersive materials. We set the layer number to 10, which provides a fairly large search space and enough flexibility (Raman et al. [23] proposes a 10 layer design). The layer number can be easily increased if necessary and less layer number can be obtained through setting some layer thicknesses to 0. It is noteworthy that such a broadband optimization is nontrivial since the materials are significantly dispersive from visible to long-wave infrared. As a result, the large parameter space could lead to onerous search with directly applying theoretical methods or numerical simulations. In addition, the MOTF cooler design problem has both discrete (material type) and continuous (layer thickness) variables, and the effective number of variables is not fixed. The design parameters of a multilayer film cooler is depicted in Figure. 1 (b). Specifically, assuming there k different layers, the design parameter $\mathbf{x}_0, ..., \mathbf{x}_{k-1}$ determine the the material used in the 1st to the kth layer respectively; the design parameter $\mathbf{x}_k, ..., \mathbf{x}_{2k-1}$ define the thicknesses of the 1st to kth layer, respectively. The material can be set to 7 different categorical values $\mathbf{x}_i \in \{ZnO, AlN, Al_2O_3, MgF_2, SiO_2, TiO_2, SiC\}, 0 < i < k$. The layer thicknesses can be set to $\mathbf{x}_i \in [0, 1], k \leq i < 2k$.

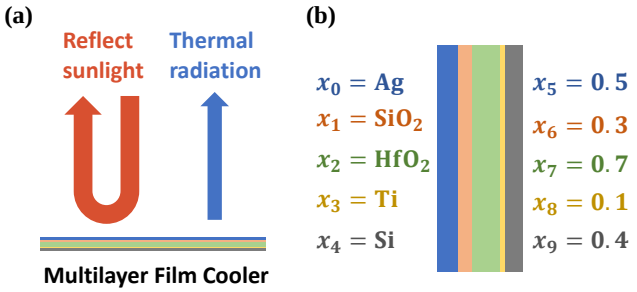

Figure 1: Depiction of MOTFs: (a) How a MOTFs works. (b) The design parameters of a MOTFs with 5 layers.

*Corresponding authors.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599385

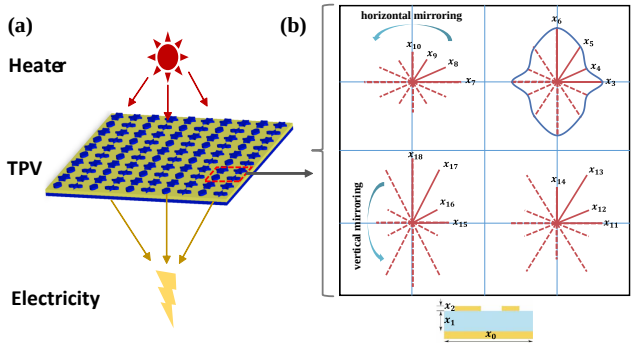

Figure 2: Depiction of the TPV problem: (a) How TPV energy conversion works. (b) The design parameters of TPV.

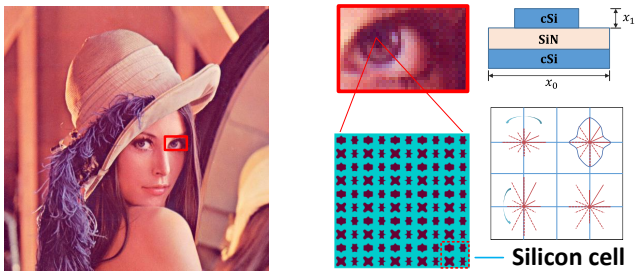

Figure 3: A schematic introduction of high-DPI printing with silicon colors. Each pixel is implemeted by a specific nanostructure cell, where the size of each cell is set to 150nm-350nm.

1.2 Thermophotovoltaics (TPV)

One of the promising applications of metamaterials is for energy harvesting using thermophotovoltaic (TPV) cells. These cells are designed to directly convert infrared energy into electricity with zero NO_x and CO_2 emission. For the TPV cells, it is essential to design a selective emitter with a spectral range overlapping with that of the external quantum efficiency (EQE) of the PV cell. In this benchmark test, we aim to design a selective emitter with the target absorptivity spectrum, which matches with the normalized EQE of GaInAsSb in the range of 1 to 4 μm.

