A “generative” model for computing electromagnetic field solutions

Ben Bartlett [*]
Department of Applied Physics, Stanford University

Abstract

We present an unsupervised machine learning model for computing approximate electromagnetic fields in a cavity containing an arbitrary spatial dielectric permittivity distribution. Our model achieves good predictive performance and is over $10\times$ faster than identically-sized finite-difference frequency-domain simulations, suggesting possible applications for accelerating optical inverse design algorithms.

1 Introduction

“Inverse design” problems – computational design of structures by specifying an objective function – are pervasive throughout physics, especially in photonics, where inverse design methods have been used to design many highly compact optical components [1], [2]. Optical inverse design algorithms involve simulating the electromagnetic fields within the device at each iteration of the design process, typically by using the finite-difference frequency-domain (FDFD) method, and then optimizing the design region using adjoint variable methods [3].

The iterative FDFD simulations, although exact, can be computationally expensive and scale poorly with the design dimensions. For many applications, an approximate field solution is sufficient. A machine learning model which could quickly compute approximate electromagnetic fields for a dielectric structure could reduce this computational bottleneck, allowing for much faster inverse design processes. [4]

In this paper, we present a machine learning model for computing approximate electromagnetic field solutions. The model takes as its input a vector of dielectric permittivities at each point in space and computes an equally-sized vector representing the approximate electric field amplitude at each point. The model is trained using an entirely unsupervised approach which is loosely analogous in structure to a generative adversarial network [5]. Our model achieves good predictive performance, generalizes well to structures outside of its training distribution, and computes fields over $10\times$ faster than an identically-sized FDFD simulation.

2 Related work

We were able to find a small body of existing work related to this problem. Shan, et al. [6] demonstrated a neural solver for Poisson’s equations using a purely-convolutional neural network. Their model took as inputs a dielectric permittivity matrix and a matrix of pixel-wise distance to the field source and was trained against labeled FDFD solutions. Our initial work followed their approach, except applied to solving Maxwell’s equations, but we were unable to reproduce their results.

Lagaris, et al. [7] presented a method similar to the one used in this paper to solve initial and boundary value problems of a specific form using artificial neural networks. Two prior CS229 projects [8], [9] applied this method to solve specific parameterizations of Poisson’s equations and studied its

[*]SUNet ID: `muon`, email: `benbartlett@stanford.edu`

error properties. McFall and Mahan [10] expanded Lagaris's method to solve problems over irregular domain boundaries with mixed Dirichlet/Neumann boundary conditions.

3 Methods

3.1 Problem summary and approach

Our model computes approximate electromagnetic field solutions for a specific type of scenario, formalized here. Suppose we have a perfectly reflective d-dimensional[2] cavity of length L with an electromagnetic source at the center. The cavity contains material forming an arbitrary spatial distribution of dielectric permittivity $\epsilon(x)$. Discretizing the cavity into "pixels" of size δL, the permittivity at each point in space can be expressed as a vector $\vec{\epsilon}$ of size $N = (L/\delta L)^d$. Given an input permittivity vector $\vec{\epsilon}$ and knowledge of the source location, the model outputs an identically-sized vector $\vec{E}_{\text{pred}}$ representing the electric field amplitude at each point in space. The cavity scenario was chosen to impose Dirichlet boundary conditions of $\vec{E} = 0$ at the cavity edges, ensuring the electric fields are standing waves, and thus real up to a global phase.[3]

The model learns to produce realistic field solutions in an entirely unsupervised manner described in Section 3.2 using a metric for physical realism we call the "Maxwell residual", denoted as $\mathcal{L}_M$. When trained to minimize the deviation from physical realism measured by $\mathcal{L}_M$, the model outputs predicted fields which are very close to the "true" solutions obtained from an FDFD solver[4], despite never seeing them during training. Because of the entirely unsupervised training approach, we can train our model on arbitrarily large (and even infinitely enumerable) datasets, as no data labeling is required.

The structure of the model is loosely analogous to a generative adversarial network [5]. The first part of the model is a "generator" which maps randomly generated input permittivities to field outputs as $G : \vec{\epsilon}^{(i)} \to \vec{E}^{(i)}_{\text{pred}}$. The second part is a "discriminator"[5] which computes the Maxwell residual of the predicted field as $D : \vec{E}^{(i)}_{\text{pred}} \to \mathcal{L}^{(i)}_M$, providing a measure of how physically realistic the generator's outputs are. In both cases, the loss of the total model is $\mathcal{L}^{(i)} = D(G(\vec{\epsilon}^{(i)}))$.

