Machine Learning Advances in Computational Electromagnetics

Robert Lupoiu and Jonathan A. Fan

Department of Electrical Engineering, Stanford University, Stanford, CA, USA

August 2023

Abstract

This chapter explores the emerging topic of computational electromagnetics (CEM) augmentation using machine learning techniques, which has disruptive potential due to its ability to evaluate solutions to CEM problems with unprecedented speeds. We discuss how recent developments in machine learning hardware and software can enhance conventional CEM techniques and enable specialized CEM algorithms using deep neural networks. We also review how generalized surrogate electromagnetic solvers can be realized by the training of deep convolutional neural networks using simulated data, Maxwell's equations, or both. The chapter concludes with a perspective on the most promising research opportunities in this highly fertile area of investigation.

Keywords— Computational electromagnetics, Electromagnetic surrogate solvers, Physics-informed neural networks, WaveY-Net, Inverse design, Simulation, Neural networks, Deep learning

1 Introduction

The field of computational electromagnetics (CEM) involves the development of algorithms that can numerically model Maxwell's equations with high accuracy. It initiated during the first computing revolution in the 1960's, during which a wide range of partial differential equation solvers were developed across many fields of science and engineering, including structural analysis, fluid flow, heat transfer, and seismology. [1, 2, 3] Conventionally, CEM algorithms have been developed to be general, with the capability to solve electromagnetics problems with few to no prior assumptions about the problem details. Accurate solutions to Maxwell's equations are achieved with quantitative convergence criteria, and rigorous frameworks have been developed to quantify the trade-off between numerical approximation and solution accuracy. To date, these methods have been developed to a high level of maturity. [4, 5, 6, 7]

The machine learning revolution of the last decade has presented entirely new opportunities for CEM, with the potential to dramatically accelerate the simulation and design of electromagnetic systems. Electromagnetic systems posed for CEM modelling are amenable to being framed in the context of machine learning because they are not arbitrary, but can instead be described in terms of application domains that each involve highly limited geometric layouts, material libraries, and electromagnetic objectives. For example, chip-based silicon photonic circuits typically involve the patterning of thin films consisting of a handful of materials and limited operating wavelengths. Structures within a technological domain therefore use strongly related physics that can be "learned" and generalized using machine learning models. CEM is particularly well suited for deep learning because conventional solvers can be readily used to curate large batches of training data, and they can also be used to quantify accuracy and convergence of electromagnetic field solutions.

In this chapter, we will discuss emergent concepts at the interface of the data sciences and conventional CEM algorithms. We will summarize the mechanics of conventional CEM algorithms, including those based on finite differences, finite elements, and the method of moments. These algorithms provide a foundation for understanding how to numerically discretize a CEM problem and find accurate solutions with rigorous convergence criteria. We will then discuss how machine learning algorithms can augment CEM solvers. Some of these concepts involve deep networks that are trained end-to-end to serve as surrogate CEM solvers, while others are embedded within the framework of conventional algorithms to leverage the numerical stability and accuracy of these methods.

2 Conventional electromagnetic simulation techniques

This section covers an introduction to three conventional CEM techniques that are the basis for most academically and commercially used CEM software: the Finite Difference (FD) Method, Finite Element Method (FEM), and the Method of Moments (MoM). The machine learning methods surveyed in sections 3-5 either directly integrate with these methods or exploit insights provided by their formulations. The finite difference methods involve approximating the differential form of Maxwell's equations with the finite difference formalism, and they include time domain (FDTD) and frequency domain (FDFD) formulations. [4, 6] FEM is a frequency domain solver in which the volumetric domain is subdivided into a generally irregular grid of voxels that describe electromagnetic fields as a combination of primitive basis functions. [7, 4] MoM is a frequency domain solver based on the integral form of Maxwell's equations and is used to evaluate currents and fields at the surfaces of homogeneous media.[4, 6, 5] Each technique has its own set of strengths and weaknesses regarding computational complexity, accuracy, and scaling. The proper choice of a solver for a given problem strongly depends on problem geometry, source configuration, and boundary conditions.

2.1 Finite Difference Frequency (FDFD) and Time (FDTD) Domain Solvers

With the finite difference methods, electromagnetic fields are discretized on a regular grid of voxels, and Maxwell's equations in the differential form are formulated to relate these discrete electric and magnetic field components. The discrete electric and magnetic field positions are defined on a "Yee" grid (Fig. 1), which staggers the field component positions and ensures that the differential relationships between electric and magnetic fields as defined by Maxwell's equations are numerically stable. [4] The Yee grid was introduced by Kane Yee in the 1960's in his original formulation of the FDTD method, [8] where he showed that field staggering mitigates the significant discretization errors inherent to collocated grid representations. The Yee-based representations of Ampere's law and Faraday's law in the frequency and time domains, with derivatives approximated using first-order central difference expressions, are summarized in Table 1. These expressions for FDTD and

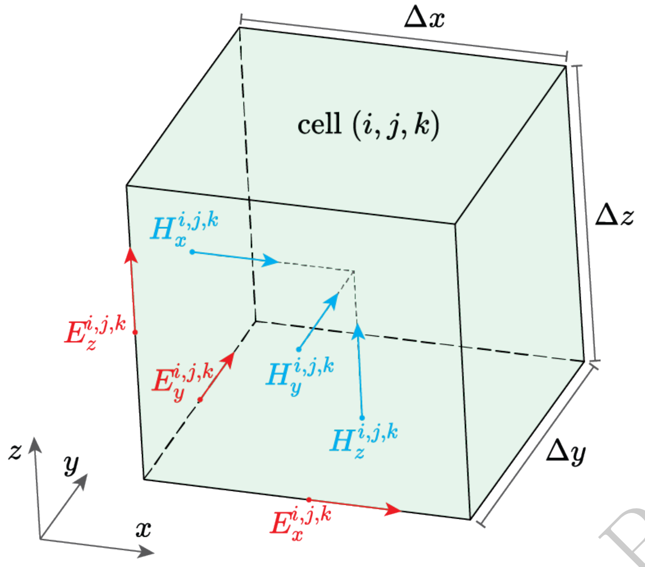

Figure 1: The structuring of the six field components within a single cell of the Yee grid formalism. The electric and magnetic field components are not collocated, but rather staggered and non-overlapping, which enforces computational stability.

FDFD are very similar, with the key difference being the assumption of harmonic wave phenomena with frequency ω in the frequency domain formulation.

In FDTD, Ampere's law and Faraday's law are formulated as update equations and are iteratively evaluated in a time-marching process with time step Δt to compute the time evolution of electromagnetic fields across the entire domain. Consider the update of field components at cell position (i, j, k) at time step t, in the absence of magnetic or current sources. First, using the FDTD formulation of Faraday's Law (Table 1), the magnetic field components are calculated at time step $t + \frac{\Delta t}{2}$ using the previously-determined magnetic field components at time step $t - \frac{\Delta t}{2}$ and the electric field components from the current time step, t. This update equation for the H_x field component, for example, is readily evaluated as:

$$H_x^{i,j,k}(t + \frac{\Delta t}{2}) = \frac{\Delta t}{\mu_x^{i,j,k}}(\frac{E_y^{i,j,k+1}(t) - E_y^{i,j,k}(t)}{\Delta z} - \frac{E_z^{i,j+1,k}(t) - E_z^{i,j,k}(t)}{\Delta y}) + H_x^{i,j,k}(t - \frac{\Delta t}{2}). \quad (1)$$

Update equations for the other field components have similar forms. Next, the electric field components are determined at time step $t + \Delta t$ using the FDTD formulation of Ampere's Law, using the newly-calculated $H(t + \frac{\Delta t}{2})$ and previously-calculated $E(t)$ field components.

While the FDTD approach to solving Maxwell's equations can apply in principle to arbitrary domains, there exist practical limitations in computational memory that limit the domain size and time step granularity. In order for the algorithm to be stable, the time step must be smaller than the stability bound, which is a function of wave velocity and voxel size. Domains featuring high refractive index media or small feature sizes therefore necessitate slow time stepping procedures. In addition, the requirement of voxels with regular shapes, such as cuboidal, means that curvilinear surfaces can be difficult to model unless the Yee grid is specified to be very fine, adding to further increased computer memory requirements and reduced Δt. Note that it is necessary to reduce the time step size for finer meshes in order to maintain the stability of the algorithm, as a consequence of the *Courant-Friedrichs-Levy (CFL)* stability condition. [9]

	Faraday's Law
Differential Continuous Time Domain Formulation	$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} - \mathbf{M}$ $\Longrightarrow \begin{cases} \frac{\partial E_z(t)}{\partial y} - \frac{\partial E_y(t)}{\partial z} = -\mu \frac{\partial H_x(t)}{\partial t} - M_x(t) \\ \frac{\partial E_x(t)}{\partial z} - \frac{\partial E_z(t)}{\partial x} = -\mu \frac{\partial H_y(t)}{\partial t} - M_y(t) \\ \frac{\partial E_y(t)}{\partial x} - \frac{\partial E_x(t)}{\partial y} = -\mu \frac{\partial H_z(t)}{\partial t} - M_z(t) \end{cases}$
FDFD Formulation	$\begin{cases} \frac{E_z^{i,j+1,k} - E_z^{i,j,k}}{\Delta y} - \frac{E_y^{i,j,k+1} - E_y^{i,j,k}}{\Delta z} = -i\omega\mu_x^{i,j,k} H_x^{i,j,k} - M_x^{i,j,k} \\ \frac{E_x^{i,j,k+1} - E_x^{i,j,k}}{\Delta z} - \frac{E_z^{i+1,j,k} - E_z^{i,j,k}}{\Delta x} = -i\omega\mu_y^{i,j,k} H_y^{i,j,k} - M_y^{i,j,k} \\ \frac{E_y^{i+1,j,k} - E_y^{i,j,k}}{\Delta x} - \frac{E_x^{i,j+1,k} - E_x^{i,j,k}}{\Delta y} = -i\omega\mu_z^{i,j,k} H_z^{i,j,k} - M_z^{i,j,k} \end{cases}$
FDTD Formulation	$\begin{cases} \frac{E_z^{i,j+1,k}(t) - E_z^{i,j,k}(t)}{\Delta y} - \frac{E_y^{i,j,k+1}(t) - E_y^{i,j,k}(t)}{\Delta z} = -\mu_x^{i,j,k} \frac{H_x^{i,j,k}(t+\frac{\Delta t}{2}) - H_x^{i,j,k}(t-\frac{\Delta t}{2})}{\Delta t} - M_x^{i,j,k}(t) \\ \frac{E_x^{i,j,k+1}(t) - E_x^{i,j,k}(t)}{\Delta z} - \frac{E_z^{i+1,j,k}(t) - E_z^{i,j,k}(t)}{\Delta x} = -\mu_y^{i,j,k} \frac{H_y^{i,j,k}(t+\frac{\Delta t}{2}) - H_y^{i,j,k}(t-\frac{\Delta t}{2})}{\Delta t} - M_y^{i,j,k}(t) \\ \frac{E_y^{i+1,j,k}(t) - E_y^{i,j,k}(t)}{\Delta x} - \frac{E_x^{i,j+1,k}(t) - E_x^{i,j,k}(t)}{\Delta y} = -\mu_z^{i,j,k} \frac{H_z^{i,j,k}(t+\frac{\Delta t}{2}) - H_z^{i,j,k}(t-\frac{\Delta t}{2})}{\Delta t} - M_z^{i,j,k}(t) \end{cases}$
	Ampere's Law
Differential Continuous Time Domain Formulation	$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J}$ $\Longrightarrow \begin{cases} \frac{\partial H_z(t)}{\partial y} - \frac{\partial H_y(t)}{\partial z} = \varepsilon \frac{\partial E_x(t)}{\partial t} + J_x(t) \\ \frac{\partial H_x(t)}{\partial z} - \frac{\partial H_z(t)}{\partial x} = \varepsilon \frac{\partial E_y(t)}{\partial t} + J_y(t) \\ \frac{\partial H_y(t)}{\partial x} - \frac{\partial H_x(t)}{\partial y} = \varepsilon \frac{\partial E_z(t)}{\partial t} + J_z(t) \end{cases}$
FDFD Formulation	$\begin{cases} \frac{H_z^{i,j+1,k} - H_z^{i,j,k}}{\Delta y} - \frac{H_y^{i,j,k+1} - H_y^{i,j,k}}{\Delta z} = i\omega\varepsilon_x^{i,j,k} E_x^{i,j,k} + J_x^{i,j,k} \\ \frac{H_x^{i,j,k+1} - H_x^{i,j,k}}{\Delta z} - \frac{H_z^{i+1,j,k} - H_z^{i,j,k}}{\Delta x} = i\omega\varepsilon_y^{i,j,k} E_y^{i,j,k} + J_y^{i,j,k} \\ \frac{H_y^{i+1,j,k} - H_y^{i,j,k}}{\Delta x} - \frac{H_x^{i,j+1,k} - H_x^{i,j,k}}{\Delta y} = i\omega\varepsilon_z^{i,j,k} E_z^{i,j,k} + J_z^{i,j,k} \end{cases}$
FDTD Formulation	$\begin{cases} \frac{H_z^{i,j+1,k}(t+\frac{\Delta t}{2}) - H_z^{i,j,k}(t+\frac{\Delta t}{2})}{\Delta y} - \frac{H_y^{i,j,k+1}(t+\frac{\Delta t}{2}) - H_y^{i,j,k}(t+\frac{\Delta t}{2})}{\Delta z} = \varepsilon_x^{i,j,k} \frac{E_x^{i,j,k}(t+\Delta t) - E_x^{i,j,k}(t)}{\Delta t} + J_x^{i,j,k}(t+\frac{\Delta t}{2}) \\ \frac{H_x^{i,j,k+1}(t+\frac{\Delta t}{2}) - H_x^{i,j,k}(t+\frac{\Delta t}{2})}{\Delta z} - \frac{H_z^{i+1,j,k}(t+\frac{\Delta t}{2}) - H_z^{i,j,k}(t+\frac{\Delta t}{2})}{\Delta x} = \varepsilon_y^{i,j,k} \frac{E_y^{i,j,k}(t+\Delta t) - E_y^{i,j,k}(t)}{\Delta t} + J_y^{i,j,k}(t+\frac{\Delta t}{2}) \\ \frac{H_y^{i+1,j,k}(t+\frac{\Delta t}{2}) - H_y^{i,j,k}(t+\frac{\Delta t}{2})}{\Delta x} - \frac{H_x^{i,j+1,k}(t+\frac{\Delta t}{2}) - H_x^{i,j,k}(t+\frac{\Delta t}{2})}{\Delta y} = \varepsilon_z^{i,j,k} \frac{E_z^{i,j,k}(t+\Delta t) - E_z^{i,j,k}(t)}{\Delta t} + J_z^{i,j,k}(t+\frac{\Delta t}{2}) \end{cases}$