We adopt a popular design scheme of metamaterial with the unit cell consisting of a bottom metal substrate and top plasmonic structures spaced by a dielectric layer, as depicted in Figure. 2 (a). Tungsten (W) was selected for both the bottom metallic layer and top structural layer. Alumina (Al_2O_3) is used as the dielectric spacer with the thickness $\mathbf{x}_1$ varying in the range 30 nm to 130 nm. The dielectric spacer is $\mathbf{x}_0$ corresponds the size of a cell, and $\mathbf{x}_2$ are the thickness of top metal layer. The bottom metal layer is thick enough to ensure not waves transmitting through. Considering the feasibility of fabricating the designed TPV, the cells should be single-connected and have smooth edges. To this end, we propose to utilize the B-Spline method to generate the shape of the cells. The control points of B-Spline are defined by the design parameters $\mathbf{x}_3$, ..., $\mathbf{x}_1 8$. To further increase the flexibility, we use 4 sub-cells in a single cell as depicted in Figure. 2 (b). Each sub-cell is controlled by 4 control points and is reflected vertically and horizontally to produce a symmetrical shape. We set $\mathbf{x}_0 \in [350, 500]$, $\mathbf{x}_2 \in [10, 80]$ and $\mathbf{x}_i \in [40, \frac{\mathbf{x}_0}{2}], 2 < i < 19$. The constraint on $\mathbf{x}_i, 2 < i < 19$ is because the radius must be smaller than $\frac{1}{2}$ of the cell size $\mathbf{x}_0$ so that the sub-cells will not overlap.

1.3 Structural color filter (SCF)

Structural color filtering (SCF) using plasmonic and dielectric metamaterials has enabled tremendous potential for coloring applications, including passive color display, information storage, Bayer filtering, etc. [9, 11, 17, 21, 29, 31] Comparing to the plasmonic metamamterials with limited coverage on International Commission on Illumination (CIE) 1931 chromaticity diagram, filters based on dielectric metasurfaces have shown much larger coverage and higher saturation. In this benchmark, we modeled the color filter using an arrayed silicon metasurface, whose supercell consists of four pillars sitting on a 70-nm SiN_3 thin film, which is supported by a silicon substrate. Since the thickness of the substrate is much larger than the operating wavelength in the visible range, the thickness of the silicon substrate was kept at 100 nm, and we implemented a perfectly matched layer in the bottom of the substrate to save the numerical simulation time. The cell size is controlled by the parameter $\mathbf{x}_0$, and the height of the four pillars are the same and are varied as the parameter of $\mathbf{x}_1$. The shapes of the four silicon pillars are independently generated with parameters $\mathbf{x}_2$, ..., $\mathbf{x}_{17}$, which are similar to the TPV problem. To obtain a structural color filter, the first step is to design a color palette where each color is implemented with a specific nanostructure. In this step, the reflectance spectra from the nanostructures are obtained from numerical simulation. Then three parameters x, y, and Y, which represent the values shown in the CIE chromaticity diagram, can be calculated using the method detailed in reference [11] . The inverse design problem is to find nanostructures that match with the colors that are to be printed.

The inverse design targets of all three problems are described in the appendix because of the page limit.

2 ARCHITECTURE OF THE IDTOOLKIT

The overall design of the IDToolkit is depicted in Figure. 4. The IDToolkit consists of a benchmark library and a separate experiment code base to reproduce the experiments presented in this paper. The benchmark library consists of inverse design algorithms, environments (inverse design problems), and a parameters space manager.

Algorithms. The Algorithm class is designed to be scalable and extendable: New algorithms can be implemented with the same interface; Large search algorithms can be deployed on computing clusters with the ray library[18]. We implement the interactive optimization algorithms with a wrapper of the ray tune library, and the deep inverse design methods are implemented with the pytorch lightning framework, which enables high-performance training on GPU clusters and is easy to extend. For example, we use MLP networks in all the deep inverse design methods, the network architectures can be easily modified to high-performance alternatives such as CNN. The details of the Algorithm class interface is documented in the comments of the source code.

