Progress In Electromagnetics Research M, Vol. 136, 33–45, 2025

PIER M

(Received 25 August 2025, Accepted 11 November 2025, Scheduled 17 November 2025)

Enhanced Low-Resolution Contrast Operator Using Neural Networks for E-Polarized EM Scattering Problems

Daan van den Hof*, Martijn C. van Beurden, and Roeland J. Dilz

Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands

ABSTRACT: Coarse discretization introduces significant errors in the solution of scattering problems, in part due to discretization errors in the contrast operator. We present a procedure for the automatic construction of a modified contrast operator for electromagnetic scattering problems by using trainable neural networks to represent a modified contrast operator. We achieve a higher accuracy on a coarse discretization while still keeping computation time down compared to a fine discretization. By using synthetic data from a full-wave Maxwell solver to train the network for one-dimensional slab scatterers and two-dimensional polygonal scatterers, we are able to use the techniques found in deep learning to improve accuracy in coarse-grid forward scattering problems.

1. INTRODUCTION

The modeling of electromagnetic (EM) scattering by dielectric objects is relevant to many applications. When the scattering object is compact and purely dielectric, the governing equations can be cast into a domain integral equation that involves an integral over a Green function and the scatterer. Furthermore, when the background medium is homogeneous, this integral becomes a simple convolution integral. There are several ways to solve such an integral equation, a prime example being the method of moments (MoM) [1]. Generally, any numerical method will start with a discretization step that turns the continuous (linear) problem into a discrete linear system. Solving this linear system will result in a discretized solution to the original scattering problems. For domain integral equations, these systems are generally dense and large. This makes an iterative solver, such as BICGStab [2] or GMRES [3], the best approach to solve the system.

The computational costs of solving these linear systems may be prohibitively high, even when using iterative linear solvers. However, there are a few strategies to mitigate this. The first of these strategies is to reduce the computational complexity of the matrix vector product used in the iterative solver. In the conjugate-gradient fast Fourier transform (CGFFT) [4], the spatial convolution integral is replaced by a discrete convolution which is then computed using fast Fourier transforms. This principle is extended in the spatial-spectral method [5–8] where simultaneous spatial and spectral discretizations in combination with efficient semi-analytical Fourier transformations are used to completely represent the convolution integral in the spectral domain. This results in a method that is both computationally efficient and flexible in its formulation. Other methods improve the matrix-vector product by making use of problem-specific information. Approaches such as the multi-level fast multipole algorithm [9] or the fast inhomogeneous plane wave algorithm [10, 11], are examples of this. The effects of local variations in the field vanish at large distances. These methods use this fact to construct low-rank representations of large-distance interactions, reducing the computational cost of the matrix-vector product.

A second approach to reduce the computation time is by simply reducing the number of samples taken in the discretization process. This will lead to a lower accuracy in the final result, but might lead to significant speedups. One way to mitigate the loss in accuracy is by using a non-equidistant sampling to make more efficient use of the number of unknowns. As an example, in [12] local discretization refinements around the scatterer geometry are used so that a higher accuracy is achieved compared to a uniform discretization with the same number of unknowns. One limitation of this approach is the requirement to find a suitable discretization mesh for any given scattering geometry. This procedure requires problem-specific formulations that might be infeasible to construct in practice. For this reason, we seek to automate this process through the use of neural networks.

Neural networks, and by extension deep learning, have reached wide use in many different fields over the past decade. Although most prominent in applications such as image processing [13–15], speech recognition [16] and, more recently, large language models [17], they have also made waves in the research areas of computational physics and computational EM. In [18], the idea of the physics informed neural network (PINN) was formalized. These networks seek to solve differential equations by embedding boundary conditions, initial conditions, or other physical constraints directly into their cost function. By mixing these physics-based losses with the losses from data, network training generalizes better outside of the training data. More specifically for EM in [19, 20], the finite-difference time-domain (FDTD) [21] method was formulated in terms of convolutional neural networks (CNNs) [22] and

* Corresponding author: Daan van den Hof (d.van.den.hof@tue.nl).

recurrent neural networks (RNNs) [23, 24]. In [25], a fully connected network was used to train perfectly matched layers to terminate an FDTD domain with a single cell boundary. Using deep learning to reduce the number of unknowns in computational EM problems was explored in [26] for the finite element method [27] and in [28] for MoM. In [29], an iterative neural network structure was employed to enhance the conjugate-gradient algorithm in a forward scattering problem, this forward solver was then used in [30] to solve inverse scattering problems. Finally, one-step field predictions using neural networks have been employed in [31, 32].

