INVERSE DESIGN OF GRATING COUPLERS USING THE POLICY GRADIENT METHOD FROM REINFORCEMENT LEARNING

SUPPLEMENTARY MATERIAL

Sean Hooten[*]
Hewlett Packard Labs
Hewlett Packard Enterprise
Milpitas, CA 95035, USA
sean.hooten@hpe.com

Raymond G. Beausoleil
Hewlett Packard Labs
Hewlett Packard Enterprise
Milpitas, CA 95035, USA
ray.beausoleil@hpe.com

Thomas Van Vaerenbergh
Hewlett Packard Labs
HPE Belgium
B-1831 Diegem, Belgium
thomas.van-vaerenbergh@hpe.com

1 Theory of GLOnet and PHORCED Optimization Algorithms

Below we will explain and derive the formulae presented in Fig. 1 of the main manuscript, particularly the gradient/backpropagation terms. The primary intention of this section is to justify the claim that PHORCED does not require an adjoint simulation. This section will assume the reader has a basic understanding of gradient-based optimization and the adjoint method. The reader is recommended to consult Refs. [1–5] for more detailed explanation of these topics.

In the following, let $\mathbf{p} = [p_1, p_2, ..., p_{n-1}, p_n]^T$ be a vector of n geometrical degrees-of-freedom (parameters) that we wish to vary in our device. Furthermore, the time-harmonic Maxwell's Equations may be written as a matrix equation:

$$\text{Maxwell's Equations: } \mathbf{A}\mathbf{x} = \mathbf{b} \tag{S.1}$$

with:

$$\mathbf{A} = \begin{bmatrix} j\omega\varepsilon(\mathbf{p}) & \nabla\times \\ \nabla\times & -j\omega\mu(\mathbf{p}) \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{E} \\ \mathbf{H} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{J}_e \\ \mathbf{J}_m \end{bmatrix} \tag{S.2}$$

where j is the imaginary unit, ω is the frequency, $\mathbf{A}$ is the Maxwell Operator, $\mathbf{x}$ is a vector of electric and magnetic fields $\mathbf{E}$ and $\mathbf{H}$, and $\mathbf{b}$ is a vector of electric and magnetic current sources $\mathbf{J}_e$ and $\mathbf{J}_m$. Notice that $\mathbf{A}$ includes the permittivity and permeability ε and μ which are explicitly parameterized by the geometrical degrees-of-freedom $\mathbf{p}$. We will assume that the electric and magnetic field vectors each have dimension k, which might be the total number of Yee cells in an FDTD simulation. Note also that the electric and magnetic fields are, in general, complex-valued.

A solution to Maxwell's Equations (e.g. a simulation), can be regarded as solving Eq. (S.1) for $\mathbf{x}$ given $\mathbf{A}$ and $\mathbf{b}$. A shorthand way to write this is:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{S.3}$$

Though, it should be emphasized that taking an explicit inverse of $\mathbf{A}$ is rarely done in practice (and not to mention, not well-defined on a continuous domain). Since $\mathbf{A}$ is a function of $\mathbf{p}$, we will rewrite Eq. (S.3) in a more convenient form. Without loss of generality, let $g : \mathcal{R}^n \to \mathcal{C}^{2k}$ be a function that maps geometrical degrees-of-freedom to electric and magnetic field quantities. Then, we can write:

$$\boxed{\text{Forward Simulation: } \mathbf{x} = g(\mathbf{p})} \tag{S.4}$$

which represents the operation given above in Eq. (S.3), but with explicit dependence on $\mathbf{p}$. Please note that g is specific to a particular electromagnetic solution of Maxwell's equations with implicitly-defined boundary conditions and geometrical mapping of the permittivity and permeability; one should always refer back to Eq. (S.1) for the most general solution to Maxwell's Equations.

[*]Also affiliated with Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA 94709, USA.

