


Exploiting graph neural networks to perform finite-difference time-domain based optical simulations


Cite as: APL Photonics **8**, 036109 (2023); <https://doi.org/10.1063/5.0139004>

Submitted: 16 December 2022 • Accepted: 14 February 2023 • Accepted Manuscript Online: 14 February 2023 • Published Online: 14 March 2023

 L. Kuhn,  T. Repän and  C. Rockstuhl

COLLECTIONS

 This paper was selected as Featured

 This paper was selected as Scilight



View Online



Export Citation



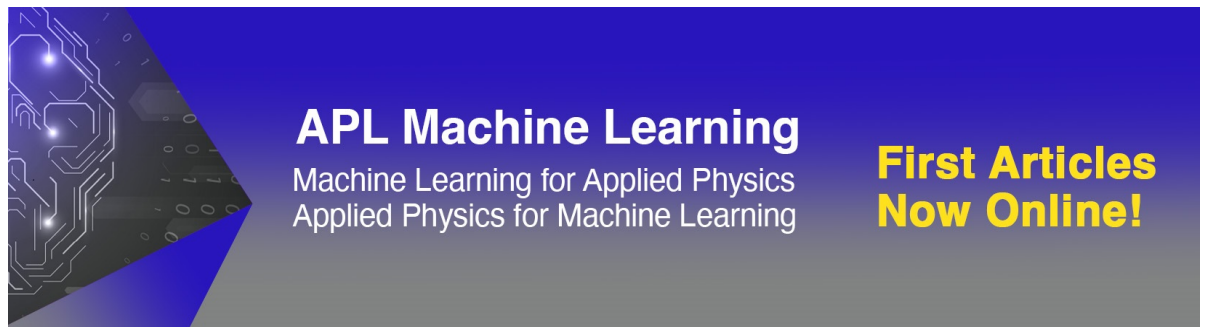
CrossMark

ARTICLES YOU MAY BE INTERESTED IN

[Graph neural networks solve Maxwell's equations numerically](#)
Scilight **2023**, 111103 (2023); <https://doi.org/10.1063/10.0017619>

[Polarization-mediated multi-state infrared system for fine temperature regulation](#)
APL Photonics **8**, 030801 (2023); <https://doi.org/10.1063/5.0136842>

[MaxwellNet: Physics-driven deep neural network training based on Maxwell's equations](#)
APL Photonics **7**, 011301 (2022); <https://doi.org/10.1063/5.0071616>



APL Machine Learning
Machine Learning for Applied Physics
Applied Physics for Machine Learning

**First Articles
Now Online!**

Exploiting graph neural networks to perform finite-difference time-domain based optical simulations



Cite as: APL Photon. 8, 036109 (2023); doi: 10.1063/5.0139004
Submitted: 16 December 2022 • Accepted: 14 February 2023 •
Published Online: 14 March 2023



L. Kuhn,^{1,2} T. Repän,³ and C. Rockstuhl^{2,4,a)}

AFFILIATIONS

¹ Steinbuch Centre for Computing—Scientific Computing and Mathematics, Karlsruhe Institute of Technology, Karlsruhe, Germany

² Institute of Theoretical Solid State Physics, Karlsruhe Institute of Technology, Karlsruhe, Germany

³ Institute of Physics, University of Tartu, Tartu, Estonia

⁴ Institute of Nanotechnology, Karlsruhe Institute of Technology, Karlsruhe, Germany

^{a)} Author to whom correspondence should be addressed: carsten.rockstuhl@kit.edu

ABSTRACT

Having an artificial neural network that solves Maxwell's equations in a general setting is an intellectual challenge and a great utility. Recently, there have been multiple successful attempts to use artificial neural networks to predict electromagnetic fields, given a specific source and interacting material distribution. However, many of these attempts are limited in domain size and restricted to object shapes similar to the learned ones. Here, we overcome these restrictions by using graph neural networks (GNNs) that adapt the propagation scheme of the finite-difference time-domain (FDTD) method to solve Maxwell's equations for a distinct time step. GNNs yield a significant advantage, i.e., size invariance, over conventional neural network architectures, such as convolutional or linear neural networks. Once trained, a GNN can work on graphs of arbitrary size and connectivity. This allows us to train them on the propagation procedure of electromagnetic fields on small domain sizes and, finally, expand the domain to an arbitrary scale. Moreover, GNNs can adapt to any material shape and work not only on structured grids, such as FDTD, but also on arbitrary meshes. This work may be seen as the first benchmark for field predictions with graph networks and could be expanded to more complex mesh-based optical simulations, e.g., those based on finite elements.

© 2023 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>). <https://doi.org/10.1063/5.0139004>

I. INTRODUCTION

Nanophotonics explores the interaction of light with materials structured on a length scale comparable to or smaller than the wavelength.^{1,2} Novel phenomena encountered at the nanoscale and the disruptive means to tailor the flow of light constitute an ever-increasing motivation to explore such systems.³ For these explorations, it is of major importance to reliably simulate the light-matter interaction.⁴ On the one hand, it drives the development of novel materials, structures, and devices. On the other hand, it provides insights into the light propagation in actual systems inaccessible by experimental means. The numerical experiment, as it is often called, is indispensable to understanding nanophotonic devices.^{5,6}

One of the most common and well-known numerical simulation tools is the finite-difference time-domain method (FDTD).⁷ It originates from a discrete version of Maxwell's equations and evolves electromagnetic fields in time-domain with high accuracy. The FDTD method is particularly useful for understanding how light interacts with a given structure. It allows us to define an incident field and a given structure and to observe the evolution of the field in that specific setting afterward.^{8,9} With that, it provides a way to observe the evolution of light exactly as it would happen in the real world. However, as with all computational techniques, the speed could always be accelerated and the domain size that can be handled could always be larger. While the parallelization of FDTD algorithms is fairly standard, it quickly requires computational infrastructures for its execution that are not accessible to everybody. Thus, even

moderately sized problems may come along with a high computational cost and time.