We seek to reduce the discretization error for coarser discretizations while maintaining a lower computational cost compared to that of a finer discretization. We do this by replacing the coarse-discretization dielectric contrast operator with a modified, trainable, contrast operator. The structure of this operator is found with a deep learning approach using neural networks. We seek the same general approach for one- and two-dimensional (1D and 2D) scattering problems. The trained operator plays a role similar to the local refinement methods mentioned above, with the difference being that it acts on both the electric field and the contrast function. Furthermore, this learned approach provides a more general framework for accomplishing this and does not require manual analysis of specific geometries. We use the spatial-spectral method as the basis for our formulation, since it gives us access to a fast matrix-vector product. Furthermore, we use the discretization library described in [33] to implement the method in a convenient high-level interface that allows for fast and flexible experimentation.

The paper is organized as follows. In Section 2, we introduce the scattering problem and the errors due to coarse discretization. We discuss the proposed method of using a trainable linear operator with which we solve the coarse system. In Section 3, we go into the structure of the network that produces this operator. In Section 4, we discuss the implementation details, specifically how the introduced method is integrated with the discretization library and TensorFlow [34]. We discuss network training in Section 5. The results using trained networks for the 1D and 2D cases are shown in Section 6. Finally, in Section 7, we draw conclusions.

2. PROBLEM FORMULATION

2.1. The Scattering Integral Equations

We are interested in EM scattering by a-periodic dielectric scatterers embedded in a homogeneous translation-invariant background. This means that the complete scattering setup can be characterized by an incident electric field and a function describing the scatterer. The problem is then to find the total field resulting from this wave-material interaction. Here, we will limit ourselves to scalar-valued problems in a homogeneous background medium. For EM, this leaves 1D and 2D E-polarized scattering problems. We use an $e^{i\omega t}$ time convention, where ω is the angular frequency. With this convention, the scattering problem can be cast into the integral equation.

$$E^i(\mathbf{x}) = E(\mathbf{x}) - \sqrt{2\pi}^{-\nu} \int_D J(\mathbf{x}')G(\mathbf{x}-\mathbf{x}')\mathbf{dx}', \quad (1)$$

where $\nu = \{1, 2\}$ is the number of spatial dimensions; ε_0 is the permittivity of free space; E, E^i are the total and incident electric field, respectively; D is the domain of the object(s) on which the permittivity is different from that of the background; and $G(\mathbf{x})$ is the spatial Green function. The contrast current density is given by

$$J(\mathbf{x}) = i\omega\varepsilon_0\chi(\mathbf{x})E(\mathbf{x}), \quad (2)$$

where $\chi(\mathbf{x}) = \varepsilon_r(\mathbf{x}) - 1$ is the contrast function. We use the following ν-dimensional Fourier transformation pairs

$$\begin{aligned} \hat{f}(\mathbf{k}) &= \mathcal{F}(f(\mathbf{x})) = \frac{1}{\sqrt{2\pi}^{\nu}} \int_{\mathbb{R}^\nu} f(\mathbf{x})e^{i\mathbf{k}\cdot\mathbf{x}}d\mathbf{x}, \\ f(\mathbf{x}) &= \mathcal{F}^{-1}\left(\hat{f}(\mathbf{k})\right) = \frac{1}{\sqrt{2\pi}^{\nu}} \int_{\mathbb{R}^\nu} \hat{f}(\mathbf{k})e^{-i\mathbf{k}\cdot\mathbf{x}}d\mathbf{k}. \end{aligned} \quad (3)$$