The last requirement for a well-defined inverse design optimization problem is a merit function, which defines the performance or success of a given electromagnetic device. Let $f : \mathcal{C}^{2k} \to \mathcal{R}$ be a function that maps electric and magnetic field quantities to a performance metric (figure-of-merit). Then, given $\mathbf{x} = g(\mathbf{p})$ for given parameters $\mathbf{p}$ we may write:

$$\text{Figure-of-Merit} = f(\mathbf{x}) \tag{S.5}$$

$$\text{Figure-of-Merit} = (f \circ g)(\mathbf{p}) \tag{S.6}$$

where in Eq. (S.6) we made use of Eq. (S.4) to write the total figure-of-merit as a function composition of the electromagnetic merit function and the solution to Maxwell's Equations given well-defined $\mathbf{p}$. The intention of an inverse design optimization is to finding the best set of geometrical degrees-of-freedom that improve the figure-of-merit. Assuming that we intend to maximize the figure-of-merit, we may write a generalized electromagnetic inverse design problem in optimization notation as the following:

$$\boxed{\text{Inverse Design Optimization: } \mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmax}} \, (f \circ g)(\mathbf{p})} \tag{S.7}$$

where $\mathbf{p}^*$ is the global optimum of $(f \circ g)$. Using optimization methods such as gradient-descent, GLOnet, and PHORCED, we hope to attain $\mathbf{p}^*$ in a computationally efficient way.

Given the convenient notation in Eqs. (S.4)-(S.7)we are now prepared to explain the three optimization algorithms given in Fig. 1 of the main text. We begin with simple gradient-based optimization.

1.1 Gradient-Based Optimization

In gradient-based optimization we intend to use the gradient of a function to inform an optimization update step. In terms of our inverse design optimization problem Eq. (S.7), we take the gradient of the figure-of-merit with respect to the geometrical variable $\mathbf{p}$ defined at some given $\mathbf{p}'$. For convenience we write this gradient in a slightly shorthand way:

$$\text{Figure-of-Merit Gradient: } \frac{\partial (f \circ g)}{\partial \mathbf{p}'} \equiv \nabla_{\mathbf{p}} \left[(f \circ g)(\mathbf{p}) \right]_{\mathbf{p}=\mathbf{p}'} \tag{S.8}$$

In electromagnetic problems, the gradient is most efficiently calculated using the adjoint method. An exact derivation of the adjoint method may be found in Refs. [1–5]. For our purposes, we require only the qualitative understanding that for any given $\mathbf{p}'$, we may solve Eq. (S.8) using just 2 simulations (the forward and adjoint simulations respectively), regardless of the dimension of the parameter vector, $n = \dim(\mathbf{p}')$. Using this gradient, we can then update the parameter vector that will be used in the subsequent iteration of our gradient-based optimization algorithm. In this work we used the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm for our baseline gradient-based optimization comparisons. This algorithm was implemented out-of-the-box from the scipy.optimize python module, where forward/adjoint simulations and gradient calculations were performed in the open-source electromagnetic optimization package EMopt [6].

1.2 GLOnet

In this section we will derive the backpropagation term from Fig. 1 of the main manuscript. Note that this derivation differs from that originally given by Jiang and Fan [7, 8], we refer the reader there for more detail and a for a different perspective from that discussed here. Let $\mathbf{z} \sim \mathcal{D}$ be a random vector drawn from a (d-dimensional) distribution $\mathcal{D}$. Importantly, this noise vector serves as an input for a neural network that deterministically generates geometrical parameter vectors for simulation, based on programmable weights denoted by θ. Specifically, let $h_{\boldsymbol{\theta}} : \mathcal{R}^d \to \mathcal{R}^n$ represent this function such that for given $\boldsymbol{\theta}$ and sampled $\mathbf{z} \sim \mathcal{D}$ we have:

$$\mathbf{p} = h_{\boldsymbol{\theta}}(\mathbf{z}) \tag{S.9}$$

By virtue of $\mathbf{z}$ being a random variable, $\mathbf{p}$ is also random but drawn from an unknown distribution parameterized by $\boldsymbol{\theta}$. Consequently, the objective function for inverse design optimization via GLOnet becomes:

$$(f \circ g)(\mathbf{p}) \to \underset{\mathbf{p}=h_{\boldsymbol{\theta}}(\mathbf{z})}{\mathbb{E}} [(f \circ g)(\mathbf{p})] = \underset{\mathbf{z}\sim\mathcal{D}}{\mathbb{E}} [(f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z})] \tag{S.10}$$

where $\mathbb{E}[\cdot]$ is the expected value operator. In the right-hand side we simplified the expression by replacing $\mathbf{p} = h_{\boldsymbol{\theta}}(\mathbf{z})$. Intuitively our intention is to optimize the average value of the electromagnetic merit function generated by the neural network. Note that because of this exchange, the variable(s) that may be explicitly controlled by the user or optimization algorithm are the programmable weights, $\boldsymbol{\theta}$. Consequently, the optimization problem for GLOnet may be written:

$$\boxed{\text{GLOnet Optimization: } \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \underset{\mathbf{z}}{\mathbb{E}} [(f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z})]} \tag{S.11}$$

where we wrote $\mathbf{z}$ as the variable of the expected value with the understanding that it is sampled from user-defined distribution $\mathcal{D}$. To optimize $\boldsymbol{\theta}$ for this objective, we will leverage the powerful neural network optimization package called PyTorch [9]. This tool allows one to invoke state-of-the-art gradient-based algorithms for the optimization of neural network weights. This will require the partial derivatives of the GLOnet objective with respect to the neural networks weights.