Lately, in the slip-stream of developments in the field of computer sciences, in general, and machine learning, in particular, there have been tremendous efforts to approach this problem with the use of deep learning-based data-driven methods.^{10–13} Many of these attempts have been very successful and promoted not only accelerated simulation but also inverse design processes.^{14–19} In addition, a few approaches were already reported that looked into the question of how to translate an FDTD simulation to a machine learning approach.^{20,21} Generally, this reads extremely rewarding. Once trained, the neural networks are expected to solve the same computational problems in a much shorter time. While the general feasibility was demonstrated in these pioneering prior contributions, these initial approaches also came along with some drawbacks caused by deep-learning architectures and a lack of generalizability due to limited data. This is a commonly encountered problem, and these surrogate physics solver approaches were restricted with respect to the materials and shapes of the scatterers, as well as the domain size and scale of the simulation. These previously considered neural networks mostly include convolutional layers, which learn features on a certain scale and are not able to extrapolate on an arbitrary resolution. Additionally, the input shape of the model has to be fixed. Thus, variable domain sizes could not be considered.

To circumvent these aforementioned problems, we consider here the use of graph neural networks (GNNs) for the same purpose. In general, GNNs are suitable to process unstructured, non-uniform data and have been already useful to classify molecules,^{22,23} to model fluid and particle dynamics,^{24,25} for computer vision tasks,²⁶ in particle and high-energy physics,^{27,28} or to work as graph neural operators.^{29–31} GNNs work independently of the graph size or structure and can easily store information about scaling and resolution.

We exploit these features to train GNNs to propagate electromagnetic field distributions in time for arbitrary domain sizes and material distributions. We stress that in this approach, we do not necessarily seek for steady-state solutions of the field but, instead, seek to advance the field propagation by a certain amount of time. The main purpose of our GNNs is to learn the FDTD scheme while exploiting all benefits offered by GNNs, e.g., working on unstructured grids, extrapolation to larger domain sizes, and extrapolation to material distributions not seen in the training process.

This paper is structured as follows: first, we discuss our methods, including FDTD, graphs, and graph neural networks. Subsequently, we outline the physical setup and training data generation. Here, we follow two approaches and focus either on the prediction of the electric field only or on performing full-field predictions. Finally, we conclude with our results and sketch improvements, extensions, and applications.

II. METHODS

A. The finite-difference time-domain method

The finite-difference time-domain method (FDTD) is an iterative numerical update procedure to compute the evolution of an electromagnetic field in space and time.³² For reasons of consistency, we will briefly sketch the approach in 2D assuming a transverse-magnetic (TM) polarization, starting from the source-free Maxwell

equations. Please note that, since different conventions exist, TM is understood in this contribution as the situation where, in an x - y plane, the considered electric field has a non-zero component perpendicular to this plane, while the magnetic field has non-zero components in this plane. An understanding of how the FDTD works is necessary to appreciate the formulation in form of a graph afterward, which is the natural starting point for a graph neural network to solve these equations.

For simplicity, we write down Maxwell's equations for a non-dispersive material characterized in the spatial domain by permittivity $\epsilon(\mathbf{r})$, permeability $\mu(\mathbf{r})$, electric conductivity $\sigma(\mathbf{r})$, and magnetic conductivity $\sigma_m(\mathbf{r})$ as the point of departure. These Maxwell equations read as

$$-\sigma_m(\mathbf{r})\mathbf{H}(\mathbf{r},t) - \mu(\mathbf{r})\frac{\partial\mathbf{H}(\mathbf{r},t)}{\partial t} = \nabla \times \mathbf{E}(\mathbf{r},t) = \hat{\mathbf{e}}_x \frac{\partial E_z(\mathbf{r},t)}{\partial y} - \hat{\mathbf{e}}_y \frac{\partial E_z(\mathbf{r},t)}{\partial x}, \quad (1)$$

$$\sigma(\mathbf{r})\mathbf{E}(\mathbf{r},t) - \epsilon(\mathbf{r})\frac{\partial\mathbf{E}(\mathbf{r},t)}{\partial t} = \nabla \times \mathbf{H}(\mathbf{r},t) = \hat{\mathbf{e}}_z \frac{\partial H_y(\mathbf{r},t)}{\partial x} - \hat{\mathbf{e}}_z \frac{\partial H_x(\mathbf{r},t)}{\partial y}, \quad (2)$$

$$\nabla \cdot \mathbf{E}(\mathbf{r},t) = 0, \quad (3)$$

$$\nabla \cdot \mathbf{H}(\mathbf{r},t) = 0. \quad (4)$$

In these equations, $\mathbf{E}(\mathbf{r},t)$ is the electric field, $\mathbf{H}(\mathbf{r},t)$ is the magnetic field, $\sigma_m(\mathbf{r})$ is the magnetic conductivity, $\sigma(\mathbf{r})$ is the electric conductivity, and $\epsilon(\mathbf{r})$ and $\mu(\mathbf{r})$ are the permeability and permittivity, respectively. If dispersive materials are required to be considered, auxiliary differential equations would be needed to supplement these Maxwell equations. These auxiliary equations would describe the temporal evolution of a polarization or magnetization depending on the electric or magnetic field, respectively.

In our considered setting, i.e., the propagation of the fields in the x - y -plane and with the restriction to TM polarization, the differential equations that describe the spatial and temporal evolution of the three involved field components are

$$-\sigma_m H_x(\mathbf{r},t) - \mu \frac{\partial H_x(\mathbf{r},t)}{\partial t} = \frac{\partial E_z(\mathbf{r},t)}{\partial y}, \quad (5)$$

$$\sigma_m H_y(\mathbf{r},t) + \mu \frac{\partial H_y(\mathbf{r},t)}{\partial t} = \frac{\partial E_z(\mathbf{r},t)}{\partial x}, \quad (6)$$

$$\sigma E_z(\mathbf{r},t) + \epsilon \frac{\partial E_z(\mathbf{r},t)}{\partial t} = \frac{H_y(\mathbf{r},t)}{\partial x} - \frac{\partial H_x(\mathbf{r},t)}{\partial y}. \quad (7)$$

To render the equations manageable on a computer, the space and time are discretized in the FDTD on a finite grid spaced by $\Delta x = \Delta y$ and Δt . For the spatial discretization, we define a grid indexed by m and n for the x - and y -direction, respectively; thus, $\mathbf{r} = (x, y) = (m\Delta x, n\Delta y)$. The temporal steps follow $\Delta t = \frac{1}{c} \left\{ \left(\frac{1}{\Delta x} \right)^2 + \left(\frac{1}{\Delta y} \right)^2 \right\}^{-\frac{1}{2}}$, where c is the speed of light in the