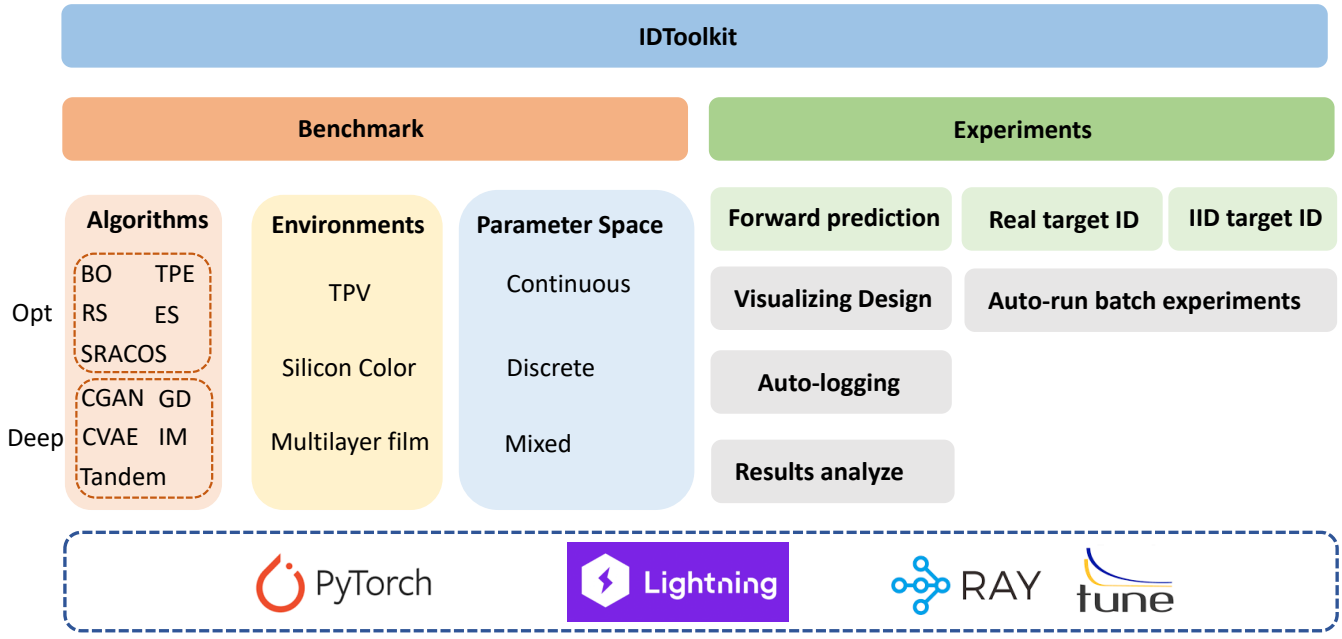

Figure 4: Architecture of the IDToolkit.

Environments. The environments denote the benchmark problems, which is similar to the reinforcement environments in the OpenAI gym[5]. Each environment can receive a batch of design parameters and return a batch of performance scores accordingly. The design parameters are fed into a physical simulation model implemented with meep[1], and the responses ($\mathbf{y}$) are outputted. The responses are compared with the design parameters to calculate the performance score. The calculation of the scores is parallelized with multiprocessing in python to accelerate computation.

Parameter space. Each environment has a parameter space, which has the following functions: (1) Define the feasible range of each design parameter, e.g., the lowest and highest value of continuous variables and available choices of categorical variables. (2) Perform constraints on the design parameters, e.g., if the constraint is $\mathbf{x}_0 > \mathbf{x}_2$, the value of $\mathbf{x}_2$ is reset to $\min(\mathbf{x}_2, \mathbf{x}_1)$. (3) Perform checking of the design parameters. Typically, the simulator (meep) will not check the design parameters, so even if the design is not feasible, the simulator may still return a result without any error. Thus, if the design parameters are not feasible and cannot be fixed, the parameter space will raise an error and end the simulation. (4) Define the random sampling distribution. This random distribution is used to generate the training dataset, to be used as a baseline of interactive optimization methods, to generate the testing datasets of inverse design of IID targets. Currently, we implement uniform sampling of continuous and discrete variables. The sampling distribution can be any distribution of the feasible design parameter space, which is also an important factor that will affect the inverse design performance. (5) Combine multiple continuous and discrete design parameters as a whole parameter space.

Experiments. The experiments code base provides all the necessary contents to reproduce the experimental results in the paper. We also provide several useful functions to conduct further experiments or algorithm design. 1. We provide visualization function of the design of TPV and silicon color problems, which may help gain a better understanding of the problem. 2. We provide an experiment manager function, which supports running a batch of hyper-parameters, auto-logging of inverse design results, and analysis of the inverse design results.

3 THE DESIGN TARGETS

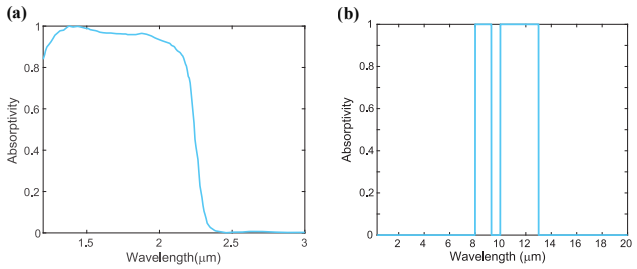

Figure 5: Spectrum design targets of MOTFs and TPV: (a) TPV design target. (b) MOTFs design target.