Table 1: Faraday's Law and Ampere's Law expressed in the differential continuous time domain, FDFD, and FDTD formulations. The FD formulations use central finite difference derivative approximations.

In FDFD, steady state solutions to Maxwell's equations are directly evaluated within the full domain. In this formalism, the relationships between electric and magnetic fields at all voxels are expressed through Faraday's law and Ampere's law, which can be written in the form of vector and matrix relations. Faraday's law is:

$$\boldsymbol{C_e}\overrightarrow{\boldsymbol{e}} = -i\omega\boldsymbol{T_\mu}\overrightarrow{\boldsymbol{h}} - \overrightarrow{\boldsymbol{m}}. \tag{2}$$

$\overrightarrow{\boldsymbol{e}}$ and $\overrightarrow{\boldsymbol{h}}$ are vectors that include field values at every voxel in the domain, the matrix $\boldsymbol{C_e}$ includes derivative expressions that relate electric field values between neighboring voxels based on finite difference relations, $\boldsymbol{T_\mu}$ contains permeability values in the simulation domain as a function of position, and $\overrightarrow{\boldsymbol{m}}$ represents magnetic currents. Ampere's law is:

$$\boldsymbol{C_h}\overrightarrow{\boldsymbol{h}} = i\omega\boldsymbol{T_\varepsilon}\overrightarrow{\boldsymbol{e}} + \overrightarrow{\boldsymbol{j}}. \tag{3}$$

The matrix $\boldsymbol{C_h}$ includes derivative expressions that relate magnetic field values between neighboring voxels based on finite difference relations, $\boldsymbol{T_\varepsilon}$ contains permittivity values in the simulation domain as a function of position, and $\overrightarrow{\boldsymbol{j}}$ represents electric currents. These equations can be combined together to produce a wave equation in terms of electric or magnetic fields. For electric fields, this expression is:

$$(\boldsymbol{C_h}\boldsymbol{T_\mu}^{-1}\boldsymbol{C_e} - \omega^2\boldsymbol{T_\varepsilon})\overrightarrow{\boldsymbol{e}} = -i\omega\overrightarrow{\boldsymbol{j}} - \boldsymbol{C_h}\boldsymbol{T_\mu}^{-1}\overrightarrow{\boldsymbol{m}}. \tag{4}$$

This expression has the form of $\boldsymbol{A}\overrightarrow{\boldsymbol{x}} = \overrightarrow{\boldsymbol{b}}$, where $\boldsymbol{A}$ and $\overrightarrow{\boldsymbol{b}}$ contain known permittivity, permeability, and source information about the domain, and $\overrightarrow{\boldsymbol{x}}$ represents the unknown fields. The fields are then readily evaluated as $\overrightarrow{\boldsymbol{x}} = \boldsymbol{A}^{-1}\overrightarrow{\boldsymbol{b}}$.

From this formalism, we see that while FDTD and FDFD use similar book keeping tools in the form of the Yee grid, the algorithms use very different computational flows. While FDTD can be readily parallelized, FDFD involves the inversion of a large sparse matrix and thus cannot be directly parallelized. In terms of computational evaluation, there are no straightforward strategies to speed up FDTD simulations, which utilize update equations with basic algebraic relations. However, various mathematical tricks can be applied to FDFD to accelerate the inversion of sparse matrices, improving its computational scaling. In addition, advanced matrix inversion methods based on the generalized minimal residual method (GMRES) [10] or related Krylov subspace methods [11] can be preconditioned with prior knowledge of the system to further accelerate solution convergence.

An important consideration for both FD methods and CEM simulators more generally is the specification of proper boundary conditions along the simulation domain boundary. For bounded problems in which the fields properly terminate within the simulation domain, boundary conditions such as Perfect Electric Conductors (PECs) can be used. For unbounded problems involving field propagation to regions outside of the domain, Absorbing Boundary Conditions (ABCs) are required that absorb incoming waves without back-reflections, effectively simulating unbounded wave propagation. One highly effective ABC model is the perfectly matched layer (PML), which utilizes an artificial anisotropic absorbing material to convert propagating waves to exponentially decaying waves without backreflection. [12] PMLs are the *de facto* modern ABC because they can robustly suppress reflections from absorbed waves over a wide range of frequencies and incident angles. [13, 5] However, while PMLs are highly effective, they require thicknesses on the order of half a free space wavelength, thereby adding significant computational overhead to the wave simulation procedure.

2.2 The Finite Element Method (FEM)

The Finite Element Method (FEM) is the most widely used scientific computing technique for solving differential equations. It was initially developed by mechanical engineers to solve solid mechanics problems in the 1940s, and it became an integral tool for CEM in the 1980s with the advent of reliable ABCs. [5] In this subsection, we will focus on FEM concepts based on the weighted residual method, which is a standard formulation of the technique. [7] The steps for solving Maxwell's equations with this version of the FEM algorithm are delineated in sections 2.2.1-2.2.3.

2.2.1 Meshing

The solution domain, Ω, is first subdivided into generally irregularly-shaped voxels termed *elements*. For a 2D domain, these are commonly triangles or quadrilaterals, and for 3D domains, they are tetrahedra, pyramids, prisms, or hexahedra. Triangles in 2D and tetrahedra in 3D are most often used, as they can accurately represent arbitrarily-shaped objects, including those with curvilinear layouts. A unique feature of FEM elements is that they can be adaptively scaled to different sizes within the domain, leading to computationally efficient representations of electromagnetic fields without loss of accuracy. For example, small element sizes are typically specified in domain regions featuring rapidly varying spatial field profiles while larger element sizes are specified in regions featuring slowly varying spatial field profiles. The meshing procedure is typically performed with specialized programs, which use heuristic criteria such as geometric feature size and refractive index distributions in the domain to achieve accurate adaptive meshing without knowledge of the ground truth fields.

2.2.2 Basis Function Expansion

The fields within each finite element are approximated as $f(\mathbf{r}) \approx \sum_{i=1}^{n}, f_i\varphi_i(\mathbf{r})$, where $\varphi_i(\mathbf{r})$ are primitive basis functions serving as analytic descriptions of the electromagnetic fields and f_i are unknown primitive basis function coefficients that are to be determined. The primitive basis functions are typically chosen to be generic low-order polynomials, and they are specified in a manner such that field continuity between neighboring elements is consistent with Maxwell's equations. The total number of primitive basis functions within an element scales with the number of nodes within the element, n. The use of more nodes leads to the use of higher order basis functions and more accurate descriptions of the fields, at the expense of computational memory and overhead.

2.2.3 Residual Formulation

To solve our differential equation problem, our goal is to specify the unknown primitive basis function coefficients such that error between $f(\mathbf{r})$ and the ground truth field solutions is minimized. This error to be minimized can be framed in terms of the residual $r = \mathcal{L}(f) - s$, where $\mathcal{L}$ is an integro-differential operator defined by Maxwell's equations and s is the source. In the weighted residual method, the problem is cast in the weak form: we solve the system of equations $\langle w_i, r\rangle = \int_\Omega w_i r d\Omega$, where w_i is a set of N weighting functions, and we set these weighted residual expressions to zero. In the case where the weighting functions are chosen to be the same as the basis functions, this formulation is termed *the Galerkin method*. The problem can now be expressed in the form $\mathbf{A}\overrightarrow{\boldsymbol{x}} = \overrightarrow{\boldsymbol{b}}$, where $\mathbf{A}$ is a matrix that corresponds to Maxwell-based terms, $\overrightarrow{\boldsymbol{x}}$ is the vector containing the unknown coefficients, and $\overrightarrow{\boldsymbol{b}}$ is a vector of source terms, and it is solved by inverting $\mathbf{A}$.

2.3 Method of Moments (MoM)

Whereas FD and FEM solve for electric and magnetic fields using a volume meshing approach, MoM is a variational method that solves for J and Q (i.e., ρ) in Maxwell's equations using a surface meshing approach. More specifically, the field solution is expressed as a superposition of integrals composed of the problem's sources and a *Green's function*. With sufficient information about the known field in the problem setup, we task MoM with determining the unknown sources. [4] With surface meshing, MoM is memory efficient and fast, and it is particularly well suited for modelling unbounded systems comprising homogeneous media, such as scattering from a perfectly conducting metal structure. [5] Most generally, the MoM solves linear equations of the form:

$$\mathcal{L}(\varphi) = f, \tag{5}$$

where $\mathcal{L}$ is a linear operator, φ is the unknown quantity, and f is the excitation function enforced by the set of equations that dictate the physics of the problem. [5]

Similarly to FEM, Eq. 5 is numerically solved by determining the unknown coefficients of a basis function expansion of the solution,

$$\varphi = \sum_{n=1}^{N} a_n v_n, \tag{6}$$

where v_n are pre-selected basis functions and a_n are the scaling coefficients for which a numerical solution is determined. [5]

Plugging the expansion of Eq. 6 back into Eq. 5, a system of equations can be formulated to solve for the unknown expansion coefficients by applying weighting functions, similarly to the method presented for finding the FEM solution in Section 2.2. That is, we again encounter a problem in the form $\mathbf{A}\overrightarrow{\boldsymbol{x}} = \overrightarrow{\boldsymbol{b}}$, where $\mathbf{A}$ is a matrix containing weighted Maxwell-based terms, $\overrightarrow{\boldsymbol{x}}$ is the vector containing the unknown expansion coefficients, and $\overrightarrow{\boldsymbol{b}}$ is a vector of weighted forcing function terms, and it is solved by inverting $\mathbf{A}$. After determining $\overrightarrow{\boldsymbol{x}}$, the problem solution is given by inputting the basis function coefficients into Eq. 6.

3 Deep Learning Methods for Augmenting Electromagnetic Solvers

A judicious approach to overcoming the limitations inherent to classical CEM techniques is augmenting existing algorithms with deep learning methods. Such approaches benefit from the combined computational stability advantages of classical CEM and the inference abilities of deep learning. As part of a burgeoning effort to improve conventional CEM using deep learning, techniques have been developed that augment all of the primary methods introduced in Section 2. We introduce here several deep learning augmentation methods that offer impressive potential for computational acceleration over conventional CEM algorithms.

3.1 Time Domain Simulators

3.1.1 Hardware Acceleration

In FDTD, the update equations for an individual voxel require field data only from nearest neighbor voxels. As such, the evaluation of field updates across the full domain can be readily subdivided and evaluated in a parallelizable manner. This parallelization follows many of the mathematical operations naturally performed by machine learning algorithms, which are executed on graphics processing unit (GPU) or tensor processing unit (TPU) computing hardware. With the *RCNN-FDTD (i.e., recurrent convolutional neural network-finite difference time domain)* algorithm, [14] FDTD is cast in the formalism of machine learning, providing a natural software-hardware interface for parallelizing and accelerating FDTD simulations. The FD operator is expressed as a convolutional neural network (CNN) kernel and the time marching procedure is formulated in the recurrent neural network (RNN) framework [14], leading to speedups of 4.5 times over comparable CPU-parallelized code with no decrease in accuracy.