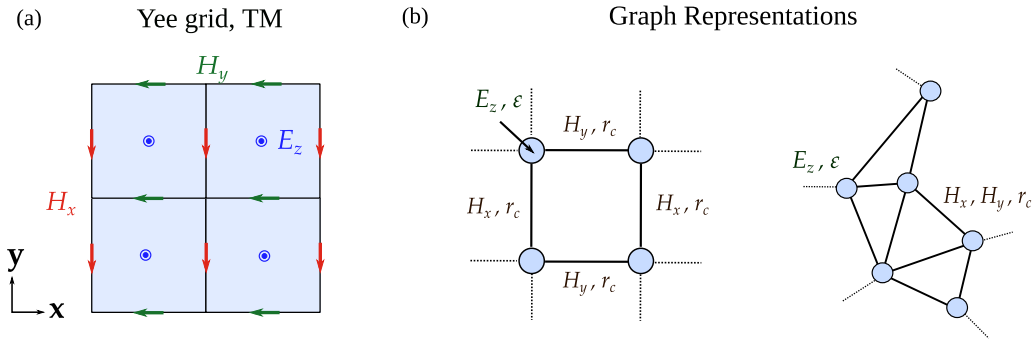


FIG. 1. (a) Yee-grid in TM polarization. (b) Graph representations as a grid and an unstructured mesh. The node features encode the electric field E_z and permittivity ϵ , and the edge features hold $H_{x,y}$ and local Cartesian vector r_c .

medium, leading to discrete times $t_q = q\Delta t$. To find suitable update rules for E and H , they have to be arranged in a staggered manner in space and time. The resulting spatial discretization follows a Yee-grid (see Fig. 1), where the components E_z and $H_{x/y}$ are displaced by $\Delta x/2$ and $\Delta y/2$ in space and $\Delta t/2$ in time. This naturally yields the validity of Eqs. (3) and (4). Replacing all partial derivations with finite differences in Eqs. (5)–(7), we can deduce update rules for the electric and magnetic fields in space and time. For the electric field, the update-expression reads

$$\begin{aligned}
 E_z(m, n, q + 1) = & \frac{1 - \frac{\sigma\Delta t}{2\epsilon}}{1 + \frac{\sigma\Delta t}{2\epsilon}} E_z(m, n, q) \\
 & + \frac{1}{1 + \frac{\sigma\Delta t}{2\epsilon}} \left(\frac{\Delta t}{\epsilon\Delta x} \left\{ H_y\left(m + \frac{1}{2}, n, q + \frac{1}{2}\right) \right. \right. \\
 & \left. \left. - H_y\left(m - \frac{1}{2}, n, q + \frac{1}{2}\right) \right\} \right. \\
 & \left. - \frac{\Delta t}{\epsilon\Delta y} \left\{ H_x\left(m, n + \frac{1}{2}, q + \frac{1}{2}\right) \right. \right. \\
 & \left. \left. - H_x\left(m, n - \frac{1}{2}, q + \frac{1}{2}\right) \right\} \right). \quad (8)
 \end{aligned}$$

Nearly identical expressions can be found for H_x and H_y , respectively. A detailed derivation can be found in the literature.^{7,33} The idea of the FDTD is now to evolve the fields in a leap-frog-scheme. To be specific, the spatial differences from the magnetic field allow to advance the electric field in time. Afterward, those spatial differences from the electric field allow advancing the magnetic field in time. To solve the PDE, we have to impose some boundary conditions and potentially some perfectly matched layers, i.e., an impedance matched material that absorbs incoming radiation without any reflection. Such a material is necessary to imitate the infinite open space on a finite grid. To execute an actual simulation, we also need to define sources of electromagnetic radiation. We provide more details on these aspects in the section titled RESULTS since they are related to the training data generation.

B. Graph representation

As sketched in Sec. II A, the whole FDTD formalism is based on a local and iterative updating scheme. That is why the use of

graph neural networks appears very suitable to treat this problem. GNNs rely on the information exchange of neighboring nodes, which is ideal to capture the locality of the updating procedure. Moreover, FDTD works independently of the domain size, which can be translated to a GNN working on arbitrary input graphs.

The purpose of this section is a brief introduction to graph representations and graph neural networks; thus, it becomes clear to the reader how and why GNNs are potentially useful for FDTD computations.

A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ comprises a set of nodes or vertices $\mathcal{V} = v_1, v_2, \dots$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We consider here only undirected graphs. Therefore, every edge represents an undirected link between two graph nodes $e_{ij} = (v_i, v_j) \in \mathcal{E}$. Both nodes and edges hold a set of attributes, the node features $\mathbf{h}_v \in \mathbb{R}^D$ and edge features $\mathbf{h}_e \in \mathbb{R}^C$, respectively. In general, the number of features for each node or edge can vary in the so-called heterogeneous graphs. However, in our case, we have a fixed number of features, making our graphs homogeneous. Additionally, we assign spatial positions $\mathbf{p}_i = (p_{i,x}, p_{i,y})$ of the nodes that coincide with the simulation coordinates.

Graphs are very suitable to represent unstructured data, such as networks, molecules, or meshes.^{22–24} For the purpose of field propagation, we represent the electromagnetic field and material distribution as graphs. To keep things simple in this initial work, we concentrate here on dielectric materials only, for which it is sufficient to describe them using a spatially resolved dielectric function only. We can think about two possible graphs.

One approach is a direct representation of the Yee-grid; see Fig. 1. Here, the node features $\mathbf{h}_v \in \mathbb{R}^2$ encode the electric field $E_z(\mathbf{r}, t_0)$ and the permittivity $\epsilon(\mathbf{r})$. In extension, the edge features $\mathbf{h}_e \in \mathbb{R}^4$ comprise the magnetic field components $H_{x,y}(\mathbf{r}, t_0)$ and the relative Cartesian vector r_{ij} of linked nodes,

$$r_{ij} = (p_{j,x} - p_{i,x}, p_{j,y} - p_{i,y}). \quad (9)$$

Another approach is a meshing of the domain [see Fig. 1(b)] with equivalent edge and node features. While, technically, there is no distinction between a regular grid and a mesh at the level of the graph, it would make a tremendous difference at the level of the FDTD. In the FDTD, frequently, a regular grid is used, which causes a stair-case approximation of the surface of rounded objects. This

approximation can lead to computational artifacts that can be avoided with such a mesh.