Let the probability density function of $\mathbf{z}$ be denoted $\text{pdf}(\mathbf{z})$, then we may write:

$$\mathop{\mathbb{E}}_{\mathbf{z}}[(f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z})] = \int (f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z}) \text{pdf}(\mathbf{z}) d\mathbf{z} \tag{S.12}$$

Then for scalar weight $\theta_j \in \boldsymbol{\theta}$ in our neural network, the partial derivative of the objective is given by:

$$\frac{\partial}{\partial \theta_j} \mathop{\mathbb{E}}_{\mathbf{z}}[(f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z})] = \int \frac{\partial (f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z})}{\partial \theta_j} \text{pdf}(\mathbf{z}) d\mathbf{z} \tag{S.13}$$

where we invoked the Leibniz integral rule for the exchange of the integral and partial derivative operators. Let $\mathbf{p} = h_{\boldsymbol{\theta}}(\mathbf{z})$ be the output of the neural network for input $\mathbf{z}$. Then we may apply the chain rule to the derivative term:

$$\frac{\partial}{\partial \theta_j} \mathop{\mathbb{E}}_{\mathbf{z}}[(f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z})] = \int \frac{\partial (f \circ g)}{\partial \mathbf{p}} \cdot \frac{\partial \mathbf{p}}{\partial \theta_j} \text{pdf}(\mathbf{z}) d\mathbf{z} \tag{S.14}$$

$$= \mathop{\mathbb{E}}_{\mathbf{z}} \left[\frac{\partial (f \circ g)}{\partial \mathbf{p}} \cdot \frac{\partial \mathbf{p}}{\partial \theta_j} \bigg|_{\mathbf{p}=h_{\boldsymbol{\theta}}(\mathbf{z})} \right] \tag{S.15}$$

where in the second step we re-wrote the integral as an expected value. Note that the "$\cdot$" operator is the vector dot product. Abusing notation slightly, we may write the full gradient as:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathop{\mathbb{E}}_{\mathbf{z}}[(f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z})] = \mathop{\mathbb{E}}_{\mathbf{z}} \left[\frac{\partial (f \circ g)}{\partial \mathbf{p}} \cdot \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}} \bigg|_{\mathbf{p}=h_{\boldsymbol{\theta}}(\mathbf{z})} \right] \tag{S.16}$$

which concludes the proof. Observe that the first expression within the expected value is simply the adjoint method gradient described previously in Eq. (S.8), and therefore requires two simulations per output vector $\mathbf{p}$ of the neural network. The additional term $\partial \mathbf{p}/\partial \boldsymbol{\theta}$ may be evaluated using automatic differentiation.

Note that in the original paper by Jiang and Fan [7, 8], the authors emphasized taking the exponential value of the electromagnetic merit function, $f \to \exp\{f/\sigma\}$, where σ is a hyperparameter. We can include this additional contribution explicitly in Eq. (S.11) and Eq. (S.16) through an application of the chain rule:

$$\boxed{\text{GLOnet Objective: } \mathop{\mathbb{E}}_{\mathbf{z}} \left[\exp \left\{ \frac{(f \circ g \circ h_{\boldsymbol{\theta}})(\mathbf{z})}{\sigma} \right\} \right]} \tag{S.17}$$

$$\boxed{\text{Backpropagation Gradient: } \mathop{\mathbb{E}}_{\mathbf{z}} \left[\frac{1}{\sigma} \exp \left\{ \frac{(f \circ g)(\mathbf{p})}{\sigma} \right\} \frac{\partial (f \circ g)}{\partial \mathbf{p}} \cdot \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}} \bigg|_{\mathbf{p}=h_{\boldsymbol{\theta}}(\mathbf{z})} \right]} \tag{S.18}$$

In the main text the exponential term was excluded for clarity, but for the GLOnet results in Fig. 2 this is the form of the objective that was used, where we chose hyperparameter $\sigma = 0.6$.