The spectrum of the design target of TPV problem is depicted in Figure. 5 (a), and the spectrum of the design target of MOTFs problem is illustrated in Figure. 5 (b). The exact value of the targets are in the `assets/tpv_model/tpv_target.txt` and `assets/multi_layer_model/target.txt` file in the benchmark code folder, respectively.

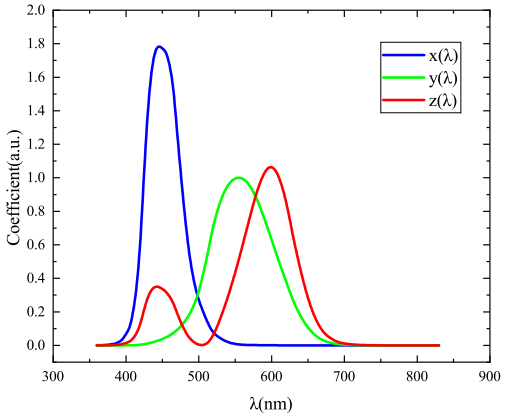

Figure 6: Color matching functions $x(\lambda)$, $y(\lambda)$, and $z(\lambda)$ for the calculation of International Commission on Illumination(CIE) chromaticity diagram, which represents "Blue", "Green", and "Red" colors.

3.1 SCF

Due to the fact that electromagnetic simulation could only provide the reflectance spectrum of the model, it is necessary to convert the simulated spectra from the color pixels to their CIE coordinates. The tristimulus X, Y, and Z values depend on the view of the observer should be obtained by calculating their overlap integral with the color-matching functions, while due to limited spectrum data points, the integration is approximated as the sum of data points:

$$\begin{aligned} X(\lambda) &= \frac{\alpha}{K}\sum_{i=1}^{n} I(\lambda_i)\, R(\lambda_i)\, x(\lambda_i)\, \Delta\lambda \\ Y(\lambda) &= \frac{\alpha}{K}\sum_{i=1}^{n} I(\lambda_i)\, R(\lambda_i)\, y(\lambda_i)\, \Delta\lambda \\ Z(\lambda) &= \frac{\alpha}{K}\sum_{i=1}^{n} I(\lambda_i)\, R(\lambda_i)\, z(\lambda_i)\, \Delta\lambda \end{aligned} \tag{1}$$

Where K is defined as follows:

$$K = \sum_{i=1}^{n} I(\lambda_i)\, y(\lambda_i) \tag{2}$$

Here $R(\lambda_i)$ means the reflectance spectrum. α is a scaling factor whose magnitude is chosen as 1 and $I(\lambda)$ is the spectral power distribution of the illuminant. The color matching functions are depicted in Figure. 6. Then the certain color could be obtained by $\tilde{x}$ and $\tilde{y}$ value:

$$\begin{aligned} \tilde{x} &= \frac{X(\lambda)}{X(\lambda)+Y(\lambda)+Z(\lambda)} \\ \tilde{y} &= \frac{Y(\lambda)}{X(\lambda)+Y(\lambda)+Z(\lambda)} \end{aligned} \tag{3}$$

The specific coordinate $(\tilde{x}, \tilde{y})$ could define a CIE chromatically diagram, and also represents a certain color. Since the human eye is most sensitive to the stimulation of green light, $Y(\lambda)$ is also adopted to measure the brightness of the color. Therefore the three parameters together form the CIE xyY color space. In our work, the coordinate $(\tilde{x}, \tilde{y}, Y(\lambda))$ is the target of inverse design. We choose 5 representative colors as our design target, as depicted in Figure. 7.

Figure 7: The design targets of the SCF problem.

4 VERIFY THE MEEP IMPLEMENTATION

Since we are re-implementing the nanophotonic devices with the open-source MEEP[1] software, we would like to verify our implementation is correct and aligns with the results generated from commercial software such as CST. We compare the results of our CST implementation and MEEP implementation in this section.

4.1 Verify TPV and SCF

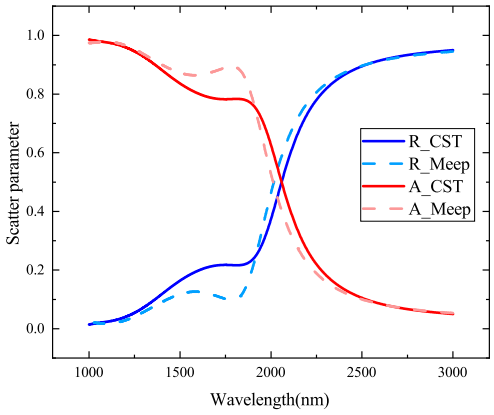

Figure 8: Simulation comparison between CST Studio Suite and Meep, where "R" and "A" represents reflectance and absorption of the simulated TPV model.