In both cases, the choice of resolution is important. It does not have to be as fine as it is chosen for the FDTD simulation, but a too-low resolution leads to poor results. This is not necessarily caused by bad predictions but rather by a poor interpolation of the predictions to achieve the fields at all points in space inside the domain. Overall, the resolution is a trade-off between exceedingly large graphs, which slows down computation, and too less nodes to achieve accurate results at all points in space. In this work, we use a resolution that is found by manually tuning and choosing a sweet spot.

C. Graph neural networks

After having discussed the classical FDTD algorithm used to advance electromagnetic fields in space and time and having discussed how to represent the discretized Maxwellian problem in terms of a graph, we provide the following short introduction to graph neural networks. In short, the purpose of these networks will be to learn how to propagate a given input field in space and time.

A graph neural network (GNN) is an artificial neural network (ANN) that works on graphs. In contrast to conventional ANNs, such as linear or convolutional networks, a GNN does not rely on a specific shape and size of the input data. It works on any graph, independent of its number of nodes, edges, or connectivity and yields permutation invariance. A GNN can be interpreted as a graph operator \mathcal{F} that acts on a graph \mathcal{G} and returns a new graph \mathcal{G}' with updated features and potentially new nodes and edges. GNNs can perform tasks on different levels, specifically node-level, edge-level, and graph-level. We want to predict the electric field at each point in space after a fixed time step. This coincides with a node-level regression task since these encode the electric field in space. The graph that our network takes corresponds to the field at a certain moment in time, and afterward, it returns the field after a time step.

Our network comprises several components, depicted in Fig. 2. First, we have two encoders that lift the graph's node and edge features in a high-dimensional embedding space. Both encoders are fully connected feedforward networks of three linear layers with ReLU activation. The number of features increases with every MLP layer to 32, 64, and 128. Node and edge features must have the same dimension to enable the following generalized graph convolution (GCN) layers.³⁴ To perform the graph convolution, each node v_i gathers edge and node features of every neighboring node $v_j : j$

$\in \mathcal{N}(i)$ and connecting edge e_{ij} , the so-called *message passing*. Subsequently, the messages get aggregated (*message aggregation*) and added to the original node features. Finally, a multi-layer perceptron (MLP) processes the features. With that, we achieve updated node features v'_i . The whole operation reads

$$v'_i = MLP(v_i + AGG(\{ReLU(v_j + e_{ij}) + \varepsilon : j \in \mathcal{N}(i)\})), \quad (10)$$

where ε is a small positive parameter to ensure non-zero division in the aggregation scheme, for which we use max aggregation. Additionally, we use a layer normalization before non-linear activation and we add residual connections after every GCN layer. The message passing allows information exchange of connected neighbors, which we use to adapt the updating scheme performed in the Yee grid. However, since we use several GCN layers, we achieve information exchange not only of directly connected but also of further away nodes. Finally, we have a single decoder, another three-layered MLP, that projects the resulting features of each node back to a single value, the updated electric field. Please note that the whole operation on the input graph \mathcal{G} is independent of the amount of nodes and edges, only the number of input features is fixed. In general, this is one of the main strengths of GNNs, which we exploit to treat different domain sizes with the same model.

We trained GNNs with different hyperparameters to find a suitable configuration. However, we observed only minor differences in the final performance; thus, a qualitative reproduction of our results can also be achieved with different parameters. In our experiments, we found that a minimum number of trainable parameters of $\sim 300\,000$ is needed.

III. RESULTS

We train different GNNs to evolve a given electromagnetic field distribution $E_z(\mathbf{r}, t_0), H_{x,y}(\mathbf{r}, t_0)$ for a distinct time step Δt . We follow two different approaches for this purpose. First, we train different GNNs to perform a single time step in electric field prediction, given a comparatively low spatial resolution. We test and evaluate this for both grid and meshing. Second, we want to be able to predict the full field. Therefore, we keep a high temporal and spatial resolution and deduce the magnetic fields from predicted electric fields following the FDTD propagation procedure. The training is performed in parallel on four NVIDIA A100 Tensor Core graphics

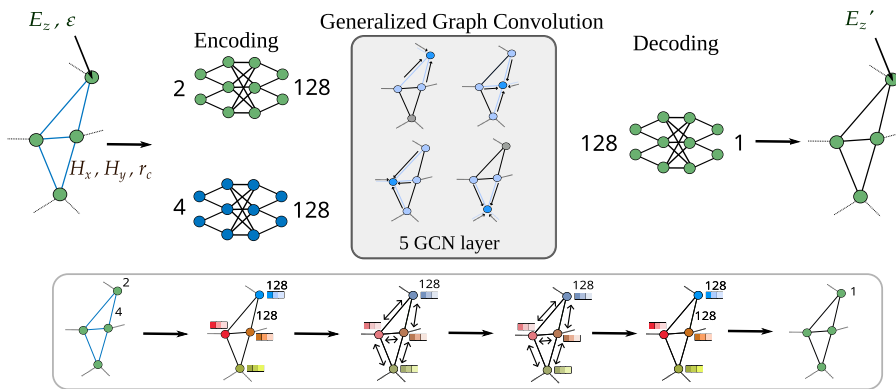


FIG. 2. GNN structure. Encoding of edge and node features followed by generalized graph convolution. Finally, decoding back to the electromagnetic field at $t = t_1$.

processing units (GPUs) with 40 GB RAM each. The training time varies, depending on model specifications, between 40 minutes and up to 4 hours.

We use the open-source Python FDTD-implementation *meeep*³⁵ for data generation and testing. We consider a plane wave source at $\lambda = 1 \mu\text{m}$ and arbitrarily shaped non-magnetic, non-conducting, non-dispersive, dielectric 2D scatterers of $\varepsilon = 2.1025$ in vacuum. The domain size varies between 1 and 3 μm side length with perfectly matched layers as boundary conditions. We chose a spatial resolution of 60 pixels/wavelength to ensure physically correct results. The source is a time-harmonic plane wave propagating along the $+x$ -direction into the considered domain. This comparably basic setting is most suitable for initial investigations. However, the method is not restricted to a specific source or material properties. An expansion to dispersive materials and different sources is possible.