Here we introduce the convention that spectral-domain quantities are denoted with a hat ($\hat{\ }$). The $(\cdot)$ used in the exponent is the standard ν-dimensional Euclidean scalar product. Throughout this work, we will use plane waves for the incident fields, these take the form

$$E^i(\mathbf{x}) = e^{i\mathbf{k}_i\cdot\mathbf{x}}, \quad (4)$$

with $|\mathbf{k}_i| = \sqrt{\mathbf{k}_i^2} = k_0$ where $k_0 = \omega\sqrt{\varepsilon_0\mu_0}$ is the free-space wavenumber where μ_0 is the permeability of free space. In the spatial spectral method [5–8], (1) is transformed to the spectral domain using Fourier transformations, which yields

$$\hat{E}^i(\mathbf{k}) = \hat{E}(\mathbf{k}) - \hat{G}(\mathbf{k})\mathcal{F}\left[\chi(\mathbf{x})\mathcal{F}^{-1}\left[\hat{E}(\mathbf{k})\right]\right]. \quad (5)$$

In this formulation, the spatial convolution integral is replaced by a spectral multiplication with the Green function. In the spectral domain, the Green function takes the form

$$\hat{G}(\mathbf{k}) = \frac{1}{\mathbf{k}^2 - k_0^2}. \quad (6)$$

To solve (5) for $\hat{E}$, we set up a discretized version of the problem and construct a linear system. We then use an iterative solver such as GMRES or BiCGstab [2, 3] to solve the linear system. For the discretization process, we make use of the discretization library described in [33]. This gives us access to the required Fourier transformations, as well as a built-in method to move data between different discretization methods. Throughout this work, we use the windowed Fourier-series based discretization method described in [35].

2.2. Discretization Error Due to Coarse Sampling of the Contrast Operator

Discretization of the integral equation introduces errors in the final result. The magnitude of these errors will depend on the coarseness of the discretization; in general, a denser sampling leads to a more accurate result. In contrast, coarser discretizations will require fewer computational resources. Our goal is to compare the results obtained using a *fine* discretization to those obtained using a *coarse* discretization and modify the contrast operator to minimize this difference. Discrete quantities and

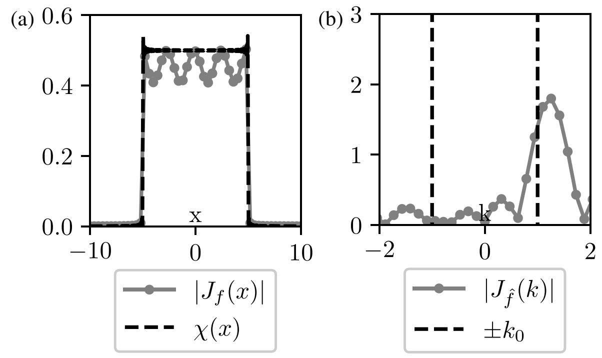


FIGURE 1. The contrast current density for a 1D scattering problem with a slab scatterer for both (a) spatial and (b) spectral domain. The parameters for the setup are shown in Table 1.

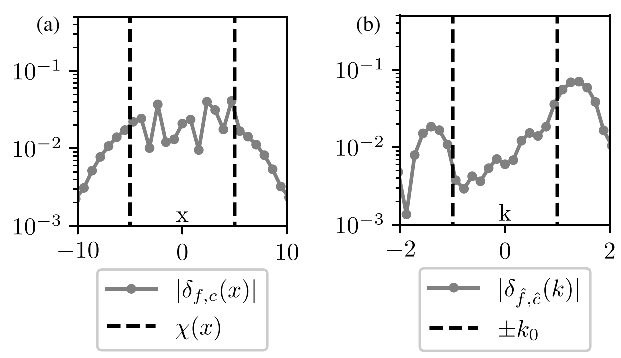


FIGURE 2. Differences in the contrast current density between a fine (f) and a coarse (c) discretization, we show results for both (a) spatial and (b) spectral domain. The discretization parameters are found in Table 1.

TABLE 1. Simulation and discretization parameters for the fine and coarse discretization methods. The discretization method and its parameters are described in [35].