1.3 PHORCED

As mentioned the main text, PHORCED is qualitatively similar to GLOnet in the sense that we wish to use a generative neural network to suggest geometrical degrees-of-freedom. However, by contrast to GLOnet where $\mathbf{p}$ is provided deterministically from the neural network, in PHORCED we treat $\mathbf{p}$ as a random vector sampled from a conditional probability distribution, $\pi_{\boldsymbol{\theta}}$:

$$\mathbf{p} \sim \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z}) \tag{S.19}$$

where $\mathbf{z} \sim \mathcal{D}$ is once again an input vector, which in this case conditions the distribution. Note that, for given $\mathbf{z}$, the distribution defined by $\pi_{\boldsymbol{\theta}}$ is not static and is in fact programmable by virtue of the neural network weights $\boldsymbol{\theta}$ that determine its statistical parameters. In the main text of this paper we chose $\pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z})$ to be a multivariate Gaussian, with mean and covariance matrix defined by a generative neural network with weights $\boldsymbol{\theta}$.

Because $\mathbf{p}$ is now explicitly a random vector (not random just by virtue of $\mathbf{z}$ being random) the objective of PHORCED is modified relative to GLOnet (from Eq. (S.11)):

$$\boxed{\text{PHORCED Optimization: } \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \underset{(\mathbf{p},\mathbf{z})}{\mathbb{E}}[(f \circ g)(\mathbf{p})]} \tag{S.20}$$

where the subtle difference is we now wish to improve the expected value of the electromagnetic merit function under the joint probability of sampling random vectors $\mathbf{p}$ and $\mathbf{z}$ [2]. As a consequence of this, we note that we may write the joint probability density function of $(\mathbf{p}, \mathbf{z})$ as:

$$\text{pdf}(\mathbf{p}, \mathbf{z}) = \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z})\text{pdf}(\mathbf{z}) \tag{S.21}$$

Hence,

$$\underset{(\mathbf{p},\mathbf{z})}{\mathbb{E}}[(f \circ g)(\mathbf{p})] = \iint (f \circ g)(\mathbf{p})\text{pdf}(\mathbf{p}, \mathbf{z}) d\mathbf{p} d\mathbf{z} \tag{S.22}$$

$$= \iint (f \circ g)(\mathbf{p})\pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z})\text{pdf}(\mathbf{z}) d\mathbf{p} d\mathbf{z} \tag{S.23}$$

Then for scalar weight $\theta_j \in \boldsymbol{\theta}$ in our neural network, the partial derivatives of this expected value are given by:

$$\frac{\partial}{\partial \theta_j} \underset{(\mathbf{p},\mathbf{z})}{\mathbb{E}}[(f \circ g)(\mathbf{p})] = \iint (f \circ g)(\mathbf{p}) \frac{\partial \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z})}{\partial \theta_j} \text{pdf}(\mathbf{z}) d\mathbf{p} d\mathbf{z} \tag{S.24}$$

where we applied the Leibniz integral rule followed by the product rule. Observe the very important fact that neither $\mathbf{p}$ nor $(f \circ g)$ are explicitly related to θ_j, and therefore a term of the form $\frac{\partial (f \circ g)(\mathbf{p})}{\partial \theta_j}$ is zero in the product rule derivative. Indeed, only the policy distribution, $\pi_{\boldsymbol{\theta}}$, is modeled by the neural network, and therefore is the only quantity subject to the derivative. Moreover, we note that by the "log trick" we may write:

$$\frac{\partial \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z})}{\partial \theta_j} = \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z}) \frac{\partial}{\partial \theta_j} \log \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z}) \tag{S.25}$$

to which Eq. (S.24) becomes:

$$\frac{\partial}{\partial \theta_j} \underset{(\mathbf{p},\mathbf{z})}{\mathbb{E}}[(f \circ g)(\mathbf{p})] = \iint (f \circ g)(\mathbf{p}) \frac{\partial \log \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z})}{\partial \theta_j} \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z}) \text{pdf}(\mathbf{z}) d\mathbf{p} d\mathbf{z} \tag{S.26}$$

$$= \underset{(\mathbf{p},\mathbf{z})}{\mathbb{E}} \left[(f \circ g)(\mathbf{p}) \frac{\partial \log \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z})}{\partial \theta_j} \right] \tag{S.27}$$

where in the second step we used Eq. (S.25) and the definition of the expected value. Eq. (S.27) was reported in the main text as the quantity used for backpropagation. In our implementation we also included a "baseline subtraction" term, where we subtract the sample average of the electromagnetic merit function, $\text{average}_{\mathbf{p}}[(f \circ g)(\mathbf{p})]$, from the electromagnetic merit function each iteration of the optimization routine. This heuristic is well-known in the reinforcement learning to reduce model variance without affecting bias in expectation (Ref. [10]).