A. Electric field prediction

To evolve the electric field only for a fixed time step, we can use a low spatial resolution and comparatively large time step $\Delta t = \frac{\lambda}{4c}$. This Δt coincides with 15 high resolution time steps in *meeep*. It is possible to choose such a high time step since the underlying training data were created with a high temporal resolution. Thus, it is physically accurate. The model then learns from correct data to adapt larger time steps. We construct graphs as structured grids, similar to the Yee grid, and meshes. In this case, edge features include both H_x and H_y values.

The training set comprises 125 different shapes, sampled at 20 different time steps of discretization $\Delta t = \frac{\lambda}{4c}$. The shapes are generated from random noise after applying a Gaussian filter. Starting from the initial field distribution, either a structured grid or a meshed graph is constructed; see Fig. 1. For meshing, we use the Python library *meshpy*³⁶ to create a triangular mesh. We control the number of mesh points by restricting the volume size. The domain

edges and the scatterer boundary are added manually. Subsequently, the GNN performs the temporal evolution for Δt , leading to a transformed graph \mathcal{G}' . The loss function compares the ground truth result of FDTD with the node features of \mathcal{G}' and computes the mean squared error (MSE),

$$\text{MSE}(\mathcal{G}, \mathcal{G}') = \frac{1}{N_v} \sum_i^{N_v} (h_{v,i} - \hat{h}_{v,i})^2 \quad (11)$$

$$= \frac{1}{N_v} \sum_i^{N_v} (E_{z,i} - \hat{E}_{z,i})^2. \quad (12)$$

Here, the FDTD ground truth values are interpolated in the case of meshed graphs to get the field values at every node position. The whole dataset is split into 2000 and 500 samples for training and validation, respectively. We use the optimizer ADAM³⁷ and a learning rate scheduler, which reduces the learning rate by a factor of 0.5 if no further improvement of the validation loss is observed within 10 epochs. We train for a total of 200 epochs and observe the convergence of validation and training loss.

To achieve the resulting field at every point in space, not only at the node positions, we linearly interpolate the nodes to a grid.

We test the resulting models on 20 different shapes and time steps and observe excellent agreements of the predicted fields with GNN and FDTD simulations. In the case of square grid graphs, we reach a mean squared error of 7.8×10^{-4} , whereas meshing results in 4.4×10^{-4} . Figure 3 shows two examples for each grid (a) and meshing (b). Here, the left figure shows the initial state. The following figure, i.e., the second to left, shows the predicted field after a time step $\Delta t = \frac{\lambda}{4c}$ with the GNN. The third figure from the left shows the predicted field values as computed with FDTD for reference purposes. Both predictions from the GNNs considering grids and meshes are in excellent agreement with the ground truth FDTD simulations. To quantify this agreement, we evaluate the normed

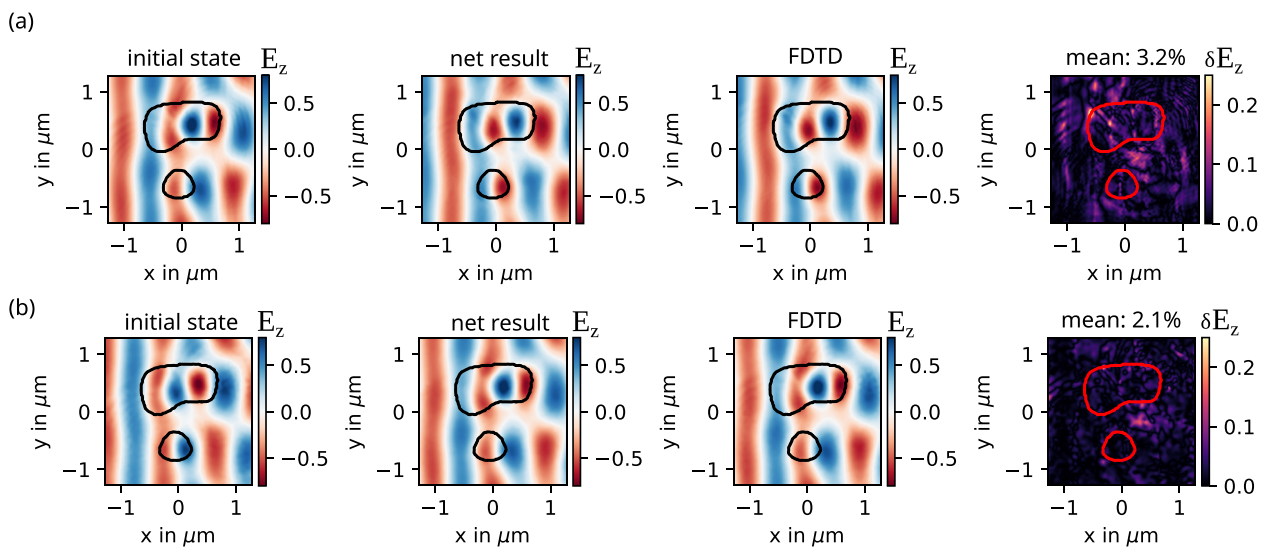


FIG. 3. Example prediction with grid graphs (a) and meshing (b). Both yield only small deviations from the actual FDTD calculations in the order of a few percent. The presented mean normed relative error follows Eq. (13).