Simulation parameters	
Scatterer location	$[-5, 5]$
ε_r	1.5
k_i	-1
Discretization parameters	(fine, coarse)
N	(1001, 51)
Δ_x	(0.040, 0.78)
$e_{x/k}$	(0.5, 0.5)
$t_{x/k}$	4

operators that are used on both discretizations will be denoted using the subscripts f, c for the fine and coarse discretizations, respectively. As an example, we introduce the discretization operators $\mathcal{D}_c(g)$, $\mathcal{D}_f(g)$, which construct a discrete representation of the continuous function g using discretizations c and f. Similarly to the continuous case, we will use a ˆ to indicate a spectral discretization.

In many cases, we are interested in the far-field information for a given scattering problem. This far-field information is completely contained in the contrast current density. For this reason, finding an accurate solution for $\hat{J} = \mathcal{F}\chi\mathcal{F}^{-1}\hat{E}$ will be our primary goal. Here, $\mathcal{F}\chi\mathcal{F}^{-1}$ is the unmodified contrast operator. We define

$$\delta_{\hat c} = \mathcal{D}_{\hat c}(\hat J) - \mathcal{F}_c\chi_c\mathcal{F}_{\hat c}^{-1}E_{\hat c} = \mathcal{D}_{\hat c}(\hat J) - \hat J_{\hat c}, \tag{7}$$

as the object that represents the difference between the approximate and the "true" solutions of the problem. Imposing a norm on this object then provides a metric for the discretization error. It is important to note that when we write implied multiplications between discretized objects, such as $\chi_c(\mathcal{F}_{\hat c}^{-1}E_{\hat c})$, we are denoting element-wise multiplication [35]. We would like to emphasize that in these contexts quantities like χ_c should be thought of as operators rather than as vector-like objects.

In practice, we do not have access to $\hat J$, since for a generic scattering setup there is no analytical solution. However, in general, we may assume that a finer discretization will result in a smaller discretization error at the cost of an increase in computation time. We let the *fine* discretization be the discretization that is "close enough" to the true solution for these purposes. Using this definition, we can let f stand in for the continuous case and define the discretization error as

$$\left\|\delta_{\hat f,\hat c}\right\| = \left\|R_{\hat f,\hat c}\delta_{\hat f} - \delta_{\hat c}\right\| = \left\|R_{\hat f,\hat c}\hat J_{\hat f} - \hat J_{\hat c}\right\|. \tag{8}$$

Here, $\|\cdot\|$ refers to some norm over the input vector, indicating a notion of distance between the coarse and fine discretizations. We have also introduced the resampling operator $R_{\hat f,\hat c}$, that is used to bring quantities from the fine to the coarse discretization.

If $\|\delta_f\|$ is sufficiently small, we may then say $\delta_{f,c} \approx \delta_c$. In Fig. 1, we show the contrast current density for a simple 1D scattering problem with a slab scatterer in the center, in both spatial and spectral domains, calculated for the fine discretization. In Fig. 2, we show the difference between a fine and coarse discretization in the same scattering problem. The pertaining simulation parameters are given in Table 1. For a computational domain with size $-20 \le r_x \le 20$ for both dimensions, these parameters correspond to roughly 8 and 160 samples per wavelength for the coarse and fine discretization, respectively. From Fig. 2, we see that the coarse discretization leads to an error over the entire spatial support of the scatterer. This error is largely due to an imperfect discretization of the contrast operator. Since χ is compact in the spatial domain, it has an infinite extent in the spectral domain. Since the spatial spectral discretization method necessarily truncates both spatial and spectral domain, this infinite spectral support gets truncated, which leads to errors throughout the spatial support.