To illustrate the mapping of conventional FDTD to *RCNN-FDTD*, consider the procedure for solving for the H_x field component at an arbitrary time step for 2D TM modes (E_z, H_x, H_y). The FDTD algorithm employs Faraday's law to determine H_x at the next time step, requiring the most updated values of H_x and E_z. The update equation for H_x can be written generally as:

$$H_{x_{i,j}}^{t+1} = W_1 \cdot H_{x_{i,j}}^{t} + W_2 \cdot (E_{z_{i,j+1}}^{t+\frac{1}{2}} - E_{z_{i,j}}^{t+\frac{1}{2}}). \tag{7}$$

W_1 is termed the *spatial coefficient matrix*, W_2 is termed the *temporal coefficient matrix*, and both matrices can be directly read off from the conventional update equations. E_z and H_y are determined in a similar fashion from their respective FD update equations. In the formalism of machine learning, W_1 and W_2 can be directly described as CNN and RNN kernels, respectively. This setup is illustrated in Fig. 2.

With this method, no modification is made to the FDTD algorithm itself, rather its casting as cascaded neural networks allows straightforward hardware acceleration with GPUs and TPUs. As such, W_1 and W_2 are deterministically formulated based on the setup of the FDTD algorithm and no network training is performed. Taking advantage of the highly efficient parallelization made possible by the machine learning community to accelerate FDTD, the *RCNN* scheme illustrates a subtle consequence of the machine learning revolution. Although FD algorithms were previously parallelizable before the advent of accessible machine learning software and hardware, implementing and employing such algorithms required specialized knowledge of how to parallelize algorithms in code. With the wealth of machine learning resources currently available, code parallelization is a trivial task if it can be expressed within the formalism of machine learning algorithms.

3.1.2 Learning Finite Difference Kernels

The conventional FDTD algorithm involves a fundamental trade-off between accuracy and time step size: greater simulation accuracy requires finer spatial grid resolution, which is accommodated by the requirement of smaller time steps to maintain stability [9] and more computational memory. While the parellelization concepts from section 3.1.1 can help mitigate some of this computational loading, the trade-off still remains. A conceptually new way to address this trade-off is to consider FDTD simulations with spatially coarse grids and large time steps, and to specify W_1 and W_2 as learnable kernels trained from data. These kernels can be treated as super-resolution operators and have the potential to utilize learned nonlinear interpolation to produce electromagnetic field updates with the accuracy of spatially fine simulations.

The first attempt to develop learned FD kernels formulated the FDTD stepping process using either RNN or CNN operations, with learned coefficients that approximated the finite differencing approximations. [15] The RNN-only formulation of Yee grid field prediction is similar to the *RCNN-FDTD* method, as both employ a simple RNN architecture to obtain the field values at succeeding time steps. The main difference in the structure of the algorithms is that instead of collecting

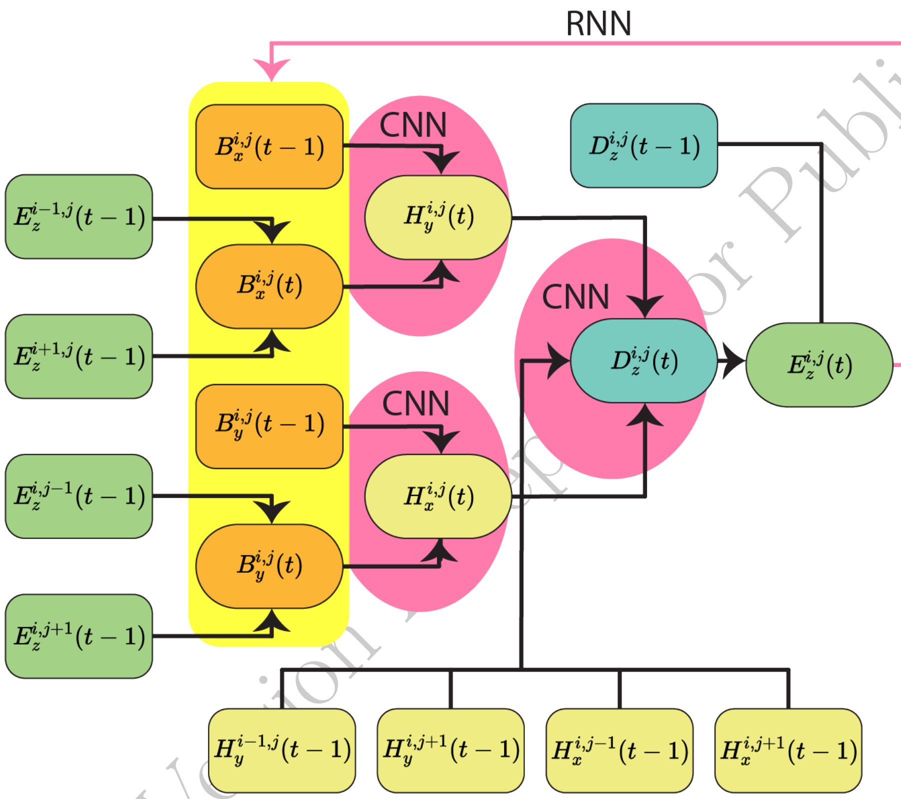

Figure 2: The *RCNN* update scheme, which expresses the FDTD formalism within the framework of CNN and RNN operations to take advantage of the accelerating software and hardware developed by the machine learning community. CNN operations collect and process field values from neighboring cells to perform the finite spatial differencing, after which the time-stepping is performed using an RNN operation. No learnable parameters are used in this scheme.

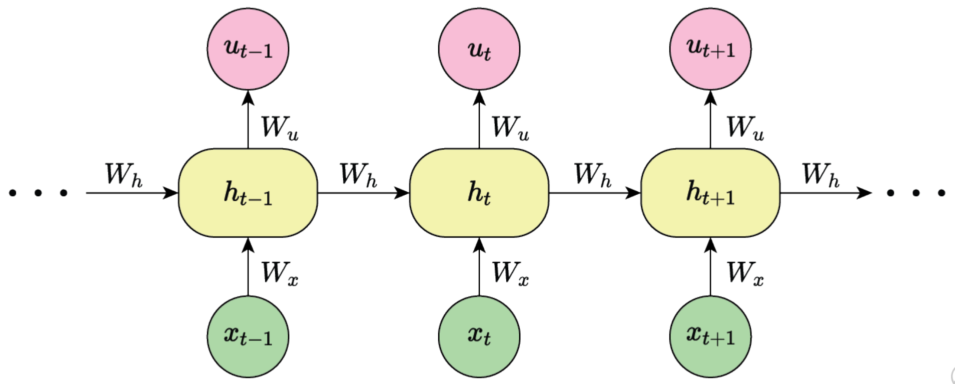

Figure 3: A basic recurrent neural network (RNN) architecture. The time sequential inputs of the RNN are processed one step at a time, with a preceding operation's processed result serving as part of the next time step's input. Note that the same set of learned weights are used at each time step of the process depicted in the figure. This "unfolded" representation is thus unfolded in time, and does not depict multiple layers of a neural network.

the scaled field values of interest from neighboring voxels through convolutional operations, the RNN-only algorithm obtains the raw voxel values of all immediate neighboring voxels as a vector:

$$x_t = [u_{n+1,m}(t), u_{n-1,m}(t), u_{n,m}(t), u_{n,m+1}(t), u_{n,m-1}(t)]. \tag{8}$$

As illustrated in Fig. 3, this is then processed using a basic recurrent neural network (RNN). Mathematically, the computational graph of these basic RNNs may be formulated as two governing equations,

$$\begin{cases} h_t = \sigma_h(W_x x_t + W_h h_{t-1}) \\ u_t = \sigma_u(W_u h_t) \end{cases}, \tag{9}$$

where x and y are the inputs and outputs of the RNN at any one time, respectively, h are the values of the hidden layers, W are the learned weight matrices of either the input, hidden, or output layers, and σ is the neural network activation function. RNNs are uniquely positioned to efficiently process sequential data because the same architecture is capable of processing arbitrarily sized inputs without increasing the number of parameters. For FDTD-like field predictions, one pass of a voxel's state vector, x_t, through the RNN yields its value at the following time step, $u_{n,m}(t+1)$.

The efficacy of the RNN kernel learning method is analyzed by training it to solve the 2D wave equation in a square domain without structures and Dirichlet boundary conditions upon a Dirac delta excitation in a fixed location at time step 0. The training data is composed of the field values over 100 time steps in the simulation domain, which is discretized into 11x11 voxels. The trained model is evaluated at a fixed time step for an excitation location different than the one used in the training data. The trained RNN's predictions maintain the ground truth's shapes and features on the relatively restricted dataset it was tested on in [15], but the average relative error obtained was a substantially high 15%.

Although the RNN is superior at processing sequential data, its architecture is not designed to inherently process the differential spatial relationships between neighboring voxels, which is a key aspect of the FDTD algorithm. Thus, it is more natural to express FDTD steps in the formalism of the CNN, where voxel values are updated as combinations of themselves and neighboring voxel values. In this particular introductory study, the input to the CNN is the entire 2D field at time step t, and the output is the entire field at the next time step, $t + \Delta t$. The chosen neural network architecture was comprised of a single convolutional layer, followed by a pooling layer and two fully connected layers. The CNN was trained using the same dataset as the RNN, but tested instead using the same excitation source as used for training at a time step beyond that present in the training dataset. This network architecture and testing procedure led to much higher accuracy compared to the RNN approach, with an average relative error of less than 3%.

While the first attempt to learn generalizable FDTD kernels using machine learning falls short of demonstrating an accurate, generalizable technique, it establishes promising results and the potential for related concepts to transcend the limitations of the conventional FDTD method. It is anticipated that more advanced neural network architectures and more training data can lead to a more scalable and generalizable time domain simulator. Future studies will need to be quantitatively benchmarked with equivalent conventional FDTD algorithms to demonstrate the efficacy of machine learning to efficiently establish super-resolution methods with learned FDTD kernels.

3.1.3 Learning Absorbing Boundary Conditions

Machine learning models can also reduce the computational burden of FDTD simulations by producing a one-voxel-thick ABC model that performs similarly to much thicker conventional PMLs. [16] The proposed model utilizes a fully connected neural network with a single hidden layer, and its performance is demonstrated for a 2-D TE$_z$ problem. The training data is accrued by collecting the PML field values for randomly-selected points, P, on the simulation domain-PML boundary. For each point, the H_z, E_x, and E_y field values are each collected for the four closest positions to P on the Yee grid at time step t of the FDTD simulation to form the input data set. The corresponding H_z, E_x, and E_y field components are collected at the single closest positions to P for time step $t + \Delta t$, forming the output data set. The training data is collected for several randomized incident angles of illumination. Throughout the entirety of the time-stepped FDTD test simulation, it is demonstrated that the machine learning FDTD method is capable of maintaining an error rate below 1%, which is comparable to that of a 5-voxel-thick conventional PML. This study demonstrates that machine learning is a viable solution for decreasing the compute requirements of unbounded CEM problems.

3.2 Augmenting Variational CEM Techniques Via Deep Learning

FEM and MoM are prime candidates for acceleration via machine learning due to the steep computational scaling for matrix inversion, resulting in long simulation times for problems with large sets of basis functions or large simulation domains. The approaches explored to date involve either attempting to directly predict the final solution within the framework of a specific method or to use machine learning to reduce the dimensionality of the problem. [17, 18] Our focus in this section will be on the latter, which takes advantage of the full rigor of the variational method, including deterministic solution error bounding and estimation. [17]

Dimensionality reduction of variational techniques can be achieved by employing a neural network to learn an ultra-reduced, but highly expressive, problem-dependent set of basis functions that are in turn used to find rigorous solutions. [17] [18] The neural network is trained to predict the coefficients, a, of the complete set of primitive basis functions, F, given the rigorously-solved coefficients of an analog of the problem formed by a significantly reduced basis. The rigorous solution to the reduced problem is computationally inexpensive compared to the full problem, as significantly smaller matrices must be inverted. Neural networks are well-suited to learn the relationship between the reduced basis and full basis solutions because of their proven strength of empirically establishing accurate relationships between high-dimensional datasets and their low-dimensional representations. [17]

To mitigate the impact of deviations between the neural network-predicted coefficients and ground truth values, the network-predicted coefficients are used to construct *macro* basis functions, F_{macro}, which comprise a linear combination of primitive basis functions weighted by the predicted coefficients. The primitive basis functions selected for these macro basis functions are chosen to be distinct from those imposing boundary conditions in the problem, F_{boundary}. Each of the macro basis functions is therefore defined as:

$$f_{\text{macro}} = \sum_{I_{\text{macro}}} a_i f_i, \tag{10}$$

where $I_{\text{macro}} = \{i \in [1...N] \mid f_i \notin F_{\text{boundary}}\}$. The set of basis functions composed of the union between F_{macro} and F_{boundary} is denoted by $\bar{F}$ and is comprised of much fewer elements than the total number of primitive basis functions N. The CEM problem is once again solved using the conventional variational method with the $\bar{F}$ basis. The careful distinction of the macro and boundary basis functions as separate entities in $\bar{F}$ allows for an accurate, convergent solution throughout the entirety of the domain.