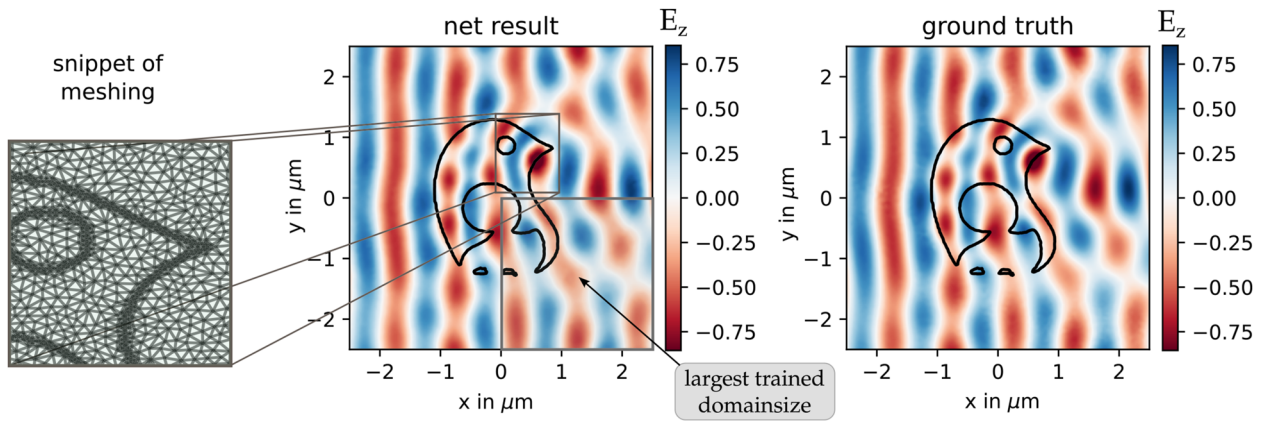


FIG. 4. Extrapolation to larger domain sizes and arbitrary scatterer shape in meshing. The left panel shows the initial, i.e., input, field. The largest trained domain size is marked in gray. The ground truth values are computed with *mEEP*, and the respective mean normed relative error is at 1.4%.

relative error of the GNN predicted field $E_z(\mathbf{r})$ and FDTD computed field $E_{z,0}(\mathbf{r})$ at $t_0 + \Delta t$,

$$\delta(\mathbf{r}, E_z, E_{z,0}) = \frac{|E_z(\mathbf{r}) - E_{z,0}(\mathbf{r})|}{|E_z(\mathbf{r})| + |E_{z,0}(\mathbf{r})|}, \quad (13)$$

where $\overline{|E_{z,0}|}$ is the average absolute value of the correct field with respect to r . The error is shown in the right of Figs. 3(a) and 3(b). The prediction is slightly worse for smaller propagation times due to transitional temporal effects arising from the source initiation. These effects are particularly hard to capture with meshing with coarser elements in the spatial domain. This results in interpolation errors when converting the graph representation to fields. However, the prediction with grids of resolution 30 pixels/wavelength covers the effects quite well.

Now, we test the trained GNNs on domain sizes outside the trained range of 1–3 μm . Figure 4 shows an example with 5 μm side length in meshing. The mean value for this example coincides with $\approx 1.4\%$ deviation, which is very low. In addition, the shape of the scatterer is rather specific compared to the ones used in training. This shows that our model learned to adapt the propagation procedure of a given electromagnetic field. It can extrapolate very well to larger domain sizes and even complex shapes not seen in training. We achieve similar results with grid graphs.

B. Full field prediction

Instead of only predicting $E_z(\mathbf{r}, t_0 + \Delta t)$, we want to train models that are able to compute the full electromagnetic field. For that, we add two feature channels to the node decoder to enable direct prediction of $H_{x,y}(\mathbf{r}, t_0 + \Delta t)$ for grid graphs. The used network architecture and hyperparameters are the same, besides the additional decoder channels. It is worth mentioning that an alternative approach is a second pure edge encoder. However, we stick to a single decoder for the full field. Again, we test our results on 20 unknown test shapes at 20 time steps and achieve a mean squared error of 4.4×10^{-4} on H_x and 5.3×10^{-4} on H_y components. The

extrapolation on larger domain sizes works very well and shows the same accuracy as pure E_z predictions.

Alternatively, we can directly deduce the magnetic fields from E_z . Since the network input comprises the full field, we can follow the FDTD propagation procedure. Therefore, the graph construction changes slightly. We stick to grid graphs of the same high resolution as the simulation, namely 60 pixels/wavelength. Adapting to the Yee grid, we only use H_x fields for vertical edges and H_y fields for horizontal ones. Additionally, we sample E_z and $H_{x,y}$ at different times, shifted by $\Delta t/2$. From the predicted field E_z , we compute $H_{x,y}$ following the FDTD formalism. To achieve accurate results for $H_{x,y}$, we add a *Maxwell loss* in training,

$$\begin{aligned} \text{Loss}(E_z, \hat{E}_z) &= \text{Loss}_{\text{data}} + \text{Loss}_{\text{Maxwell}} \\ &= \frac{1}{N_v} \sum (E_z - \hat{E}_z)^2 \\ &\quad + \alpha \frac{1}{N_e} \sum (H_{x,y} - \hat{H}_{x,y}(\hat{E}_z))^2. \end{aligned} \quad (14)$$

Here, $H_{x,y}$ is the ground truth magnetic field from the simulation at $t = t_1 + 1/2\Delta t$ and $\hat{H}_{x,y}(\hat{E}_z)$ is the field resulting from the network output \hat{E}_z plugged in the FDTD update equation. By that, we compare not only the electric field predictions to ground truth values in training but also the derived magnetic fields. Additionally, we introduce α as a parameter to control the strength of the Maxwell loss. Including Maxwell's equation in the loss has been applied successfully before, yielding performance improvements.^{38–40}

Figure 5 shows the results for a single test case. The mean error on the magnetic field components is slightly higher than on the electric field. In any case, the deviation is in the order of only a few percent.

In principle, the prediction of the whole field allows passing the output back to the input of the GNN. Unfortunately, we observe a fast error accumulation leading to very bad results after just a few iterations, independent of the choice of the $H_{x,y}$ prediction. This problem is known²⁴ and can be treated or partially solved by training

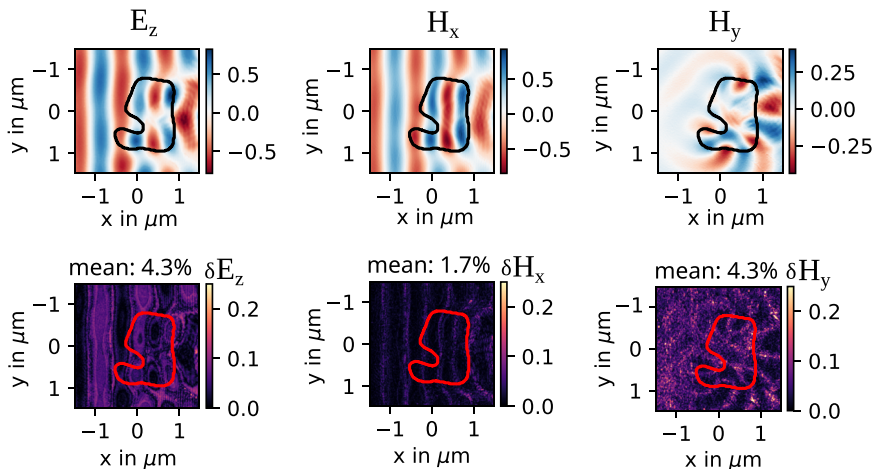


FIG. 5. Full field predictions comprising E_z , H_x , and H_y . The magnetic field components are computed using the predicted E_z following the FDTD formalism. The deviations are low in the order of a few percent.

on noisy input data or including several iterations in the loss function.