2.3. The Modified Contrast Operator

In solving the scattering problem we want to keep computation times short. However, as demonstrated, taking a coarser discretization comes at the cost of a reduced accuracy in the final result. In the spatial spectral method, we have a (near) exact representation of $\hat{G}$; therefore, we can say that the error must come from the contrast operator $\mathcal{F}_c\chi_c\mathcal{F}_{\hat{c}}^{-1}$. A natural question that arises from this observation is whether this operator is the best choice when representing the problem on a coarse grid or if we can find a better alternative. We seek to answer this question by introducing a modified contrast operator, A. With this operator we have

$$\tilde{J} = A\tilde{E}, \tag{9}$$

where $\tilde{J}$ is the modified contrast current density, and $\tilde{E}$ is the corresponding electric field. For compactness, we omit subscripts for these quantities since they are only defined on the coarse discretization. The goal is to find the A operator such that the discretization error

$$\begin{aligned}\left\|\delta_{\hat{f},A}\right\| &= \left\|R_{\hat{f},\hat{c}}\mathcal{F}_f\chi_f\mathcal{F}_{\hat{f}}^{-1}E_{\hat{f}} - A\tilde{E}\right\| \\ &= \left\|R_{\hat{f},\hat{c}}J_{\hat{f}} - \tilde{J}\right\|,\end{aligned} \tag{10}$$

is minimized. This new operator yields the corresponding linear system

$$\left(I_{\hat{c}} - G_{\hat{c}}A\right)\tilde{E} = E^i_{\hat{c}}. \tag{11}$$

We still want to solve this system using an iterative solver, which means that $A\tilde{E}$ has to be linear in $\tilde{E}$. The use of an iterative solver also implies that we want to keep the execution time of the matrix-vector product $A\tilde{E}$ low, since it occurs at every iteration. Importantly, even though A is a linear operator in E, it does not need to be linear with respect to χ_c.

The original coarse system already approximates the fine system and therefore serves as a good starting point for finding A. We will assume that we can write A as the original contrast operator with a correction term that depends on χ_c, that is,

$$A(\chi_c) = \mathcal{F}_c\left(\chi_c + A_1(\chi_c)\right)\mathcal{F}_{\hat{c}}^{-1}, \tag{12}$$

where $\mathcal{F}_c\chi_c\mathcal{F}_{\hat{c}}^{-1}$ will be the dominant contribution.

Identifying a suitable candidate for the operator A explicitly is a challenge, especially if we want to do this for a broad class of scattering problems. For this reason, we will look at a learned approach by representing A with a neural network and training it using synthetically generated data.

3. OPERATOR NETWORK

By construction, the operator A acts as a modified contrast operator. The errors in $J_{\hat{c}}$, although significant, are still small compared to the field magnitude. This implies that the original contrast operator can serve as an adequate starting point for determining A. This would also be true for different classes of scattering problems (e.g., 3D, vector-valued functions, or periodic problems), which leads to a generally applicable procedure for constructing A. We decided to implement the trainable part of A, namely A_1, as two separate networks

$$A_1(\chi_c)\mathcal{F}^{-1}\tilde{E} = \sum_{n=1}^{N}\left(A_\chi(\chi_c)\right)_n\left(A_E\mathcal{F}^{-1}\tilde{E}\right)_n, \tag{13}$$

where A_χ produces N contrast functions, and A_E produces $N\hat{E}$ fields. These learned mappings are multiplied to obtain N contrast current contributions $\tilde{J}_n$ $(n \in \{1, 2, ..., N\})$. Note that if A_E is linear, then the entire operator is linear in $\tilde{E}$. The resulting contrast current contributions are summed and added to the term $\tilde{J}_0 = \chi_c\mathcal{F}_{\hat{c}}^{-1}\tilde{E}$, to arrive at the final modified contrast current density. The entire network representing A is shown in Fig. 3.

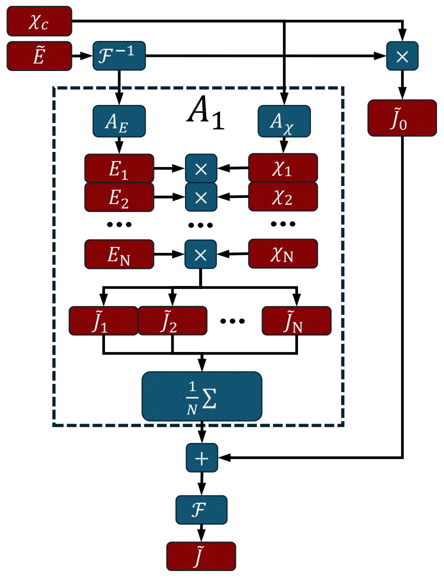


FIGURE 3. The contrast operator is a sum between the standard contrast multiplication with a number of additional trainable mappings of $\tilde{E}$ and χ being multiplied and added.