This technique of utilizing macro basis functions in conjunction with the variational method performs significantly better than more naive techniques. For a simple 1-D scattered field prediction problem, the macro basis technique was benchmarked against conventionally solving the problem using the same number of primitive basis functions as in $\bar{F}$ and also against the direct neural network-predicted solution. For the real component of the calculated field, the macro basis function formulation method achieved an average root-mean-square error (RMSE) of about 0.15 on the test dataset, whereas the conventional approach attained an average RMSE of about 0.3 and the direct neural network prediction realized an average RMSE of about 0.6. The macro basis function formulation technique results in significant reductions in computational complexity for conventional CEM solvers. Conventional iterative methods for performing the variational calculation face computational complexity of $O(N^2)$ and direct methods suffer from $O(N^3)$ scaling. By reducing the number of basis functions used by the variational algorithm using the macro function technique (i.e., a reduction from $|F|$ to $|\bar{F}|$) by a factor of γ, iterative and direct solvers experience speedups of a factor of $1/\gamma^2$ and $1/\gamma^3$, respectively. [17] The computational cost for solving the initial dimensionally-reduced problem is typically negligible and thus ignored in this analysis. These computational savings are significant as problem domains scale in size and complexity. It is also noted that this method can also extend to FDFD techniques by expressing the field as a weighted sum of Dirac-delta basis functions that are centered on the Yee grid sample points. Here, the "reduced basis" is a coarser grid than that of the original problem, thus framing the task of this acceleration technique as a super-resolution algorithm.

4 Deep Electromagnetic Surrogate Solvers Trained Purely with Data

In this Section, we will discuss how direct solution predictions of electromagnetic problems can be cast as data-driven computer vision problems solved via neural networks. These deep networks are tasked with learning a one-to-one mapping between a set of structures and their respective boundary conditions to an output consisting of the set of electromagnetic field distributions. In

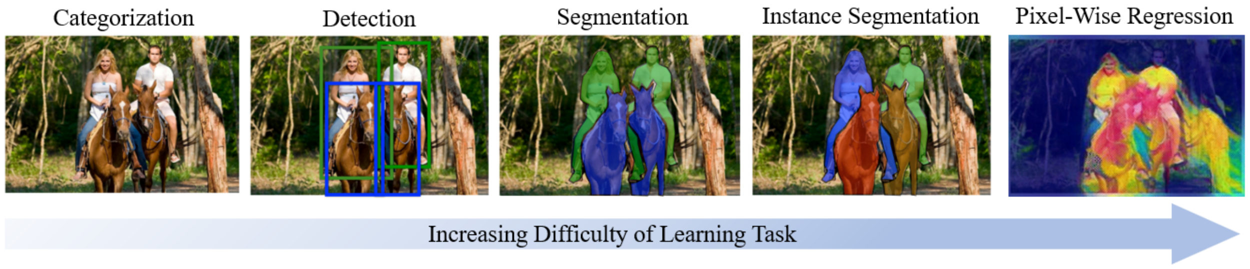

Figure 4: The five main tasks of computer vision, sorted in order of increasing training difficulty. *Categorization* involves predicting which category an image belongs to from a pre-determined set of categories, *Detection* identifies and locates the positions of categories of objects in an image, *Segmentation* determines which pixels of an image specifically belong to each category, *Instance Segmentation* additionally identifies unique instances of known categories of objects within the image, and *Pixel-Wise Regression* predicts a continuous value of interest for each pixel in the image. Pixel-wise regression provides the framework for learning the relationship between an inputted electromagnetic domain and its corresponding field response. The figure is adapted with permission from Professor Leonid Sigal's CPSC 532S course, *Topics in Artificial Intelligence: Multimodal Learning with Vision, Language, and Sound*, delivered at the University of British Columbia, 2021.

this manner, the network is tasked with learning the operator that maps device structure to electric fields, where an operator by definition maps a function to another function. More specifically, it learns the Green's function operators that map spatial dielectric structure to field for a given source. This task of predicting the electromagnetic field value at each pixel position of the inputted "image" is referred to as *pixel-wise regression* and is the most demanding class of computer vision problems (Fig. 4). [19, 20] To perform this learning task, training data is generated using conventional CEM solvers, which provides a straightforward pathway to producing up to millions of highly accurate input-output data pairs needed for training machine learning models.

In an initial demonstration of a surrogate solver, a deep CNN [21] was trained to predict the electrostatic response of arbitrarily-shaped dielectric ellipsoids in 2-D 64x64 pixel and 3-D 64x64x64 pixel domains excited by a point source located in one of eleven positions on a line in the periphery of the simulation domain. [22] The CNN's first input channel is composed of the dielectric distribution within the full domain, and the second is the Euclidean distance from each voxel to the excitation point source. The single-channel output of the CNN is the electromagnetic response of the dielectric distribution on a restricted output domain of 32x32 pixels for the 2-D case or 32x32x32 pixels for the 3-D case, located in the center of the full-size input domain. The input-output training data pairs were obtained using a finite difference solver that calculated Poisson's equation constrained by Dirichlet boundary conditions. The model was trained using an Adam optimizer [23] for a loss function that took the squared L^2 norm between the base-10 logarithms of the CNN-predicted and ground truth fields. The network architecture is relatively generic and not explicitly configured for the task of pixel-wise regression. As depicted in Fig. 5, six consecutive 3D convolution encoding operations encode the inputted source and dielectric permittivity information, which is then decoded into corresponding field values throughout the computational domain by two consecutive 3D convolution operations.

The most generalized problems undertaken by the CNN in Fig. 5 involved the modeling of the electrostatic response of four ellipsoids for the 2-D case and two arbitrarily-shaped ellipsoids for the

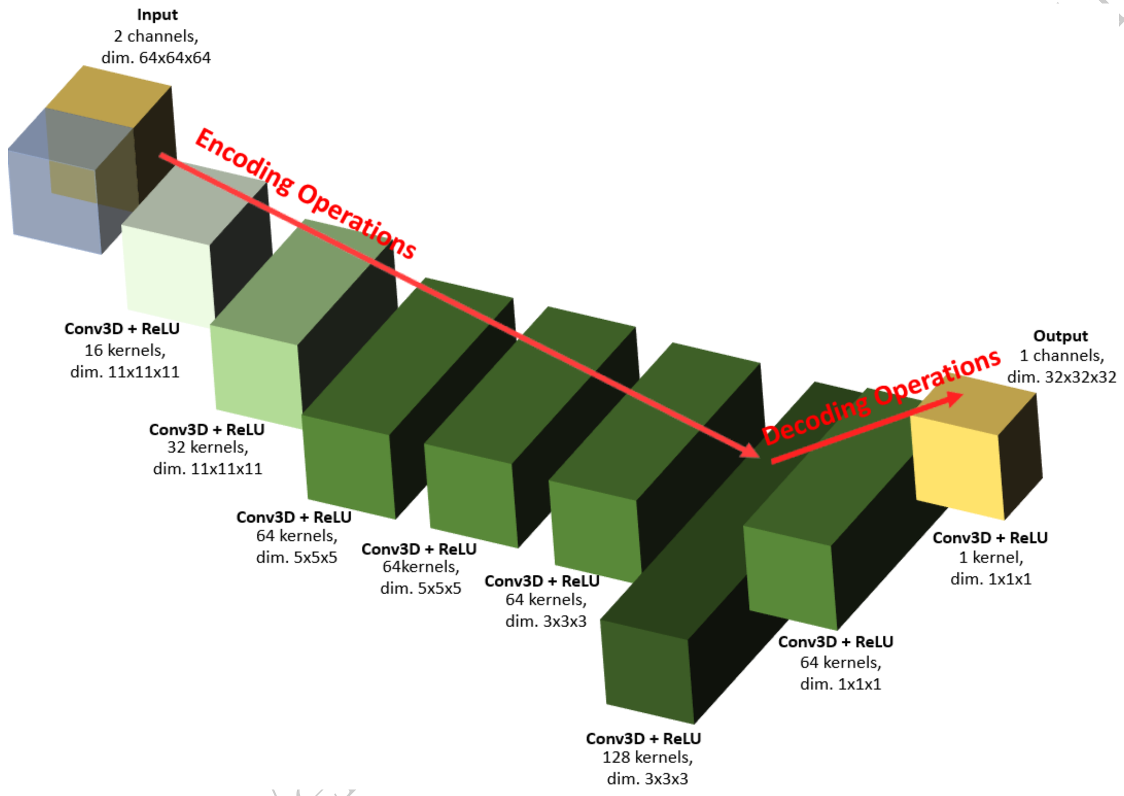

Figure 5: The 3D CNN-based architecture employed in [22]. Two $64 \times 64 \times 64$ channels are inputted, one containing the dielectric information from the input domain and the other the distance between each pixel and the point source. The neural network input then undergoes an encoding and decoding process executed by 3D convolutional operations to output a one-channel $32 \times 32 \times 32$ map of the electromagnetic response in a simulation domain centered in the input domain. The number of kernels and their respective dimensions that yield each processed block are labeled.

3-D case. In the former case, the ellipsoids are always centered in each quadrant of the domain, with a fixed major axis length and variable minor axis length, rotation angle, and dielectric permittivity. The 3-D case contains one ellipsoid centered in each half of the simulation domain, each with variable semiaxes and permittivity. The 2-D CNN was trained with $40,000$ training samples and tested with $10,000$ samples. Both the 2-D and 3-D variations of the simulator achieved relative errors below 3%. While this preliminary demonstration was applied to a relatively narrow range of variable input parameters, it introduced the potential for deep networks to serve as surrogate electromagnetic solvers.

To model broader classes of electromagnetics systems with high accuracy, more careful consideration of the neural network architecture is required. In this aim, convolutional-based encoder-decoder neural networks that first encode the input image and then decode its latent representation have been developed by the machine learning community for pixel-wise regression tasks, making them well suited for modeling entire classes of electromagnetics systems. One of the highest performing segmentation architectures is the U-Net, [24] depicted in Fig. 6. The network is a symmetric, fully-convolutional encoder-decoder neural network composed of a series of "convolutional blocks," which consist of a series of convolution, batch normalization, and ReLU nonlinear activation operations. The first part of the network performs *encoding* and contracts in spatial dimension size from one convolutional block to the next using max pooling operations at the end of each block. The second part of the network performs *decoding* and is composed of blocks that are symmetrically sized compared to the encoding part of the network, with the expansion of the spatial dimensions being realized through upsampling operations executed at the end of each convolutional block. Residual connections are employed within each residual block, adding the result of the first convolution operation to the batch-normalized result at the end of the block. Shortcut connections are also used between the result of each encoding convolutional block before the max pooling operation to the input of each decoding block, with the connections made between symmetric blocks of each half of the network.