In order to ensure the accuracy of electromagnetic simulation results calculated by Meep, the comparison of simulation results of CST Studio Suite and Meep is as shown in Figure. 8. Here TPV model is selected to be simulated to make the comparison of reflectivity and absorption rate under certain parameter settings. The simulation results given by the two simulation software are close in general. Difference between the two simulation results in partial frequency bands is mainly due to inconsistent boundary condition settings and different methods of mesh generation of the model between CST Studio Suite and Meep. In addition, inconsistent interpolation methods for the dielectric constant of dispersive materials

in different simulation software also has a slight impact on the overall simulation results. For both TPV model and SCF model, several times of simulation comparisons have been done to ensure the accuracy of the simulation results derived from Meep.

4.2 Verify MOTFs

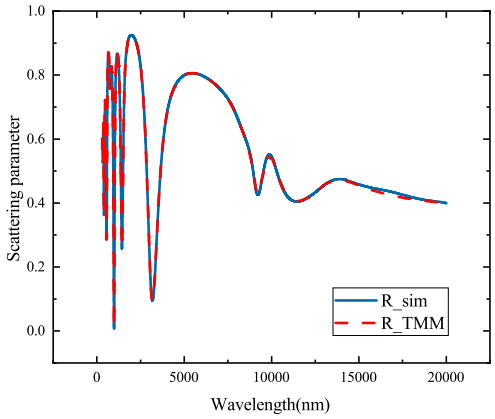

Figure 9: Comparison between TMM and electromagnetic simulation, where "R" represents reflectance of the multi-layer model.

To verify whether model results achieved by TMM and electromagnetic simulation are consistent in calculating the reflectance of the multi-layer model, here a four-layer model (SiC-TiO2-SiO2-SiC, and thickness of which are 55, 200, 100, 240 nanometers, respectively) is selected, and corresponding results are shown in Figure. 9. Transverse electric (TE) wave adopted as the wave source is incident at an angle of 60° to the normal. It can be seen that the two calculation results are almost identical.

5 REPRODUCTION

We provide code to reproduce all the results in the paper at https://anonymous.4open.science/r/IDToolkit/, which is fully anonymized for review. Due to the large size, we did not upload the datasets during reviewing, and full datasets will be published once accepted. Small sample datasets can be generated with the script `gen_data.py` if necessary. We briefly summarize the configurations of the methods and experiments here. We refer the readers to the comments in the code for more details.

5.1 Inverse Design

- Random search (**RS**): Uniform sampling distribution is used for the three problems.
- Sequential randomized coordinate shrinking (**SRACOS**)[15]: We use the asynchronous SRACOS (ASRACOS) implementation to enable parallel search for multiple design parameters. The number of parallels is set to 8.
- Bayesian Optimization (**BO**)[27]: We use the default configuration in the BayesOpt package. The Bayesian optimization does not support discrete parameters. To apply BO in the MOTFs problem, we convert the categorical values to integers before feeding them to BO, then convert the design parameters produced by BO to categorical values. Specifically, the 7 materials are mapped to $[0, 7]$, $[0, 1)$ corresponds to the first material, $[1, 2)$ corresponds to the second material, and so on.
- Tree-structured Parzen Estimator Approach (**TPE**) [3]: We use the default hyper-parameters in the HyperOpt package [4].
- Evolution Strategy (**ES**)[2]: We use the default configuration in the NeverGrad package [24].
- Inverse Model (**IM**): We use an MLP with 6 layers and ReLU activation. The hidden layer sizes of all the deep learning methods are set to 256, batch size is set to 128, learning rate is set to 1e-3. The networks are trained for 100 epochs.
- Gradient Descent (**GD**): The hyper-parameters of the surrogate forward model are the same as IM. We randomly select 1000 design parameters as the start points. Then the design parameters are optimized with gradient descent on the loss function. The initial learning rate is 1e-1, and the learning rate is divided by 2 if the loss function does not decrease for 30 steps. We use the Adam optimizer[16] for 500 steps of gradient descent. Following [25], we drop the first 10% optimized parameters to alleviate overfitting.
- **Tandem**[19]: The hyper-parameters of the surrogate forward model and the inverse model are the same as IM.
- Conditional Generative Adversarial Networks (**CGAN**)[13, 22]: The dimension of the latent variable z is set to 128. Since the validation loss of GANs can not reflect generalization performance, we train the CGAN method for 100 epochs and use the latest model without validation.
- Conditional Variational Auto-Encoder (**CVAE**) [28]: The dimension of the latent variable z is set to 128. The weight on the KL divergence loss is set to 1, which is also the β parameter in the β–VAE[14].