IV. CONCLUSION AND OUTLOOK

We demonstrated that a GNN is capable of propagating electromagnetic fields for a fixed time step, given the initial field distribution. After the training is finished, the model is able to interpolate very well for unknown scenarios with parameters similar to the trained ones. Moreover, the GNN can also extrapolate on very large domains and very specific scatterer shapes. By that, we proved to overcome one of the main issues of conventional ANN architectures: fixed input size, i.e., domain size. The very good performance on many different scatterer shapes implies that the model actually learned to propagate fields by adapting the physical propagation procedure. Despite the high prediction accuracy, the computation of multiple time steps by iteratively feeding the output of the network back as input was not successful. However, there exist approaches to circumvent the problem of error accumulation, which we have outlined in the section titled RESULTS.

Yet, the use of GNNs to perform a single time step is not advantageous to proper FDTD simulations, since at least in 2D, those are cheap and fast. However, in the case of 3D scenarios with more complex materials and scatterer shapes, the use of a GNN potentially saves time and computational efforts. Along these lines, our method could particularly be suitable for dispersive materials, since these are more costly and time-consuming in FDTD computations. Additionally, a generalization to different sources may be possible by adding the respective information to the input graph. We also showed that the model can work with meshing, which is not possible in a pure FDTD formulation. These findings support that GNNs can also be used for more complex mesh-based simulations, for example finite element methods, which are, in general, very demanding. Additionally, the graphs we process with the GNNs are very large with only a few connections to the nearest neighbors. This results in a very sparse matrix representation. Unfortunately, common graph learning frameworks only support static representations of these matrices; thus, a lot of unnecessary computations are performed. However,

there are already efforts to include the treatment of sparse matrices,⁴¹ which would potentially speed up computations with graphs similar to ours. In the long run, we foresee hybrid implementations where the time stepping of fields is alternatively done in a classical FDTD scheme and using GNNs. This will save computational resources and time and will speed up simulations.

ACKNOWLEDGMENTS

L.K. acknowledges the support by the NHR@KIT program and the Karlsruhe School of Optics and Photonics. L.K. acknowledges the Jülich Supercomputing Centre, specifically the JUWELS Booster for training and validation. C.R. acknowledges the support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy via the Excellence Cluster 3D Matter Made to Order (Grant No. EXC-2082/1-390761711). T.R. is supported by Estonian Research Council Grant (No. PSG716). We acknowledge support by the KIT-Publication Fund of the Karlsruhe Institute of Technology.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

L. Kuhn: Conceptualization (equal); Formal analysis (equal); Investigation (equal); Writing – original draft (equal). **T. Repän:** Supervision (equal); Validation (equal); Writing – review & editing (equal). **C. Rockstuhl:** Project administration (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