The direct connections between the two halves of the network between the same resolution levels are key to preserving physically-relevant information at each pixel between the input and output images of the network. Besides the residual and shortcut connections facilitating the flow of gradients during the training process to prevent the *vanishing gradient problem*, [25] they help to capture a strong relationship between the inputted dielectric geometry and the characteristics of the outputted wave profile. They also preserve a direct relationship between dielectric discontinuities and learned boundary conditions that are localized in the pixels of the output that are proximal to those containing the discontinuity region in the input. Owing to these features of its architecture, the U-Net scales well to the processing of large amounts of training data and has been successfully applied to both 2-D and 3-D domains. [26, 27]

Preliminary U-Net-based networks accurately predicted the magnetic field response of arbitrarily superimposed ellipsoid and polygonal-shaped dielectric scatterers with a permittivity ranging from 2 to 10, under the illumination of arbitrarily-angled plane waves. [28] With this approach, the dielectric scatterer image input of the neural network is augmented with a second channel containing the excitation source, demonstrating that a single trained network is capable of incorporating general simulation configurations. The U-Net can also be modified to function with 3D convolutions, as shown in Fig. 7. [26] Over a domain of size 45x45x10 pixels, the U-Net was trained to predict the full electromagnetic response of one or more randomly-arranged cuboidal dielectric blocks with fixed height upon normally incident plane wave illumination. The neural network, trained with $28,000$ training samples, achieved an average cross-correlation of 0.953 with the ground truth fields, with only 2.3% of tested devices exhibiting markedly poor outlier perfor-

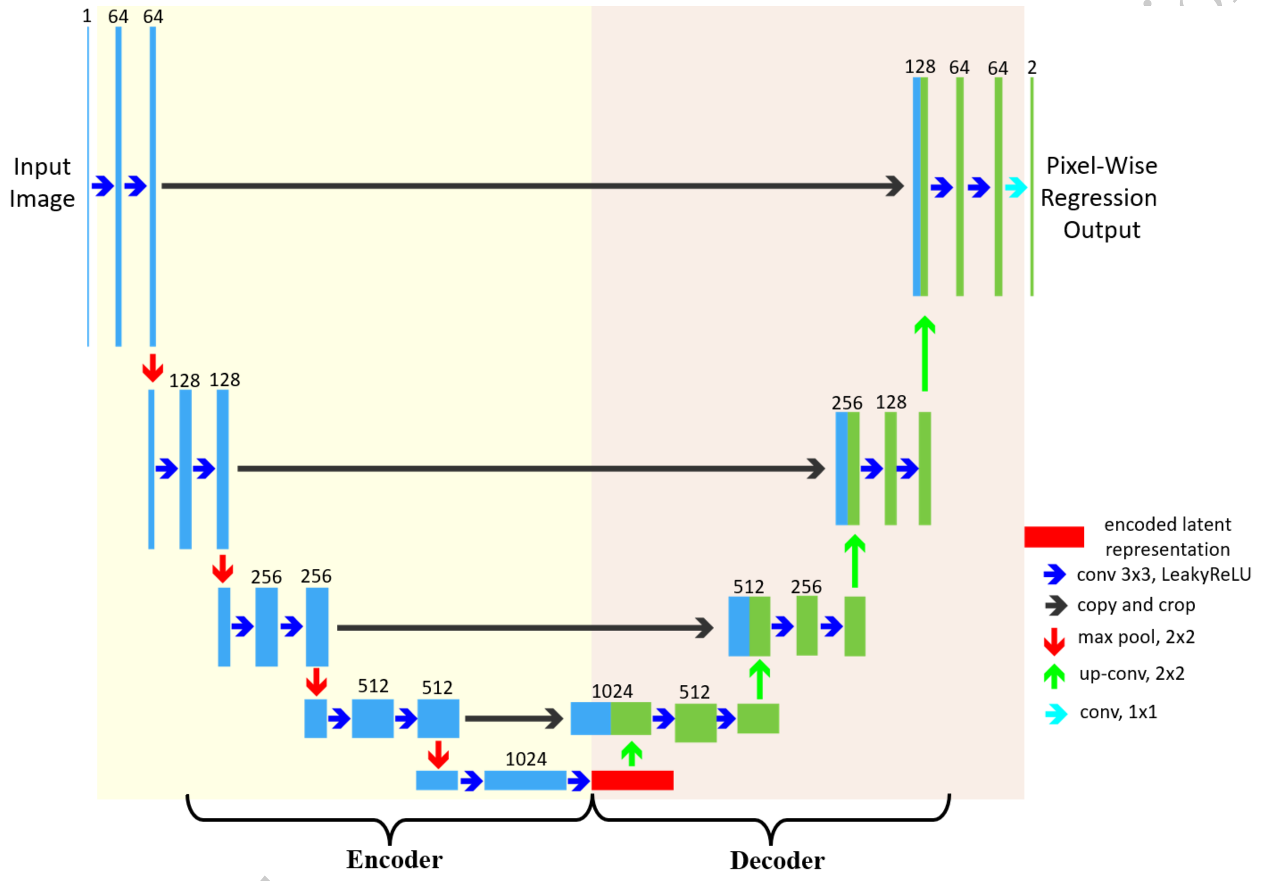

Figure 6: A generic U-Net architecture, which in this case accepts a one-channel input and produces a two-channel output. The first half of the U-Net consists of a series of *encoding* “convolutional blocks,” which are followed by a symmetrical series of *decoding* “convolutional blocks.” Spatial dimension reduction of the input is achieved in the encoding portion of the network using max pool operations, and the input dimensions are restored in the decoding portion by up-sampling using the transposed convolution operation.

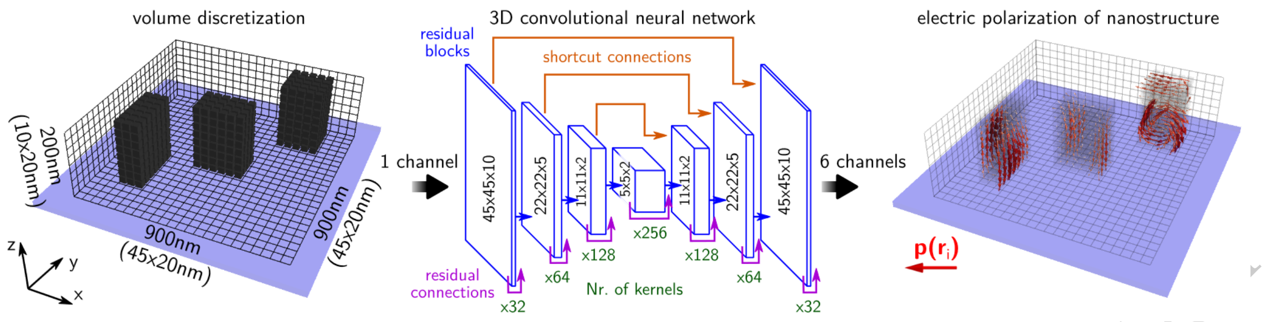

Figure 7: The U-Net fashioned in [26] utilizes 3-D convolutions to process an arbitrary configuration of 3-D dielectric pillars (as shown) or 2-D metallic polygons into the corresponding real and imaginary parts of the electromagnetic field components. Reprinted with permission from [26]. Copyright 2022 American Chemical Society.

mance, under 0.80. These field solutions could be used in conjunction with other physics-based calculations, such as near-to-far-field transformations, to accurately compute other physical quantities, such as far-field scattering profiles. Upon calculation of such secondary physical quantities using the same neural network trained using $28,000$ devices, the near-field was manifestly incorrect (with a ground truth cross-correlation under 0.80) for only 5.3% outlier devices, and the far-field was similarly significantly incorrect for 10.1% of devices. The occurrence of failed predictions is the result of the U-Net providing a non-physically rigorous solution, which has a risk of producing significantly erroneous results. For example, the neural network performs poorly in regimes of resonance, which are acutely underrepresented in the randomly-generated training dataset. The same study also demonstrated that a U-Net can be generalized to non-dielectric devices, by training one to predict the field response of arbitrarily-shaped planar metallic polygons. [26]

5 Deep Surrogate Solvers Trained with Physical Regularization

Although state-of-the-art segmentation neural networks, including the U-Net, are capable of learning the relationship between complex structures and the corresponding electromagnetic field responses, the approach of training solely based on input-output ground truth data pairs has its limits. As electromagnetic fields are constrained by Maxwell's equations, incorporating this knowledge explicitly into the loss functions can lead to more robust field solutions. This section focuses on methods that exploit the governing physics of a solution space to train more rigorous general machine learning solvers: physics-informed neural networks (PINNs), physics-informed neural networks with hard constraints (hPINNs), and WaveY-Net.

5.1 Physics-Informed Neural Networks (PINNs)

PINNs are a newly developed class of neural networks that utilize the underlying governing physical relations of a simulation domain to regularize the training data to produce robust physical function approximators. [29] As they model the functional mapping between structure and fields for a single device and source, they are distinct from the operator mapping concepts for general classes of problems introduced in section 4. As illustrated in Fig. 8, PINNs take the governing partial differential equation (PDE) parameters as inputs, including space and time coordinates that specify a location in the solution space, and they output the corresponding solutions to the PDE. These networks therefore produce field values as a continuous function of space-time coordinates, exceeding the resolution limits set by discretized meshes and cubic scattered grids utilized by conventional CEM techniques (Section 2). PINNs can be employed to either deduce solutions to PDE problems or to discover the parameters that best satisfy a general PDE form based on data. The former use case, for which the governing PDEs are Maxwell's equations, will be the focus of this Section.

The key to encoding the physics of a governing PDE into a PINN is to specify a training loss that balances data error (i.e., deviations between predicted and ground truth values) with physical error (i.e., self-consistency of the predicted values and its derivatives with the governing PDEs). Consider a general one-dimensional PDE of the form:

$$\frac{\partial u}{\partial t} + \mathcal{N}[u] = 0,\ x \in \Omega,\ t \in [0, T], \tag{11}$$

where $u(t, x)$ is the underlying solution of the PDE within the domain of interest, Ω, and $\mathcal{N}$ is a nonlinear differential operator. The loss is:

$$MSE = MSE_d + \alpha MSE_f, \tag{12}$$

where MSE_d is the data error and MSE_f is physical error. The latter can be further decomposed into volumetric and boundary terms, MSE_u and MSE_b, respectively, resulting in a total loss of:

$$MSE = MSE_d + \alpha(MSE_u + \gamma MSE_b);\ \alpha, \gamma > 0. \tag{13}$$

In this generalized formulation of the PINN loss, hyperparameters α and γ are used to tune the relative contributions from the data-driven, volumetric physical, and boundary physical errors to the overall loss. It is necessary to carefully adjust these relative weights in order to obtain a highly accurate converged solution for non-trivial problems. [30] Initial condition terms can also be added to the loss function for enforcement as needed.

Fully connected architectures are generally chosen for PINNs. To train the networks, predicted values are used to calculate the loss function, which is backpropagated back into the network using automatic differentiation (AD) to update the network weights and reduce prediction loss. AD is also used to calculate the derivative of predicted values with respect to the input space and time variables, making the evaluation of derivative-based PDE loss terms straightforward. Backpropagation can also be exploited by PINNs to optimize an objective functional and perform inverse design of devices, as explored in Section 5.2.

While PINNs introduce a qualitatively new way of solving PDEs with machine learning, they exhibit shortcomings that limit their direct application to solving broad classes of electromagnetics problems. As reviewed in Section 4, the efficacy of deep networks to learn full wave physics can strongly benefit from specialized network architectures. PINNs, as presented in this section, utilize generic, fully connected architectures that limit the complexity of problems and size of domains to which they are applicable. Furthermore, PINNs require the physical system (i.e., the geometry and dielectric distribution in the domain) to be fixed, limiting the method from being applied to general classes of electromagnetics problems.

5.2 Physics-Informed Neural Networks with Hard Constraints (hPINNs)

A broadly useful application of Maxwell solvers is photonic device optimization, which involves the identification of a dielectric distribution that maximizes a desired objective function within a

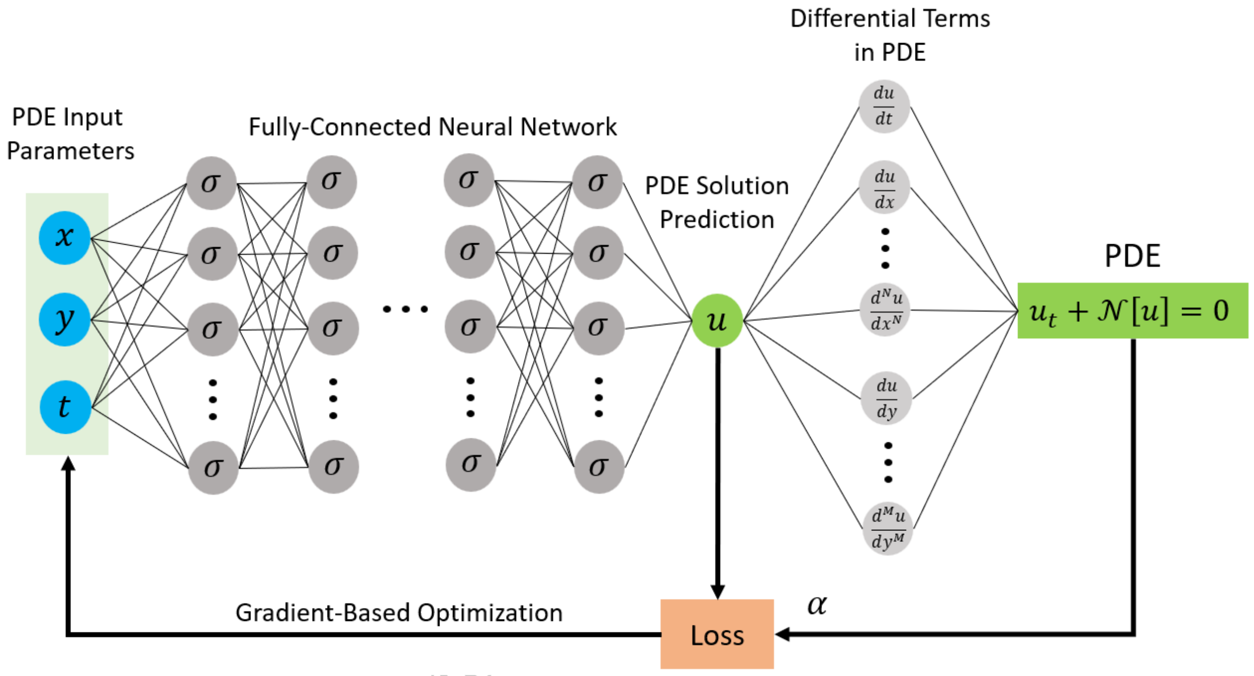

Figure 8: The general PINN training framework. A fully-connected neural network (FCNN) is trained to predict the solution to a PDE, or more generally a set of PDEs, at the location in space-time specified by the inputted PDE parameters. The differential terms appearing in the PDE being solved are calculated using automatic differentiation. The loss is then constructed as a combination of data loss, which is known for a limited set of points in the domain, and the physics loss calculated across the domain using the PDE. A special case of PINN operation allows it to function without the use of any data and instead relies purely on the PDE itself to find the solutions.