5.2 Forward Models

- Linear Regression (**LR**): We use the default parameters in sklearn[6].
- Decision Tree (**DT**). We use CART[20] implementation in sklearn[6] which is a variant of DT. We use the default parameters in sklearn[6].
- Gradient-Boosted Decision Tree (**GBDT**): We use the default parameters in XgBoost[8].
- Multilayer perceptron (**MLP**). We use a 6 layer MLP with ReLU activations. The learning rate is set to 1e-3, batch size is 128. The model is trained for 100 epochs, and the version with the best validation loss is used.
- Convolutional neural networks (**CNNs**): We use 6 1-dimensional convolution layers to learn the embeddings, each with kernel size=5, stride=3, pad=1. Then we use a transposed convolution layer to up-sample the outputs to the desired dimensions.

6 VISUALIZATION OF INVERSE DESIGN RESULTS

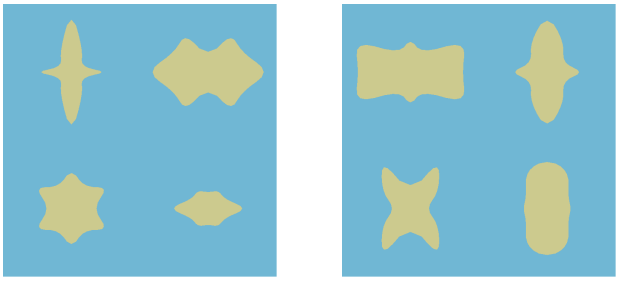

Figure 10: Visualization of design parameters of SCF problem, produced by Bayesian Optimization (BO) without training data. The MSE are 0.0068 and 0.0078, respectively.

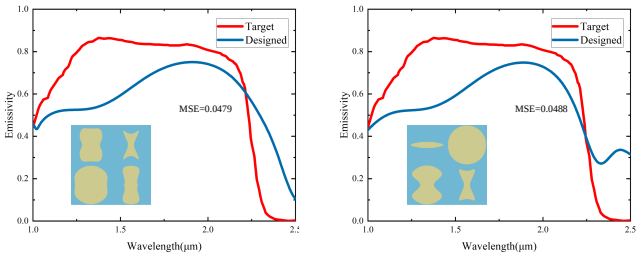

Figure 11: Visualization of design parameters of TPV problem, produced by Gradient Descent (GD) trained with the full dataset.

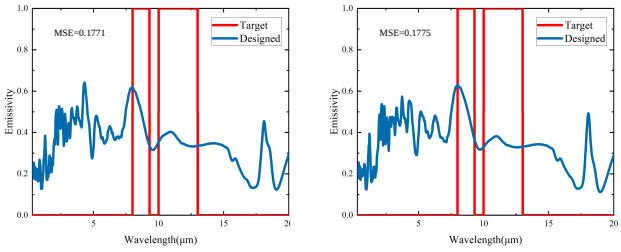

Figure 12: Visualization of design parameters of MOTFs problem, produced by TPE without training data.

We visualize some design parameters obtained by the inverse design algorithms in Figure. 10, Figure. 11, and Figure. 12. It's noteworthy that although some of the algorithms outperform the random search baseline by a large margin, the performance is still far from satisfactory, considering the requirements of real-world applications. There are two reasons: First, the tried number of designs is limited in our experiments to accelerate evaluation, which may not achieve the best results found in the literature. Our comparison is fair between different methods, and the limited trials are enough to show the trends. Secondly, our benchmark problems are selected from recent literature. Thus, these inverse design problems are still in active development and have much room for further improvement.

6.1 Forward Prediction Task

Our dataset can also be used to benchmark forward prediction methods. In our benchmark, we implemented 5 different methods.

- Linear Regression (**LR**): A simple baseline of regression tasks.
- Decision Tree (**DT**). Decision trees are tree-structured learning algorithms that split data samples into different branches with specific splitting methods layer by layer.
- Gradient-Boosted Decision Tree (**GBDT**): A widely used tree-based boosting learning method which has state-of-the-art performance on various applications with discrete values and small datasets.
- Multilayer perceptron (**MLP**). For prediction problems without specific structures, MLP may be a strong baseline, which consists of multiple linear layers with non-linear activation functions[12].
- Convolutional neural networks (**CNNs**): CNNs perform better for data with spacial structures such as image and audio. Here, we implement transposed convolution [10] as the output layer, which may perform well for one-dimensional structured outputs (spectrums).