To summarize, the objective and backprogation terms for PHORCED are given by,

$$\boxed{\text{PHORCED Objective: } \underset{(\mathbf{p},\mathbf{z})}{\mathbb{E}}[(f \circ g)(\mathbf{p})]} \tag{S.28}$$

$$\boxed{\text{Backpropagation Gradient: } \underset{(\mathbf{p},\mathbf{z})}{\mathbb{E}} \left[\big((f \circ g)(\mathbf{p}) - b\big) \frac{\partial \log \pi_{\boldsymbol{\theta}}(\mathbf{p}|\mathbf{z})}{\partial \boldsymbol{\theta}} \right]} \tag{S.29}$$

$$\boxed{b \approx \underset{(\mathbf{p},\mathbf{z})}{\mathbb{E}}[(f \circ g)(\mathbf{p})]} \tag{S.30}$$

Observe that PHORCED requires no evaluation of the gradient of the electromagnetic figure of merit: $\frac{\partial (f \circ g)}{\partial \mathbf{p}}$. Indeed, the foremost term within the gradient expression requires only "forward simulation" evaluations. Meanwhile, the latter term $\partial \log \pi_{\boldsymbol{\theta}} / \partial \boldsymbol{\theta}$ may be computed using pytorch's automatic differentiation feature. These features of PHORCED should be contrasted with the similar equations from GLOnet (Eqs. (S.17)-(S.18)), where we note that the optimization objective is identical but the gradient required for backpropagation changes drastically by our representation of the neural network model.

[2] In the general reinforcement learning literature, the merit function within the expected value may be a function of both $\mathbf{p}$ and $\mathbf{z}$, but this situation was excluded because it is non-applicable here.

2 Neural Network Model Specifications

In Fig. S1 we illustrate the architectures and hyperparameters for the implementation of the generative neural network models used to obtain the results in Figs. 2-4 from the main text. Both PHORCED and GLOnet are used to output design vectors of dimension 60, representing the width and spacing between 30 grating etches. Note that there are better performing choices of grating coupler etch parameterization [1, 3], but this parameterization was chosen to illustrate how the algorithm would behave in high-dimensional situations where less physical intuition is available. Both models were implemented in PyTorch and electromagnetic simulations were performed in EMopt [6]. Hyperparameter definitions are provided in Fig. S2. Note that we were unable to perform exhaustive hyperparameter testing in this work due to the prohibitively slow speed of electromagnetic simulations of this size. Training with PHORCED and GLOnet required ~12 and ~36 hours on a high-performance server.