- ¹A. F. Koenderink, A. Alù, and A. Polman, "Nanophotonics: Shrinking light-based technology," *Science* **348**(6234), 516–521 (2015).
- ²F. Monticone and A. Alù, "Metamaterial, plasmonic and nanophotonic devices," *Rep. Prog. Phys.* **80**(3), 036401 (2017).
- ³C. Lienau, M. A. Noginov, and M. Lončar, "Light–matter interactions at the nanoscale," *J. Opt.* **16**(11), 110201 (2014).
- ⁴A. V. Lavrinenko, J. Lægsgaard, N. Gregersen, F. Schmidt, and T. Søndergaard, *Numerical Methods in Photonics* (CRC Press, 2018).
- ⁵B. Gallinet, J. Butet, and O. J. F. Martin, "Numerical methods for nanophotonics: Standard problems and future challenges," *Laser Photonics Rev.* **9**(6), 577–603 (2015).
- ⁶P. Lalanne, W. Yan, K. Vynck, C. Sauvan, and J.-P. Hugonin, "Light interaction with photonic and plasmonic resonances," *Laser Photonics Rev.* **12**(5), 1700113 (2018).
- ⁷A. Taflove, S. C. Hagness, and M. Piket-May, "Computational electromagnetics: The finite-difference time-domain method," in *The Electrical Engineering Handbook* (Elsevier, Inc., 2005), pp. 629–670.
- ⁸V. Jovanov, U. Palanchoke, P. Magnus, H. Stiebig, J. Hüpkens, P. Sicanugrist, M. Konagai, S. Wiesendanger, C. Rockstuhl, and D. Knipp, "Light trapping in periodically textured amorphous silicon thin film solar cells using realistic interface morphologies," *Opt. Express* **21**(S4), A595–A606 (2013).
- ⁹A. Vaccari, L. Cristoforetti, A. C. Lesina, L. Ramunno, A. Chiappini, F. Prudenzano, A. Bozzoli, and L. Calliari, "Parallel finite-difference time-domain modeling of an opal photonic crystal," *Opt. Eng.* **53**(7), 071809 (2014).
- ¹⁰J. Park, S. Kim, D. W. Nam, H. Chung, C. Y. Park, and M. S. Jang, "Free-form optimization of nanophotonic devices: from classical methods to deep learning," *Nanophotonics* **11**(9), 1809–1845 (n.d.).
- ¹¹W. Ma, Z. Liu, Z. A. Kudyshev, A. Boltasseva, W. Cai, and Y. Liu, "Deep learning for the design of photonic structures," *Nat. Photonics* **15**(2), 77–90 (2021).
- ¹²Z. A. Kudyshev, A. V. Kildishev, V. M. Shalaev, and A. Boltasseva, "Machine learning–assisted global optimization of photonic devices," *Nanophotonics* **10**(1), 371–383 (2021).
- ¹³T. Repän, V. Ramakrishna, and C. Rockstuhl, "Artificial neural networks used to retrieve effective properties of metamaterials," *Opt. Express* **29**(22), 36072–36085 (2021).
- ¹⁴Y. Tang, K. Kojima, T. Koike-Akino, Y. Wang, P. Wu, Y. Xie, M. H. Tahersima, D. K. Jha, K. Parsons, and M. Qi, "Generative deep learning model for inverse design of integrated nanophotonic devices," *Laser Photonics Rev.* **14**(12), 2000287 (2020).
- ¹⁵P. R. Wiecha, A. Arbouet, C. Girard, and O. L. Muskens, "Deep learning in nano-photonics: Inverse design and beyond," *Photonics Res.* **9**(5), B182–B200 (2021).
- ¹⁶S. So, T. Badloe, J. Noh, J. Bravo-Abad, and J. Rho, "Deep learning enabled inverse design in nanophotonics," *Nanophotonics* **9**(5), 1041–1057 (2020).
- ¹⁷L. Kuhn, T. Repän, and C. Rockstuhl, "Inverse design of core-shell particles with discrete material classes using neural networks," *Sci. Rep.* **12**(1), 19019 (2022).
- ¹⁸I. Sajedian, T. Badloe, and J. Rho, "Optimisation of colour generation from dielectric nanostructures using reinforcement learning," *Opt. Express* **27**(4), 5874–5883 (2019).
- ¹⁹C. Majorel, C. Girard, A. Arbouet, O. L. Muskens, and P. R. Wiecha, "Deep learning enabled strategies for modeling of complex aperiodic plasmonic metasurfaces of arbitrary size," *ACS Photonics* **9**(2), 575–585 (2022).
- ²⁰H. M. Yao and L. J. Jiang, "Machine learning based neural network solving methods for the FDTD method," in *2018 IEEE International Symposium on Antennas and Propagation and USNC/URSI National Radio Science Meeting* (IEEE, 2018), pp. 2321–2322.
- ²¹D. Zhu, Q. Zhao, Y. Bo, W. Chen, and L. Yang, "Application of deep learning in FDTD method," in *2021 13th International Symposium on Antennas, Propagation and EM Theory (ISAPE)* (IEEE, 2021), Vol. 1, pp. 1–3.
- ²²O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer, "A compact review of molecular property prediction with graph neural networks," *Drug Discovery Today: Technol.* **37**, 1–12 (2020).
- ²³H. Ma, Y. Bian, R. Yu, W. Huang, T. Xu, W. Xie, G. Ye, and J. Huang, "Multi-view graph neural networks for molecular property prediction," [arXiv:2005.13607](https://arxiv.org/abs/2005.13607) (2020).
- ²⁴P. Tobias, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning mesh-based simulation with graph networks," [arXiv:2010.03409](https://arxiv.org/abs/2010.03409) (2020).
- ²⁵F. De Avila Belbute-Peres, T. Economou, and Z. Kolter, "Combining differential PDE solvers and graph neural networks for fluid flow prediction," in *Proceedings of the 37th International Conference on Machine Learning, Proceedings of Machine Learning Research*, edited by H. Daumé III and A. Singh (PMLR, 2020), Vol. 119, pp. 2402–2411.
- ²⁶K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision GNN: An image is worth graph of nodes," [arXiv:2206.00272](https://arxiv.org/abs/2206.00272) (2022).
- ²⁷X. Ju, S. Farrell, P. Calafiura, D. Murnane, Prabhat, L. Gray, T. Klijnsma, K. Pedro, G. Cerati, J. Kowalkowski, G. Perdue, P. Spentzouris, N. Tran, J.-R. Vlimant, A. Zlokapá, J. Pata, M. Spiropulu, S. An, A. Adam, J. Hewes, A. Tsaris, K. Terao, and T. Usher, "Graph neural networks for particle reconstruction in high energy physics detectors," [arXiv:2003.11603](https://arxiv.org/abs/2003.11603) (2020).
- ²⁸J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Mach. Learn.: Sci. Technol.* **2**(2), 021001 (2020).
- ²⁹Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Graph kernel network for partial differential equations," [arXiv:2003.03485](https://arxiv.org/abs/2003.03485) (2020).
- ³⁰Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar, "Multipole graph neural operator for parametric partial differential equations," *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020*.
- ³¹N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Learning maps between function spaces," [arXiv:2108.08481](https://arxiv.org/abs/2108.08481) (2021).
- ³²J. B. Schneider, "Understanding the finite-difference time-domain method," School of Electrical Engineering and Computer Science, Washington State University (2010), 28.
- ³³A. Taflove, A. Oskooi, and S. G. Johnson, *Advances in FDTD Computational Electrodynamics: Photonics and Nanotechnology* (Artech house, 2013).
- ³⁴G. Li, C. Xiong, A. Thabet, and B. Ghanem, "All you need to train deeper GCNs," [arXiv:2006.07739](https://arxiv.org/abs/2006.07739) (2020).
- ³⁵A. F. Oskooi, D. Roundy, M. Ibanescu, P. Bermel, J. D. Joannopoulos, and S. G. Johnson, "MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method," *Comput. Phys. Commun.* **181**(3), 687–702 (2010).
- ³⁶I. Steinbrecher and A. Popp, "MeshPy: A general purpose 3D beam finite element input generator," <https://compsim.gitlab.io/codes/meshpy>, 2021.
- ³⁷D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).ISO 690 (2014).
- ³⁸M. Chen, R. Lupoiu, C. Mao, D.-H. Huang, J. Jiang, P. Lalanne, and J. A. Fan, "High speed simulation and freeform optimization of nanophotonic devices with physics-augmented deep learning," *ACS Photonics* **9**(9), 3110–3123 (2022).
- ³⁹Z. Fang and J. Zhan, "Deep physical informed neural networks for metamaterial design," *IEEE Access* **8**, 24506–24513 (2020).
- ⁴⁰P. Zhang, Y. Hu, Y. Jin, S. Deng, X. Wu, and J. Chen, "A Maxwell's equations based deep learning method for time domain electromagnetic simulations," *IEEE J. Multiscale Multiphys. Comput. Tech.* **6**, 35–40 (2021).
- ⁴¹S. Qiu, L. You, and Z. Wang, "Optimizing sparse matrix multiplications for graph neural networks," *Languages and Compilers for Parallel Computing* (Springer International Publishing, 2022), pp. 101–117.