3.1. Network Considerations

The A_E network acts directly on the E field and therefore must be linear. When solving (11) using an iterative solver, we will have to reapply A_E every iteration, as this input field changes every iteration. This means that a quick execution time of this operator is of particular importance, since it may be incurred hundreds of times for a given system. For this reason, we chose to use a simple CNN [22] consisting of just one 1D convolution layer per dimension.

Unlike A_E, A_χ does not act on the input field $\hat{E}$. In the training of the network this makes no difference, but in the inference step, when the operator is called as part of the iterative solver, it means that we can precompute all the χ_n terms, since χ itself does not change. This fact, combined with the absence of a linearity requirement, allows great freedom for the structure of A_χ. We would like this operator to be generally applicable, and it should be somewhat independent of the location and shape of the scatterer. This means that the operator should preferably be a spatially local transformation. For these reasons we again choose a CNN. However, unlike for the A_E network, we are free to use a large number of layers as well as nonlinear activation functions.

Both A_E and A_χ produce N outputs that are then multiplied element-wise. This requires these outputs to have a consistent size, which is accomplished by zero padding after the convolution operation. This zero padding may lead to artifacts at the edges of the domain, and to ensure that this does not affect the end result, we multiply by a window function that smoothly decays to 0 at the edge of the spatial domain.

We have limited ourselves to fully convolutional networks without any pooling or normalization layers. For nonlinear activations, we have used the tanh function. The only exception to this is the activation function corresponding to the channels in the last CNN layer that represent the imaginary part of A_χ. For this layer, we chose the swish [36] function to prevent unbounded negative growth, since this would correspond to a medium with passive gain, which is not physical and, more importantly, could potentially result in unstable behavior.

We note that our choice of the fully convolutional network may not be the optimal solution, especially for training efficiency. However, in experimenting with more elaborate networks, we have not observed improvements in the performance as compared to the CNN, for a similar number of parameters.

4. IMPLEMENTATION

The programming for this work was performed in Python 3.12, where we used numpy [37] for data processing and matplotlib [38] for creating plots. We use TensorFlow (version 2.16.1) [34] to implement neural networks.

4.1. Complex Layers and Functions

The scattering problems discussed here are complex-valued. Although TensorFlow has some support for complex numbers, at the time of writing the current version has not added this support to all relevant functions, e.g., convolutions.

The operator A consists of the standard contrast operator $\mathcal{F}_c\chi_c\mathcal{F}_{\hat{c}}^{-1}$ and a trained operator A_1, which functions as a set of modified contrast operators. This trained part consists of an "E network" (A_E) and a "χ network" (A_χ), both of which have several sequential TensorFlow layers. In general, we treat the real and imaginary parts of a discretized object as separate channels in a TensorFlow tensor. For example, in a 2D problem, where we have batch size B, we would have real-valued tensors of shape $B \times N_x \times N_y \times 2$ instead of complex-valued tensors of shape $B \times N_x \times N_y \times 1$. Consequently, we need to define custom layers and functions that can handle this format. We attempted to keep these layers as general as possible, with the dimension on which we store the complex/real channels being user configurable. The list of functions that we added can be seen in Table 2. In particular, this formulation implies that the output channels of A_E and A_χ are evenly split into real and imaginary components.

TABLE 2. Implemented functions for use with TensorFlow.

Function	Description
`SplitComplex`	Get T_a, T_b for $T = T_a + iT_b$.
`CombineComplex`	Get T given T_a, T_b.
`MultComplex`	Multiply complex T_1, T_2.
`ConvComplex1D`	1D complex convolution.
`ConvComplex2D`	2D complex convolution.
`RMSComplex`	Return RMS of complex T.

4.2. Interaction with the Discretization Library

Discretizing the prescribed quantities, as well as solving the pertaining linear system, is performed by the discretization library [33]. Within this library, data is represented in a *data object* in an abstract manner, while the underlying discretization method is defined by an associated *discretization object*. To combine this"classical" part of the solver with the "neural network" part, we need to define smooth conversions between TensorFlow and the discretization library.