3.2 Unsupervised training with Maxwell residuals

Maxwell's equations govern the dynamics and propagation of electromagnetic fields in materials and form the foundation for classical electromagnetism. [13] In SI units, they are written as:

$$\begin{aligned} \nabla \cdot \vec{E} &= \frac{\rho}{\epsilon} & \qquad \nabla \cdot \vec{B} &= 0 \\ \nabla \times \vec{E} &= -\frac{\partial \vec{B}}{\partial t} & \qquad \nabla \times \vec{B} &= \mu \vec{J} + \mu\epsilon \frac{\partial \vec{E}}{\partial t}, \end{aligned} \tag{1}$$

where $\vec{E}, \vec{B}$ are electric and magnetic fields at a given point in space and time, ϵ, μ are the permittivity and permeability of the material, t is time, ρ is charge density, and $\vec{J}$ is current density. In a non-magnetic, electrically neutral, linear material (such as many cases of interest), $\rho = 0$, $\mu = \mu_0$, and these equations can be simplified to:

[2]In our research for this project, we explored $d = 1, 2$, although we only have space to present $d = 1$ results; 2D simulations can be found in the project repository and in the poster.

[3]This was important as PyTorch [11] currently lacks complex number support.

[4]All FDFD simulations in this paper were computed using the `angler` FDFD package. [12]

[5]The biggest difference between the structure of our model and a GAN is that our "discriminator" is not a trainable model: rather, it computes $\mathcal{L}_M$ from a predicted field using a static transformation. Our model is arguably generative, but not truly adversarial.

$$\nabla \times \vec{E} = -\mu_0 \frac{\partial \vec{H}}{\partial t} \tag{2}$$

$$\nabla \times \vec{H} = \epsilon \frac{\partial \vec{E}}{\partial t} + \vec{J}, \tag{3}$$

where $\vec{H} \equiv \vec{B}/\mu_0$ is the magnetizing field. In a steady-state frequency domain solution such as the ones found with FDFD methods, $\vec{E}(t) = \vec{E}e^{i\omega t}$, so $\partial_t(\cdot) \to \omega(\cdot)$, where ω is frequency. We can combine (2) and (3) to obtain an equation which any solution to Maxwell's equations must satisfy:

$$\left[(\nabla \times \nabla \times) - \omega^2 \mu_0 \epsilon\right] \vec{E} - \vec{J} = 0. \tag{4}$$

If the electromagnetic field is polarized, say, with E_z polarization, then $\vec{E} = E\hat{z}$ and $\vec{J} = J\hat{z}$ at each point in space. We can then "vectorize" this such that $\vec{E}$ and $\vec{J}$ are the electric field and free current amplitudes in the $\hat{z}$ direction and $\vec{\epsilon}$ is the dielectric permittivity at each point in space. If we have a model which takes in a permittivity distribution $\vec{\epsilon}$ and a source term $\vec{J}$ and returns a predicted field $\vec{E}_{\text{pred}}$, then we use Eq. 4 to define the "Maxwell residual" $\mathcal{L}_M$ as:

$$\mathcal{L}_M \equiv \left[(\nabla \times \nabla \times) - \omega^2 \mu_0 \vec{\epsilon}\right] \vec{E}_{\text{pred}} - \vec{J}. \tag{5}$$

The Maxwell residual provides an element-wise measure of the physical realism of the predicted field $\vec{E}_{\text{pred}}$ (a measure of how far the predicted solution is from satisfying Maxwell's equations at each point). If the model can sufficiently minimize $\mathcal{L}_M$, then it can produce solutions which approximately satisfy Maxwell's equations at each point, and thus are approximate global electromagnetic field solutions for the system described by $\vec{\epsilon}$ and $\vec{J}$. This training does not require the model to ever see the exact FDFD field solution (the outputs it attempts to replicate) and is thus unsupervised.

3.3 Model architecture and implementation

We found that when trained (or more precisely, overfit) to predict the field of a single permittivity distribution $\vec{\epsilon}$, virtually any network architecture would allow the predicted field to converge to the true field given enough training time. (For more on this, see Section 4.1.) The challenge was finding a network architecture with the correct structure to capture the generalized transformation $\vec{\epsilon} \mapsto \vec{E}$ when trained on a large number of possible permittivities.