solution space that is constrained by Maxwell's equations. For this purpose, a variant of PINNs with hard constraints, hPINNs, has been developed, for which network training is facilitated during a device optimization process without training data. The loss function in hPINNs is specified to satisfy the governing set of PDEs and simultaneously maximize the objective functional. [31] hPINNs use a loss function formulation different from the one employed by PINNs. PINNs cannot be used directly for inverse optimization because the gradient of the network's loss, generated from the set of governing PDEs, is not generally consistent with the gradient of the objective function. This means that a naive optimization implementation using PINNs would generally lead to a solution that does not satisfy the governing PDEs, which is a hard constraint of the problem. hPINNs introduce hard constraints using either the *penalty method* or the *augmented Lagrangian method*. Furthermore, hPINNs are trained using only physics information from the governing PDEs, without training data.

hPINNs directly impose boundary conditions in the formulation of the optimization problem. Formulations for Dirichlet and periodic boundary conditions have been developed, which can be straightforwardly extended to all other types of boundary conditions, such as Neumann and Robin. For Dirichlet boundary conditions, the neural network output is constructed as a sum of the analytical boundary condition and a scaled PINN output, which is scaled to zero where boundary conditions apply. The total field solution, $\hat{u}$, can therefore be expressed as:

$$\hat{u}(\mathbf{x};\theta_{\mathbf{u}}) = g(\mathbf{x}) + \ell(\mathbf{x})\mathcal{N}(\mathbf{x};\theta_{\mathbf{u}}), \tag{14}$$

where $\theta_{\mathbf{u}}$ are the trainable weights of the network, $g(\mathbf{x})$ is the boundary conditions function imposed for the specified domain coordinates, $\mathbf{x}$, and $\mathcal{N}(\mathbf{x};\theta_{\mathbf{u}})$ is the neural network output that is scaled to zero by ℓ where boundary conditions apply. For non-trivial domains, spline functions are used to approximate $\ell(\mathbf{x})$. For periodic systems, periodic boundary conditions are implemented by expressing the periodic direction coordinate in $\mathbf{x}$ as a Fourier basis expansion, which imposes periodicity due to the periodicity of the basis functions.

The *penalty method* further enforces the hPINN's solution to adhere to PDE constraints. Consider the objective function $\mathcal{J}(\mathbf{u};\gamma)$, which is dependent on the PDE solution, $\mathbf{u}(\mathbf{x})$, and the quantity of interest (i.e., the optimized device design), $\gamma(\mathbf{x})$. The design optimization problem is expressed as the following unconstrained problem:

$$\min_{\theta_u,\theta_\gamma}\mathcal{L}(\theta_u,\theta_\gamma) = \mathcal{J} + \mu_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}, \tag{15}$$

where $\mathcal{F}$ represents the N governing PDEs, $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_N\}$, and $\mu_{\mathcal{F}}$ is a fixed penalty coefficient. To prevent either the ill-conditioning or insufficient PDE constraint forcing of the gradient-based minimization of this equation, the design optimization problem is reframed as a sequence of unconstrained problems, with $\mu_{\mathcal{F}}$ incremented with each iteration by a constant multiplicative factor that is greater than 1. This thus yields,

$$\min_{\theta_u,\theta_\gamma}\mathcal{L}^k(\theta_u,\theta_\gamma) = \mathcal{J} + \mu_{\mathcal{F}}^k\mathcal{L}_{\mathcal{F}}, \tag{16}$$

for the k^{th} iteration. This optimization process yields two neural networks, $\hat{\mathbf{u}}(\mathbf{x};\theta_u)$ and $\hat{\gamma}(\mathbf{x};\theta_\gamma)$, the latter of which provides the structure of the optimized device. Note that this process trains the hPINNs without data, relying instead solely on the physics from the governing PDEs. While the penalty method is capable of optimization, it still suffers from potential ill-conditioning and also problem-dependent poor convergence rates. These are alleviated by the augmented Lagrangian method formulation of the optimization problem, which adds a Lagrangian term to the minimization expression, resulting in:

$$\min_{\theta_u,\theta_\gamma}\mathcal{L}^k(\theta_u,\theta_\gamma) = \mathcal{J} + \mu^k_{\mathcal{F}}\mathcal{L}_{\mathcal{F}} + \frac{1}{MN}\sum_{j=1}^{M}\sum_{i=1}^{N}\lambda^k_{i,j}\mathcal{F}_i[\hat{\mathbf{u}}(\mathbf{x_j});\hat{\gamma}(\mathbf{x_j})]. \tag{17}$$

Eq. 17 is then optimized using a method inspired by the Lagrangian multiplier optimization method. [32]

The penalty and Lagrangian methods were utilized to design a 2-D transmissive dielectric metamaterial structure capable of a desired holographic wavefront task. The neural network was constructed to impose periodic boundary conditions along one axis and Dirichlet boundary conditions along the other. The penalty method outperforms the soft constraints-optimized objective by about 2%, though it still suffers from the ill-conditioning problem in some hyperparameter regimes. The difficulty with convergence is overcome by the Lagrangian method. Both the well-converged penalty method and the Lagrangian method achieve average final solution PDE losses of about 10^{-5}, as defined by Eq. 18. Such strong convergence to physically-valid solutions indicates that hPINNs are capable of serving as both full-wave solvers and gradient-based optimizers for photonics systems.

hPINNs are readily treated as surrogate electromagnetic solvers by dropping the objective function term in the loss function. In this formulation, the loss function simply becomes the mean squared sum of the real and imaginary components of the governing PDEs at each coordinate:

$$\mathcal{L}_{\mathcal{F}} = \frac{1}{2M}\sum_{j=1}^{M}((\Re[\mathcal{F}[\mathbf{x}_j]])^2 + (\Im[\mathcal{F}[\mathbf{x}_j]])^2). \tag{18}$$

As a simple proof-of-concept demonstration, the hPINN solver, with the same boundary conditions as above, was employed to solve for the fields propagating through a domain with a constant permittivity. As illustrated in Fig. 9, these resulting field profiles were qualitatively consistent with those solved using a conventional FDFD algorithm. Quantitatively, the converged solution for this test problem achieved an L^2 relative E-field error of 0.12% for both the real and imaginary components and an average PDE-informed loss, as defined by Eq. 18, of 0.0001% across the entire simulation domain. For the non-trivial case of an optimized dielectric distribution slab, both the penalty and Lagrangian methods achieved PDE losses of approximately 0.0012%.

5.3 WaveY-Net

WaveY-Net is a deep learning paradigm that combines operator learning concepts from Section 4 with physical regularization to produce a high fidelity surrogate Maxwell solver for broad classes of electromagnetics systems. [27] The training overview and loss formulation for the generation of the field maps of periodic silicon nanostructure arrays from normally incident TM-polarized illumination is illustrated in Fig. 10. The input to WaveY-Net is an image of the simulation domain and it outputs two channels, the real and imaginary magnetic field components of the solution. Only the magnetic fields, as opposed to all electromagnetic fields, are predicted to reduce the load on the network's learning capacity. The electric fields are subsequently computed by applying Ampere's law to the network-predicted magnetic field maps. In a manner consistent with PINNs, the WaveY-Nets loss function is a combination of a hyperparameter-scaled physical residue term, L_{Maxwell}, and a data loss term, L_{data}:

$$L_{\text{total}} = L_{\text{data}} + \alpha L_{\text{Maxwell}}. \tag{19}$$

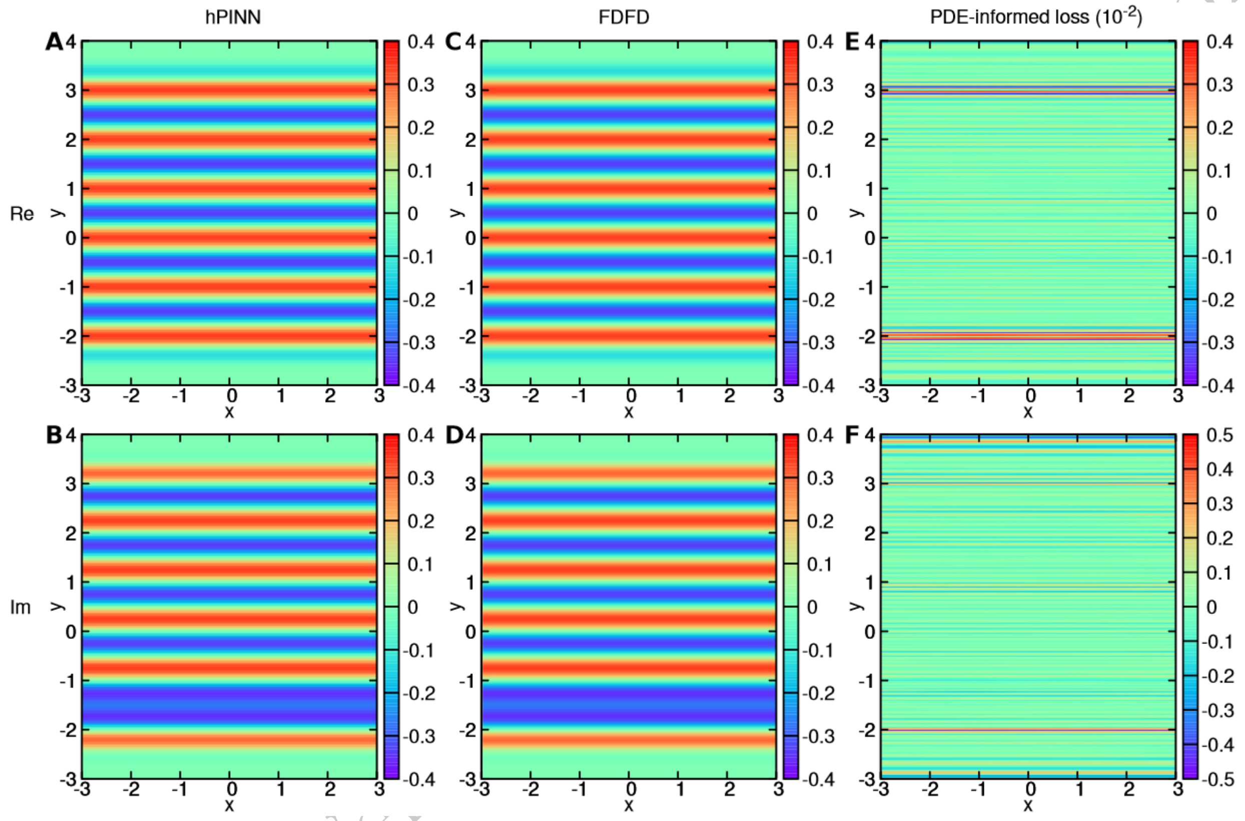

Figure 9: Benchmarking the hPINN in "forward" mode against FDFD, which is equivalent to training a PINN with physics information only. This demonstration was carried out on the trivial case of a domain with unity permittivity, but constrained by periodic and PEC boundary conditions. The hPINN (A) real and (B) imaginary component solutions are plotted across the entire domain alongside the FDFD (C) real and (D) imaginary component solutions. The PDE-informed loss, as defined in Eq. 18, of the forward-mode hPINN after convergence is plotted for the real (E) and imaginary (F) components. Reprinted from [33]. Licensed under CC BY-NC-ND 4.0.