The results are depicted in Figure. 13. First, GBDT is the best-performing method in the SCF problem, which may be because the output dimension is small, and GBDT performs better with simpler targets. Secondly, CNN performs best in the MOTFs and TPV problems, which indicates the advantage of CNNs on structured spatial data (the y of multilayer cooler and TPV are spectrums that are typically smooth locally).

7 COMPARISON BETWEEN ML-BASED METHODS AND ADJOINT METHOD

We think the primary potentials of machine learning based methods are:

Fast design speed after training: Deep models such as VAE and GAN can generate 1000 different designs (for 1000 different targets) within 10 seconds. And their training time is less than 1 hour. For a fair comparison, we implemented the Adjoint method with the MEEP framework and tested on the SCF problem. The Adjoint method needs more than 20 hours to converge with a single target. (for reference, the official example provided by MEEP takes 14 hours to converge on a much simpler 2D problem, more details in https://meep.readthedocs.io/en/latest/Python_Tutorials/Adjoint_Solver/#adjoint-solver) Since we typically have thousands of different targets to design in the SCF problem, the speed advantage of ML methods is considerable (more than 7200*1000 times faster in SCF task). The comparison is conducted with the same hardware to ensure fairness.

Transferability among different targets and tasks: We must run the Adjoint method on every different task (note that using a different material or configuration also means a different task) and target. On the contrary, the transferability of large ML models on various tasks has been proven in recent years. For example, GPT

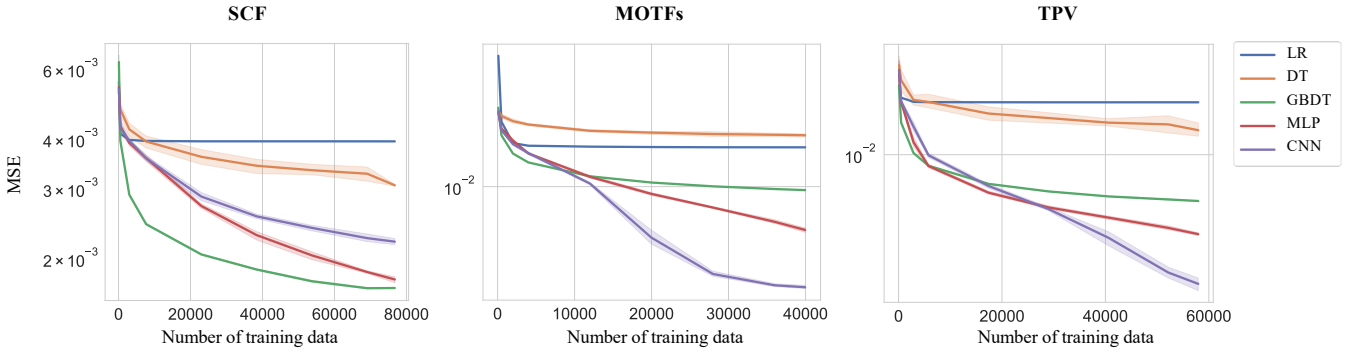

Figure 13: Benchmark on the forward prediction tasks. We experiment with different training dataset sizes.

can achieve state-of-the-art performance in many different NLP tasks. If such a universal model can be developed for inverse design, the speed advantage compared with the Adjoint method will be more significant.

REFERENCES

[1] 2010. Meep: A flexible free-software package for electromagnetic simulations by the FDTD method. *Computer Physics Communications* 181, 3 (2010), 687–702. https://doi.org/10.1016/j.cpc.2009.11.008

[2] Anne Auger. 2009. Benchmarking the (1+1) evolution strategy with one-fifth success rule on the BBOB-2009 function testbed. In *GECCO*, Franz Rothlauf (Ed.). ACM, 2447–2452. https://doi.org/10.1145/1570256.1570342

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *NIPS*, John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger (Eds.). 2546–2554.

[4] James Bergstra, Dan Yamins, David D Cox, et al. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, Vol. 13. Citeseer, 20.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016).