We tested many different network architectures for this project. Purely convolutional architectures, like the ones used by Ref. [6], did not perform well and seemed incapable of capturing nonlocal field dependence due to distant reflections with the cavity walls. Purely dense architectures, like the ones used (for training single structures) by Refs. [7]–[9] did not seem to capture the physics of the problem by generalizing well to structures very different from the training distribution.

Our final network architecture employed a hybrid convolutional / dense / deconvolutional approach and is shown in Figure 1. The network starts with three convolutional layers, intended to capture certain features of the permittivity input $\vec{\epsilon}$ such as refractive index changes and layer thicknesses. These feed into two dense layers, which allows the model to better account for nonlocal field interactions. Finally, three transposed-convolutional layers expand the signal to the original input size, providing the prediction for $\vec{E}$. We found that the performance of the model was relatively insensitive to the choice of kernel size and number of convolutional/deconvolutional layers in excess of 3.

During training, the network outputs the Maxwell residual $\mathcal{L}_M(\vec{E})$. Dropout layers with $p = 0.1$ and ReLU activations are present after every layer except the last one. Our model was implemented using PyTorch [11], and the code is available at `https://github.com/bencbartlett/neural-maxwell`.

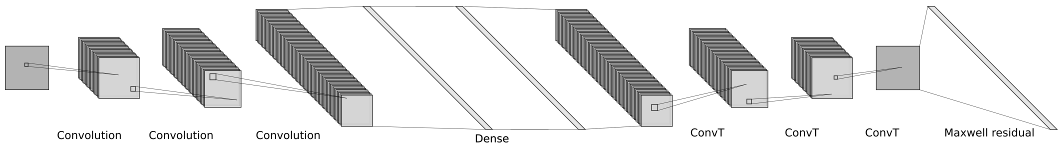

Figure 1: Final architecture for the neural Maxwell solver. The model takes as inputs a vector of permittivities $\vec{\epsilon}$. Three successive convolutional layers (kernel sizes: 5, 7, 9, channels: 32, 64, 128) output into two appropriately-sized dense layers. This outputs into three successive transposed-convolutional layers (kernel sizes: 9, 7, 5, channels: 128, 64, 32), expanding the signal to the original dimensions of $\vec{\epsilon}$. Dropout layers with $p = 0.1$ and ReLU activations follow all but the last layer.

4 Experiments

4.1 Fitting to single $\vec{\epsilon}$

As an initial experiment, we trained the model to predict the field of only a single $\vec{\epsilon}$ input using the Maxwell residual method described in Section 3.2. The evolution of the predicted field as the network is trained on a sample permittivity is shown in Figure 2. (An animated version of this figure is available online at `https://gfycat.com/TestyWanIsopod`.) We ran this procedure dozens of times, varying network architectures and permittivity, and found that the model would eventually converge (with loss less than 10^{-7}) to the exact FDFD field solution for virtually all $\vec{\epsilon}$ and all network architectures given sufficient training time. This experiment was not terribly useful in terms of obtaining practical speedup for EM simulation problems, as the typical time to train to convergence exceeded the time to run an equivalent FDFD simulation by a factor of about 100. However, from an academic standpoint, it is an interesting method to numerically solve Maxwell's equations, and looking at the time to convergence for a single $\vec{\epsilon}$ provided some insight into prototyping optimal network architectures for the main experiment detailed in Section 4.2.

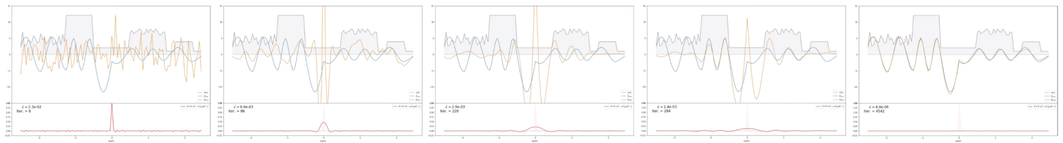

Figure 2: Evolution of the predicted field as the network is trained on a single permittivity input. The top panel of each image depicts the permittivity at each point (grey), the "true" electric field from an FDFD simulation (blue) and the predicted field at the given iteration (orange). The bottom panel depicts $\mathcal{L}_M$ at each iteration. An animated version of this figure (which is more informative and which we encourage readers to view) is available online at `https://gfycat.com/TestyWanIsopod`.