The Maxwell residue is calculated using the Helmholtz relation derived for the magnetic field:

$$\nabla \times (\frac{1}{\varepsilon}\nabla \times \hat{H}) - \omega^2 \mu_0 \hat{H} = 0, \tag{20}$$

where $\hat{H}$ is the magnetic field predicted by the neural network. WaveY-Net is able to successfully implement robust physical regularization using Maxwell's equations by employing the Yee grid formalism to calculate L_{Maxwell}.

The high-level working principle of PINNs and WaveY-Net appear similar: both inject prior knowledge about governing physical laws explicitly into the loss function in a manner that makes the outputted fields more consistent with Maxwell's equations. In spite of this, PINNs and WaveY-Net are ultimately fundamentally different: PINNs learn the solution to a governing PDE over a continuous spatio-temporal domain with fixed geometries and boundary conditions, whereas WaveY-Net enforces consistency with Maxwell's equations using the Yee grid formalism to assist in the data training of entire classes of electromagnetic structures. Furthermore, unlike PINNs, Wavey-Net is not performance-bound to any specific type of neural network architecture, meaning that the framework can be applied to training the best-performing image segmentation neural networks, as outlined in Section 4, or to purpose-built and other high-performing future network architectures.

Calculating the physical residue for each point in the domain and calculating the electric fields from the magnetic field predictions requires the computation of spatial derivatives along each spatial axis at all points. This task must be done efficiently to manage the computational cost of the procedure. WaveY-Net casts Yee grid-discretized differentiation as a convolutional operation in the deep learning framework, enabling highly optimized parallelized computing for these calculations. Specifically, the differentiation operations are discretized using first-order central difference approximations and then executed using GPU-accelerated convolutional operations, as introduced in Section 3.1. It has been demonstrated that WaveY-Net is more efficient than a data-only U-Net tasked with full E and H field prediction, and it performs equally as well as an orders of magnitude more computationally expensive FDFD solver for the task of adjoint optimization. When tasked with predicting the H field of a periodic nanostructure illuminated with TM-polarized light, the WaveY-Net demonstrated an average of 8.33% accuracy improvement for H field prediction and a 60.99% average accuracy improvement in the analytically-calculated E fields over a data-only trained U-Net with the same structure, number of trained weights, and optimization process. The consequential improvement in the E field calculation is a direct result of the problem's physical underpinnings being encoded into the neural network by the WaveY-Net training scheme.

The hybridization of data and physics training enables exceptionally accurate prediction of full-wave electromagnetic fields that can be directly used to perform high level scientific computing tasks, such as the computation of performance gradients for gradient-based device optimization. The adjoint variables optimization method is one such algorithm and calculates the *adjoint gradient* of a particular device in order to iteratively improve its performance with respect to a metric. The computational graph of the adjoint optimization method for metagratings using forward and adjoint WaveY-Net surrogate solvers is illustrated in Fig. 11 (A). For diffractive metagratings, the performance metric is the efficiency for which incident light is routed to a pre-determined angle. Gradients are calculated by evaluating the real dot product of electric fields from forward and adjoint simulations. As demonstrated in Fig. 11 (B), WaveY-Net matches the adjoint gradient calculated via FDFD for a representative metagrating, whereas the purely data-trained U-Net is incapable of predicting the device's electromagnetic responses to a degree of accuracy high enough to faithfully calculate adjoint gradients. Compared to variations of the U-Net trained solely with data and no Maxwell loss, WaveY-Net consistently produces higher-fidelity field predictions, as

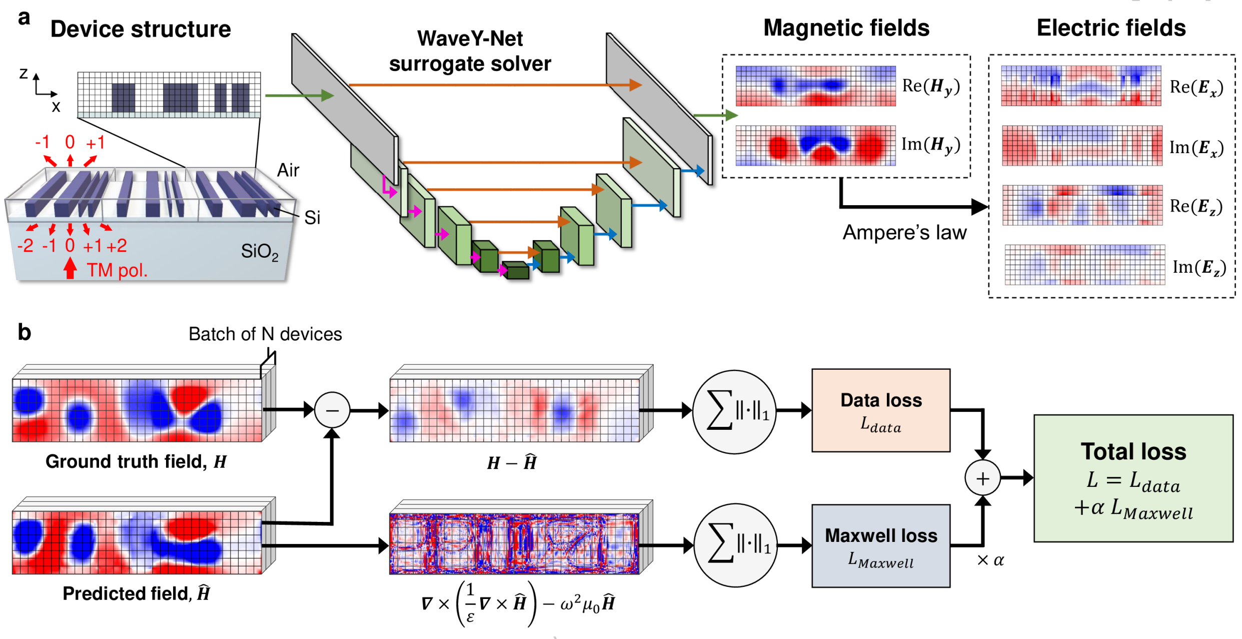

Figure 10: Overview of the WaveY-Net architecture and training technique, as presented in [27]. (A) A U-Net is trained to learn the electromagnetic response of silicon-based metagratings upon normally-incident TM-polarized illumination from the substrate. Given the single-channel input consisting of the dielectric distribution of the input device, WaveY-Net outputs the real and imaginary components of the resulting magnetic field, which is then used to calculate the remaining electric field components using Ampere's Law. (B) The total loss used to train the WaveY-Net using Adam optimization is formulated as a combination of data loss and physics-based "Maxwell loss". The data loss is defined as the MAE between the batch of predicted magnetic fields and their corresponding ground truth, and the Maxwell loss is determined from the network-predicted field using the Helmholtz relation for the magnetic field. A hyperparameter, α, is used to tune the contribution of the Maxwell loss to the total loss. Reprinted from [27]. Licensed under CC BY 4.0.

demonstrated in Figs. 12 (A)-(C). The normalized mean absolute error (MAE) of WaveY-Net's full field prediction for a set of test structures unseen during training is plotted as a function of the Maxwell loss coefficient, α from Eq. 19, in Fig. 12 (D). The result is that the U-Net trained solely with data, when $\alpha = 0$, does not produce electromagnetic response simulations with sufficiently high fidelity to Maxwell's equations for the optimizer to be able to converge to an adequately efficient solution. In contrast, WaveY-Net, with an optimal α coefficient, produces devices with final efficiencies that are equally as high as those produced using FDFD, as demonstrated in Figs. 11 (C)-(D).

Compared to conventional CEM methods, a trained WaveY-Net can solve for electromagnetic fields with speeds faster by orders of magnitude (Fig. 13). This speed advantage is particularly amplified when the fields of multiple devices require evaluation, which can be performed in parallel using WaveY-Net due to its use of GPU hardware. WaveY-Net is thus particularly advantageous for applications requiring multiple batches of simulations, such as population-based optimization algorithms. [34, 35] The computationally efficient, high throughput, physically robust, and general nature of the WaveY-Net platform redefines the quantity of EM simulations and domain sizes that are considered computationally feasible by conventional CEM standards.

6 Conclusions and Perspectives

The rapid growth of machine learning in the last decade paved the way for a new class of CEM solvers that will provide practitioners the opportunity to operate beyond the confines of conventionally formulated solvers. Coupled with a concerted effort within the computational electromagnetics research community to share data and open-source code [36], the conditions are in place for the rapid development of high-performance, quasi-general machine learning-based EM solvers. In this Chapter, we explored illustrative studies of purely machine learning-based surrogate solvers and hybrid conventional CEM-machine learning solvers that can operate orders of magnitude faster than conventional CEM solvers. These speed ups promise to transform CEM: by reducing the simulation and optimization time for electromagnetic devices from hours to seconds, the design and evaluation process is dramatically accelerated. High speed full-wave simulators will enable new classes of devices and systems to be evaluated. For example, hybrid metamaterial-refractive elements have the potential to exhibit new wavefront shaping capabilities, but there is a computational mismatch between conventional wave-based simulators, which are slow and difficult to scale to large domains, and ray optics simulators, which are fast and operate on large scales. High speed surrogate full-wave solvers could eliminate this mismatch and provide a practical route to simulating and optimizing these systems.

The concepts introduced in this chapter represent a relatively nascent, few-year effort, and demonstrations involving, thus far, relatively simple model systems. While promising, the discussed architectures and network training schemes are limiting in ways that make their extension to non-trivial problems a challenge. For example, the state-of-the-art image segmentation networks adapted to CEM are limited to rectilinearly-organized pixels and convolutional operators, which are not directly compatible with irregular meshes. Furthermore, although the loss functions of the segmentation networks can be engineered to incorporate physical intuition into the learned function approximators, the neural network architectures themselves do not explicitly incorporate rigorous physical or mathematical structure tailored for CEM. Much more further research is required to develop purpose-built neural network architectures that can push the boundaries of surrogate solver accuracy, generalizability, and robustness.

For systems involving irregular meshes, such as those in FEM, graph neural networks (GNNs)

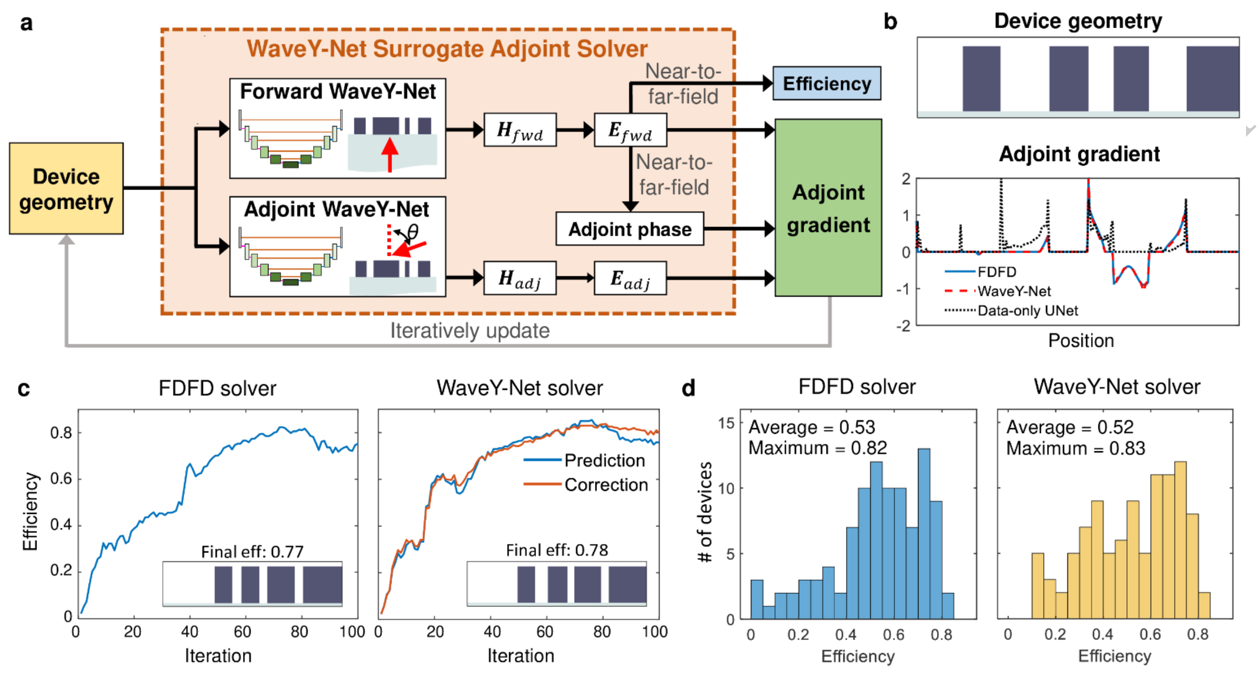

Figure 11: Local freeform metagrating optimization employing WaveY-Net as a surrogate adjoint solver. (A) Computational graph demonstrating the use of WaveY-Nets trained to predict the forward and adjoint responses of metagratings as surrogate solvers in the iterative adjoint optimization method. The field predictions from the separately-trained WaveY-Nets are directly used to calculate the diffraction efficiency and the adjoint gradient, where the latter quantity drives the optimization process. (B) A randomly sampled representative metagrating device layout (top) and its corresponding adjoint gradients (bottom) from the process of local freeform metagrating optimization using WaveY-Net as a surrogate solver. FDFD is used to generate the ground truth gradient values, which are nearly identical to those obtained using WaveY-Net. The data-only U-Net is incapable of capturing the smoothness of the wavelike field solution accurately enough to produce accurate gradients during the adjoint optimization process. (C) The efficiency trajectory of a single metagrating device, comparing the performance of utilizing an FDFD solver in the adjoint optimization process to that obtained using a WaveY-Net surrogate solver. The "prediction" efficiency curve of the WaveY-Net solver plot is directly predicted by WaveY-Net, whereas the "correction" efficiency curve is determined using an FDFD solver for the device optimized solely with the WaveY-Net surrogate solver. (D) Histograms depicting the efficiencies of 100 locally optimized devices, comparing the performance of an FDFD-based optimizer and a WaveY-Net surrogate solver optimizer. Reprinted from [27]. Licensed under CC BY 4.0.