[6] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[7] Dongwoo Chae, Mingeon Kim, Pil-Hoon Jung, Soomin Son, Junyong Seo, Yuting Liu, Bong Jae Lee, and Heon Lee. 2020. Spectrally Selective Inorganic-Based Multilayer Emitter for Daytime Radiative Cooling. *ACS Applied Materials & Interfaces* 12, 7 (Jan. 2020), 8073–8081. https://doi.org/10.1021/acsami.9b16742

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *KDD* (San Francisco, California, USA). 785–794. https://doi.org/10.1145/2939672.2939785

[9] Xiaoyang Duan, Simon Kamin, and Na Liu. 2017. Dynamic plasmonic colour display. *Nature Communications* 8, 1 (Feb. 2017). https://doi.org/10.1038/ncomms14606

[10] Vincent Dumoulin and Francesco Visin. 2016. A guide to convolution arithmetic for deep learning. *CoRR* abs/1603.07285 (2016). arXiv:1603.07285

[11] Li Gao, Xiaozhong Li, Dianjing Liu, Lianhui Wang, and Zongfu Yu. 2019. A Bidirectional Deep Neural Network for Accurate Silicon Color Design. *Advanced Materials* 31, 51 (Nov. 2019), 1905467. https://doi.org/10.1002/adma.201905467

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *CoRR* abs/1406.2661 (2014). arXiv:1406.2661

[14] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*. OpenReview.net.

[15] Yi-Qi Hu, Hong Qian, and Yang Yu. 2017. Sequential Classification-Based Optimization for Direct Policy Search. In *AAAI*, Satinder Singh and Shaul Markovitch (Eds.). AAAI Press, 2029–2035.

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] Karthik Kumar, Huigao Duan, Ravi S. Hegde, Samuel C. W. Koh, Jennifer N. Wei, and Joel K. W. Yang. 2012. Printing colour at the optical diffraction limit. *Nature Nanotechnology* 7, 9 (Aug. 2012), 557–561. https://doi.org/10.1038/nnano.2012.128

[18] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. 2018. Tune: A Research Platform for Distributed Model Selection and Training. *CoRR* abs/1807.05118 (2018). arXiv:1807.05118

[19] 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. https://doi.org/10.1021/acsphotonics.7b01377

[20] Wei-Yin Loh. 2011. Classification and regression trees. *WIREs Data Mining Knowl. Discov.* 1, 1 (2011), 14–23. https://doi.org/10.1002/widm.8

[21] Taigao Ma, , Mustafa Tobah, Haozhu Wang, L. Jay Guo, and and. 2022. Benchmarking deep learning-based models on nanophotonic inverse design problems. *Opto-Electronic Science* 1, 1 (2022), 210012–210012. https://doi.org/10.29026/oes.2022.210012

[22] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014). arXiv:1411.1784

[23] Aaswath P. Raman, Marc Abou Anoma, Linxiao Zhu, Eden Rephaeli, and Shanhui Fan. 2014. Passive radiative cooling below ambient air temperature under direct sunlight. *Nature* 515, 7528 (Nov. 2014), 540–544. https://doi.org/10.1038/nature13883

[24] J. Rapin and O. Teytaud. 2018. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad.

[25] Simiao Ren, Willie Padilla, and Jordan M. Malof. 2020. Benchmarking Deep Inverse Models over time, and the Neural-Adjoint method. In *NeurIPS*.

[26] Yu Shi, Wei Li, Aaswath Raman, and Shanhui Fan. 2017. Optimization of Multilayer Optical Films with a Memetic Algorithm and Mixed Integer Programming. *ACS Photonics* 5, 3 (Dec. 2017), 684–691. https://doi.org/10.1021/acsphotonics.7b01136

[27] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *NIPS*. 2960–2968.

[28] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *NIPS*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 3483–3491.

[29] Maowen Song, Lei Feng, Pengcheng Huo, Mingze Liu, Chunyu Huang, Feng Yan, Yan qing Lu, and Ting Xu. 2022. Versatile full-colour nanopainting enabled by a pixelated plasmonic metasurface. *Nature Nanotechnology* (Dec. 2022). https://doi.org/10.1038/s41565-022-01256-4

[30] Haozhu Wang, Zeyu Zheng, Chengang Ji, and L Jay Guo. 2021. Automated multilayer optical design via deep reinforcement learning. *Machine Learning: Science and Technology* 2, 2 (Feb. 2021), 025013. https://doi.org/10.1088/2632-2153/abc327

[31] Xiujuan Zou, Youming Zhang, Ruoyu Lin, Guangxing Gong, Shuming Wang, Shining Zhu, and Zhenlin Wang. 2022. Pixel-level Bayer-type colour router based on metasurfaces. *Nature Communications* 13, 1 (June 2022). https://doi.org/10.1038/s41467-022-31019-7