4.2 Training on permittivity datasets

For the main experiment in this paper, we trained a model with the architecture described in Figure 1 on a large dataset of 10^6 randomly-generated permittivities. Each $\vec{\epsilon}$ sample represented the permittivity formed by a random number of material layers, each of random thickness, of alternating silicon and vacuum. The model was trained using an Adam optimizer [14] with batch size 200 and learning rate 5×10^{-6} on an NVIDIA Tesla K80 until convergence (after about 8 hours and 400 epochs, with a final average loss of 8×10^{-4}).

To evaluate the results of the trained model, a test set of 10^4 new permittivity samples was generated using the same generation procedure. The model was run on each of these inputs, the loss for each sample was calculated (average loss of 8.8×10^{-4}), and the results were sorted from best to worst. Example good (best 10/10000), typical (middle 10/10000), and bad (worst 10/10000) field predictions from the test set are shown in the first three panels of Figure 3. For each sample, the forward-pass time was recorded and compared to the FDFD simulation time; the trained model takes an average of 1.2ms to compute predicted fields – over $10\times$ faster[6] than the 14ms equivalent FDFD simulation time.

[6]The model performs even faster per sample if it evaluates batched inputs.

Finally, we tested the model's capability to generalize to inputs outside of the training distribution – that is, permittivities representing a different set of structures than the ones generated for the training and test sets. As an example, the predicted field amplitudes for a sample $\vec{\epsilon}$ where each point in space has a permittivity value randomly chosen between vacuum and silicon is shown in the last panel of Figure 3. (This $\vec{\epsilon}$ is pathological and would not represent any type of device which could be easily fabricable, but illustrates the generalization capabilities of the model.)

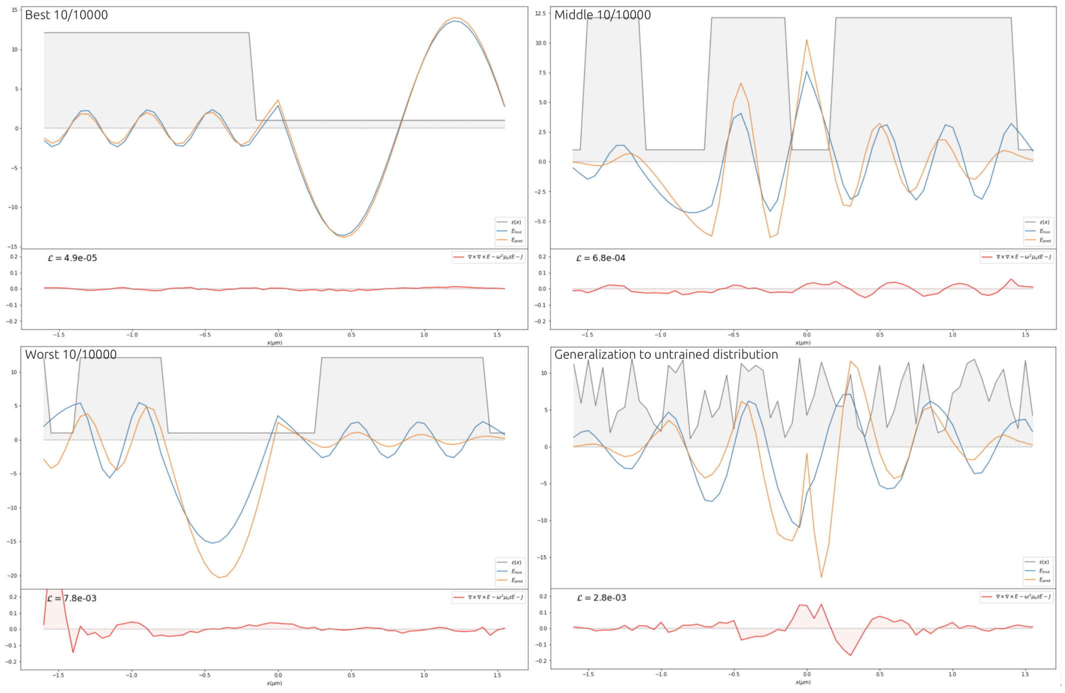

Figure 3: Permittivities and predicted fields for samples in the test set with good, typical, and poor performance. The lower right panel shows the predicted fields for a uniform-random permittivity input from a distribution the model was not trained on.

5 Discussion

In this paper, we presented a machine learning model capable of computing approximate solutions to Maxwell's equations over an arbitrary dielectric permittivity in a cavity. The model was trained using an unsupervised approach where it learned to minimize the "Maxwell residual" of its predicted fields, thereby maximizing the physical realism of the solutions. Our model demonstrates good predictive performance and is over $10\times$ faster than comparable FDFD simulations, suggesting applications for accelerating optical inverse design algorithms.