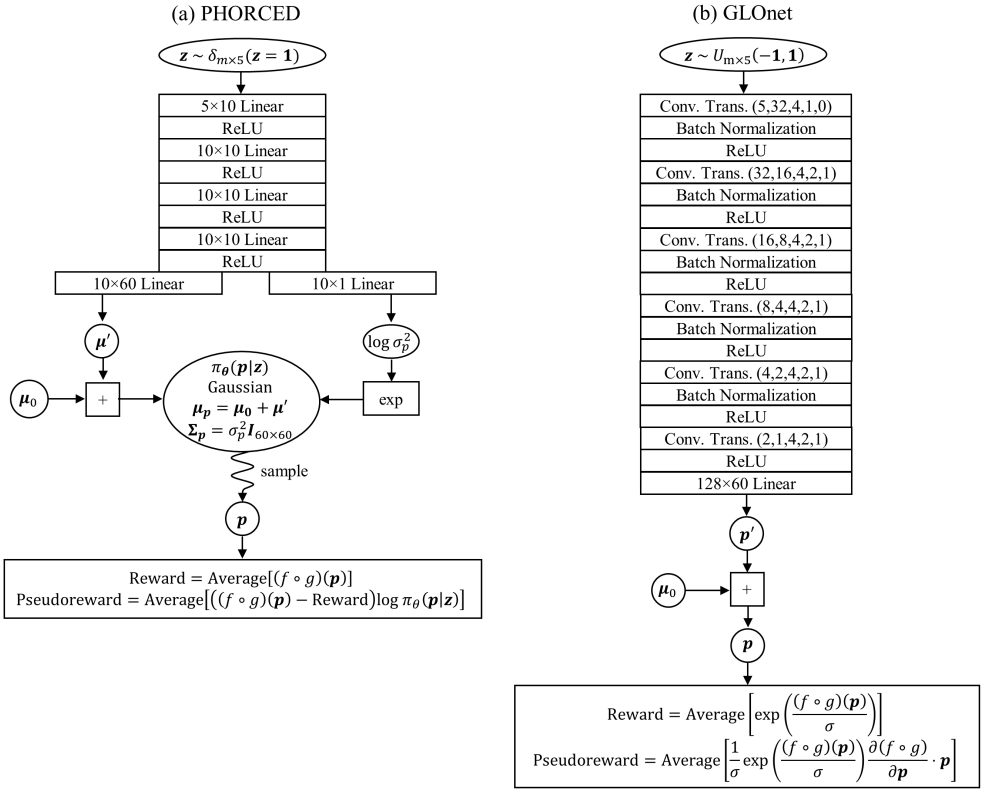

Figure S1: Neural network models used for optimization with (a) PHORCED and (b) GLOnet. Note that "Reward" is the objective of interest, whereas the "Pseudoreward" is the objective value actually seen by the neural network. This allows us to conveniently invoke automatic differentiation despite performing electromagnetic simulations and gradient calculations outside of PyTorch's autograd framework. Other definitions and hyperparameters may be found in Fig. S2.

μ_0 = Initial Seed (of Grating Coupler)
Optimization Settings: Adam w/ AMSgrad enabled
Learning Rate (PHORCED) = 3e-3 (2e-3 for some transfer learning opts)
Learning Rate (GLOnet) = 1e-3
m = batch size = 10 samples per iteration
Iterations = 1000
σ = 0.6
Conv. Trans. (v,w,x,y,z) = 1-Dimensional Convolution Transpose
v = in channels, w = out channels, x = kernel size
y = stride, z = padding
All applicable neural net layers have bias enabled

Figure S2: Definitions and hyperparameter specifications.

References

[1] A. Michaels and E. Yablonovitch, "Inverse design of near unity efficiency perfectly vertical grating couplers," *Optics Express*, vol. 26, no. 4, pp. 4766–4779, Feb. 2018. DOI: 10.1364/OE.26.004766.

[2] L. Su, R. Trivedi, N. V. Sapra, A. Y. Piggott, D. Vercruysse, and J. Vučković, "Fully-automated optimization of grating couplers," *Optics Express*, vol. 26, no. 4, pp. 2614–2617, 2017. DOI: 10.1364/OE.26.004023.

[3] S. Hooten, T. V. Vaerenbergh, P. Sun, S. Mathai, Z. Huang, and R. G. Beausoleil, "Adjoint Optimization of Efficient CMOS-Compatible Si-SiN Vertical Grating Couplers for DWDM Applications," *Journal of Lightwave Technology*, vol. 38, no. 13, pp. 3422–3430, Jul. 2020. DOI: 10.1109/JLT.2020.2969097.

[4] P. Sun, T. V. Vaerenbergh, M. Fiorentino, and R. Beausoleil, "Adjoint-method-inspired grating couplers for CWDM O-band applications," *Optics Express*, vol. 28, no. 3, pp. 3756–3767, Feb. 2020. DOI: 10.1364/OE.382986.

[5] M. K. Dezfouli, Y. Grinberg, D. Melati, J. H. Schmid, P. Cheben, S. Janz, and D.-X. Xu, "Design of fully apodized and perfectly vertical surface grating couplers using machine learning optimization," in *Integrated Optics: Devices, Materials, and Technologies XXV*, vol. 11689, International Society for Optics and Photonics, Mar. 2021, 116890J. DOI: 10.1117/12.2576945.

[6] A. Michaels, *EMopt*, May 2019. [Online]. Available: https://github.com/anstmichaels/emopt.

[7] J. Jiang and J. A. Fan, "Global Optimization of Dielectric Metasurfaces Using a Physics-Driven Neural Network," *Nano Letters*, vol. 19, no. 8, pp. 5366–5372, Aug. 2019. DOI: 10.1021/acs.nanolett.9b01857.

[8] J. Jiang and J. A. Fan, "Simulator-based training of generative neural networks for the inverse design of metasurfaces," *Nanophotonics*, vol. 9, no. 5, pp. 1059–1069, 2020. DOI: doi:10.1515/nanoph-2019-0330.

[9] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.