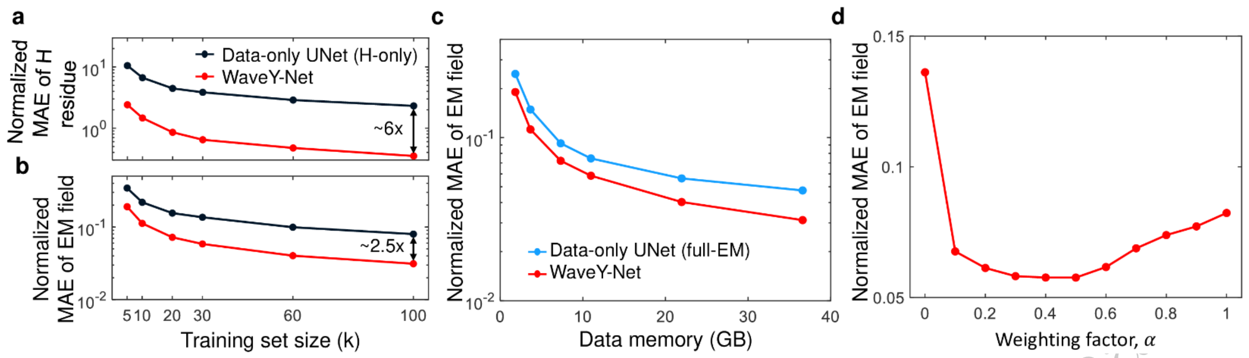

Figure 12: Numerical experiments benchmarking WaveY-Net field prediction performance. (A) Plot contrasting the predicted magnetic field MAE to that of a data-only U-Net that only predicts magnetic fields, for increasing training set size. (B) Plot contrasting the predicted electromagnetic field MAE to that of a data-only U-Net that only predicts magnetic fields, for increasing training set size. The electric field components are analytically calculated from the predicted magnetic fields using Maxwell's equations for both networks. (C) Plot contrasting the MAE of all electromagnetic field components outputted by a full-EM data-only U-Net and a WaveY-Net, plotted for increasing training set size. The data-only U-Net directly predicts all electromagnetic field components, whereas the WaveY-Net predicts the magnetic fields and the electric fields are subsequently calculated using Maxwell's equations. (D) Plot of a trained WaveY-Net's MAE of the predicted magnetic field with respect to the FDFD-generated test set ground truth versus the Maxwell loss coefficient α. The MAE is plotted as a sum of normalized L_{maxwell} and L_{data} loss terms. Reprinted from [27]. Licensed under CC BY 4.0.

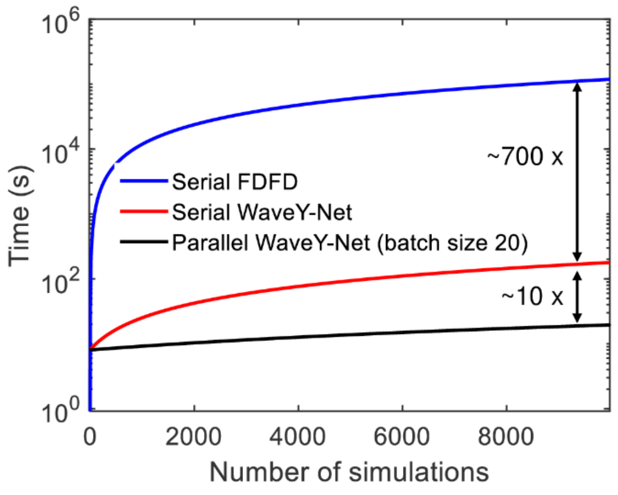

Figure 13: Comparison plot of computation time versus number of simulations for WaveY-Net utilized in serial and parallel modes of operation, benchmarked against serial FDFD. The parallel WaveY-Net processes batches of 20 devices simultaneously. Reprinted from [27]. Licensed under CC BY 4.0.

are a newly developed class of neural networks that promise to be an active area for machine learning-based CEM research. [33, 37] GNNs are designed to process data that are represented on graphs, which are most generally composed of sets of vertices called *nodes* that are connected by a set of directed *edges*. Graphs therefore lend themselves to naturally represent meshes, which can either store electromagnetic field values directly or a set of basis functions that are subsequently used to determine the fields of interest. GNNs support the framework to process and learn regressive functions for data on graph structures, through operations that propagate information from the nodes, along the edges, in a spatially invariant manner (i.e., in a way that is applicable to any graph structure, with any number of neighboring nodes). This is similar to CNNs, which apply the convolutional operation to images composed of pixels, except that the CNN framework requires a fixed number of neighboring pixels for each pixel targeted by a convolutional operation. The GNN framework has already demonstrated its success in learning to make accurate predictions for incompressible fluid dynamics simulations from data only, [38] which indicates that they are also likely to be able to learn physics problems that are governed by Maxwell's equations. GNNs thus unlock a new machine learning paradigm for mesh-based CEM that, at the time of writing, has not been substantively explored.

Fourier neural operators (FNOs) offer a novel formulation of neural networks that is capable of learning the mapping between the parameter space of a PDE directly to its solution space, rather than learning a discretized relationship for a single instance of a PDE. [39] The FNO is mesh-independent, as it learns a continuous function rather than the discretized weight matrices of conventional deep learning methods, making it a more flexible tool than discretizing conventional CEM methods, including FEM and FD techniques. This unlocks the ability to train the model using data on arbitrarily-connected meshes and evaluate it on a completely different mesh type. The main working principle of the FNO is that it learns sets of global sinusoidal kernel functions that are integrated over the kernel's entire input domain, in contrast to the CNN's method of learning local kernels that are spatially convolved across the domain. The integration of the learned global kernel function across its entire input domain is performed using a Fourier transform, resulting in a very quick and efficient algorithm. Given that the inputs and outputs of PDEs are continuous, the FNO technique is ideal for learning the solution space of PDEs. The FNO has been successfully applied to a variety of incompressible fluid problems constrained by sets of PDEs, [39] suggesting that the method is extendable to Maxwell's equations. The FNO is a sensible contender to succeed the state-of-the-art segmentation networks utilizing spatial convolutional operators currently powering most machine learning electromagnetics solvers.

Acknowledgments

We thank Jiaqi Jiang and Der-Han Huang for insightful discussions. Robert Lupoiu is supported by a graduate fellowship award from Knight-Hennessy Scholars at Stanford University. The authors acknowledge support from the National Science Foundation under award no. 2103301 and ARPA-E with award no. DE-AR0001212.

References

[1] Karl Eugen Kurrer. The History of the Theory of Structures: From Arch Analysis to Computational Mechanics. 2009.

[2] Yogesh Jaluria and Satya N. Atluri. Computational heat transfer, 1994.

[3] Heiner Igel. Computational Seismology. 2017.

[4] Thomas Rylander, Par Inglestrom, and Anders Bondeson. Computational Electromagnetics. Springer, 2 edition, 2013.

[5] T. Hubing, C. Su, H. Zeng, and H. Ke. Survey of current computational electromagnetics techniques and software, 6 2009.

[6] Xin Qing Sheng and Wei Song. Essentials of Computational Electromagnetics. 2012.

[7] Jian-Ming Jin. The finite element method in electromagnetics, 3rd edition. Journal of Chemical Information and Modeling, 2014.

[8] Kane S. Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. IEEE Transactions on Antennas and Propagation, 1996.

[9] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. IBM Journal of Research and Development, 11(2):215–234, 1967.

[10] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 7, 1986.

[11] Jörg Liesen and Zdenek Strakos. Krylov Subspace Methods: Principles and Analysis, volume 9780199655410. 2013.

[12] S.D. Gedney. An anisotropic perfectly matched layer-absorbing medium for the truncation of fdtd lattices. IEEE Transactions on Antennas and Propagation, 1996.

[13] Daniel S. Katz and Allen Taflove. Validation and extension to three dimensions of the berenger pml absorbing boundary condition for fdtd meshes. IEEE Microwave and Guided Wave Letters, 1994.

[14] Proc. Study on a Recurrent Convolutional Neural Network Based FDTD Method, International Applied Computational Electromagnetics Society Symposium - China (ACES). IEEE, 2019.

[15] Proc. Machine Learning Based Neural Network Solving Methods for the FDTD Method, International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting. IEEE, 2018.

[16] He Ming Yao and Lijun Jiang. Machine-learning-based pml for the fdtd method. IEEE Antennas and Wireless Propagation Letters, 2018.

[17] Cam Key and Branislav M. Notaros. Data enabled advancement of computation in engineering: A robust machine learning approach to accelerating variational methods in electromagnetics and other disciplines. IEEE Antennas and Wireless Propagation Letters, 2020.

[18] Cam Key and Branislav M. Notaros. Predicting macro basis functions for method of moments scattering problems using deep neural networks. IEEE Antennas and Wireless Propagation Letters, 2021.

[19] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review, 2018.

[20] Huajun Liu, Fuqiang Liu, Xinyi Fan, and Dong Huang. Polarized self-attention: Towards high-quality pixel-wise regression. ArXiv, 7 2021.

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. volume 07-12-June-2015, 2015.

[22] Wei Tang, Tao Shan, Xunwang Dang, Maokun Li, Fan Yang, Shenheng Xu, and Ji Wu. Study on a poisson's equation solver based on deep learning technique. volume 2018-January, pages 1–3. Institute of Electrical and Electronics Engineers Inc., 1 2018.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. volume 9351, 2015.

[25] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2012.

[26] Peter R. Wiecha and Otto L. Muskens. Deep learning meets nanophotonics: A generalized accurate predictor for near fields and far fields of arbitrary 3d nanostructures. Nano Letters, 20, 2020.

[27] Mingkun Chen, Robert Lupoiu, Chenkai Mao, Der-Han Huang, Jiaqi Jiang, Philippe Lalanne, and Jonathan A. Fan. High speed simulation and freeform optimization of nanophotonic devices with physics-augmented deep learning. ACS Photonics, 9(9):3110–3123, 2022.

[28] Shutong Qi, Yinpeng Wang, Yongzhong Li, Xuan Wu, Qiang Ren, and Yi Ren. Two-dimensional electromagnetic solver based on deep learning technique. IEEE Journal on Multiscale and Multiphysics Computational Techniques, 5, 2020.

[29] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 2019.

[30] Yinhao Zhu, Nicholas Zabaras, Phaedon Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Journal of Computational Physics, 394, 2019.

[31] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. Physics-informed neural networks with hard constraints for inverse design. Figure license: https://creativecommons.org/licenses/by-nc-nd/4.0/, 2021.

[32] Dimitri P Bertsekas. Constrained optimization and Lagrange multiplier methods. Academic press, 2014.

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4–24, 2021.

[34] Jiaqi Jiang and Jonathan A. Fan. Global optimization of dielectric metasurfaces using a physics-driven neural network. Nano Letters, 19(8):5366–5372, 2019. PMID: 31294997.

[35] Jiaqi Jiang and Jonathan A. Fan. Simulator-based training of generative neural networks for the inverse design of metasurfaces. Nanophotonics, 9(5):1059–1069, nov 2019.

[36] Jiaqi Jiang, Robert Lupoiu, Evan W. Wang, David Sell, Jean Paul Hugonin, Philippe Lalanne, and Jonathan A. Fan. Metanet: a new paradigm for data sharing in photonics research. Opt. Express, 28(9):13670–13681, Apr 2020.

[37] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. AI Open, 1, 2020.

[38] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In International Conference on Learning Representations, 2021.

[39] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020.