For future work, we would like to implement a complex-valued model to solve the more general problem of predicting fields outside of a cavity environment. Our choice of the cavity problem was driven primarily by PyTorch's lack of support for complex tensors. (In the project repository, we have an initial implementation of this which explicitly parameterizes $\Re(\vec{E})$ and $\Im(\vec{E})$, although this approach was only mildly successful.) We would also like to explore using our model for dimensionality reduction, especially for 2D and 3D problems. We were able to achieve a 1:16 dimensionality reduction with our model applied to a 32×32 2D input of permittivities by adjusting the network parameters to force a 64-value chokepoint in the middle dense layers of the network. (This figure is present in the poster but omitted here due to length constraints.) This could force the model to learn more efficient representations of the relationships between permittivities and fields.

Acknowledgements

We would like to thank Shanhui Fan, Sunil Pai, and Tyler Hughes for several illuminating discussions relating to this work.

Source code

All source code used for this paper is available at `https://github.com/bencbartlett/neural-maxwell`. Trained model parameters were too large to include in the repository but are are available upon request.

References

[1] A. Y. Piggott, J. Lu, T. M. Babinec, K. G. Lagoudakis, J. Petykiewicz, and J. Vučković, "Inverse design and implementation of a wavelength demultiplexing grating coupler," *Scientific Reports*, 2014, ISSN: 20452322. DOI: `10.1038/srep07210`.

[2] L. Su, A. Y. Piggott, N. V. Sapra, J. Petykiewicz, and J. Vučković, "Inverse Design and Demonstration of a Compact on-Chip Narrowband Three-Channel Wavelength Demultiplexer," *ACS Photonics*, 2018, ISSN: 23304022. DOI: `10.1021/acsphotonics.7b00987`.

[3] T. Hughes, G. Veronis, K. P. Wootton, R. Joel England, and S. Fan, "Method for computationally efficient design of dielectric laser accelerator structures," *Optics Express*, vol. 25, no. 13, p. 15 414, Jun. 2017, ISSN: 1094-4087. DOI: `10.1364/OE.25.015414`. [Online]. Available: `https://www.osapublishing.org/abstract.cfm?URI=oe-25-13-15414`.

[4] J. Peurifoy, Y. Shen, L. Jing, Y. Yang, F. Cano-Renteria, B. G. DeLacy, J. D. Joannopoulos, M. Tegmark, and M. Soljačić, "Nanophotonic particle simulation and inverse design using artificial neural networks," *Science Advances*, vol. 4, no. 6, pp. 1–8, 2018, ISSN: 23752548. DOI: `10.1126/sciadv.aar4206`.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," Jun. 2014. [Online]. Available: `https://arxiv.org/abs/1406.2661`.

[6] T. Shan, W. Tang, X. Dang, M. Li, F. Yang, S. Xu, and J. Wu, "Study on a Poisson's Equation Solver Based On Deep Learning Technique," Dec. 2017. [Online]. Available: `http://arxiv.org/abs/1712.05559`.

[7] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, 1998, ISSN: 10459227. DOI: `10.1109/72.712178`.

[8] M. Chiaramonte and M. Kiener, "Solving differential equations using neural networks," PhD thesis, Stanford University, 2013.

[9] S. H. Kolluru, "A Neural Network Based ElectroMagnetic Solver," PhD thesis, Stanford University, 2017.

[10] K. S. McFall and J. R. Mahan, "Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions," *IEEE Transactions on Neural Networks*, 2009, ISSN: 10459227. DOI: `10.1109/TNN.2009.2020735`.

[11] Facebook AI Research, "PyTorch: tensors and dynamic neural networks in Python with strong GPU acceleration," 2018. [Online]. Available: `https://pytorch.org/`.

[12] T. W. Hughes, M. Minkov, I. A. D. Williamson, and S. Fan, "Adjoint method and inverse design for nonlinear nanophotonic devices," Nov. 2018. [Online]. Available: `https://arxiv.org/abs/1811.01255`.

[13] J. D. Jackson, *Classical Electrodynamics, 3rd Edition*. 1998, ISBN: 047130932X. DOI: `10.1119/1.19136`.

[14] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014. [Online]. Available: `https://arxiv.org/abs/1412.6980`.