Both of these pieces of software provide direct Python interfaces that call computationally efficient binaries. Within the discretization library, the specifics of how data is represented depends on the discretization method used. However, every discretization method implements the `get_samples`, `from_samples` and `get_nodes` methods. The former two give a generic way to convert between data objects and Numpy arrays, while the latter method provides the corresponding grid coordinates. Using these methods, we define functions that convert between complex-valued data objects and TensorFlow tensor objects that contain channels for the real and imaginary parts of the number. Therefore, there is always a mapping between library objects and a 2D grid with uniform spacing, regardless of the underlying discretization method.

4.3. Fourier Transformations

The operator A partly acts on spatial-domain quantities. Since $\tilde{E}$ and $\tilde{J}$ are spectral quantities, we require forward and inverse Fourier transformations in the operator. Since A is trained using backpropagation, we also need to include the Fourier transformations in the gradient graph. For consistency within the EM solver, we also want these transformations to be the same as the one used in the discretization library. For this reason, we implement the forward and inverse transformations as a TensorFlow layer.

TensorFlow provides the definition of layers with a custom gradient function. We build these layers around the Fourier transformations of the discretization library. Although the specifics of these transformations depend on the underlying discretization method, we can treat them as a black-box for our purposes. The transformations are linear operators acting on finite-dimensional objects, which means we can conceptually represent them by matrices. This means that we can write

$$\begin{aligned} \frac{\partial}{\partial x_c}(\mathcal{F}_c x_c) &= \mathcal{F}_c, \\ \frac{\partial}{\partial k_{\hat{c}}}(\mathcal{F}_{\hat{c}}^{-1} k_{\hat{c}}) &= \mathcal{F}_{\hat{c}}^{-1}. \end{aligned} \tag{14}$$

In the TensorFlow backpropagation, the up-stream gradient, g_u is left-multiplied to obtain the down-stream gradient g_d. This means that for the forward transformation we can write

$$g_d = g_u \mathcal{F}_c = \left(\mathcal{F}_c^H g_u^H\right)^H, \tag{15}$$

where H is the Hermitian transpose. Generally, for a matrix we have

$$\langle A\mathbf{x}, \mathbf{y}\rangle_d = \langle \mathbf{x}, A^H \mathbf{y}\rangle_d, \tag{16}$$

where $\langle , \rangle_d$ denotes an inner product on ℓ^2. For continuous Fourier transformations we have

$$\left\langle \mathcal{F}(f(x)), \hat{h}(k) \right\rangle_{\mathbb{C}} = \left\langle f(x), \mathcal{F}^{-1}\left(\hat{h}(k)\right) \right\rangle_{\mathbb{C}}, \tag{17}$$

where $\langle , \rangle_{\mathbb{C}}$ is an L^2 inner product on a complex vector space. We will assume that the following approximation holds for the discrete Fourier operators

$$\begin{aligned} \Delta_x \langle \mathcal{F}_c[f_c], h_{\hat{c}} \rangle_d &\approx \left\langle \mathcal{F}(f(x)), \hat{h}(k) \right\rangle_{\mathbb{C}} \\ \Delta_k \left\langle f_c, \mathcal{F}_{\hat{c}}^{-1}[h_{\hat{c}}] \right\rangle_d &\approx \left\langle f(x), \mathcal{F}^{-1}(\hat{h}(k)) \right\rangle_{\mathbb{C}} \end{aligned} \tag{18}$$

with $f_c = \mathcal{D}_c(f(x))$, $h_{\hat{c}} = \mathcal{D}_{\hat{c}}(\hat{h}(k))$ and Δ_x, Δ_k spatial and spectral sampling distances, respectively. This assumption, combined with (17) suggests

$$\mathcal{F}_c^H \approx \left(\frac{\Delta_x}{\Delta_k}\right)^\nu \mathcal{F}_{\hat{c}}^{-1}. \tag{19}$$

So we can write the gradient function for the forward or inverse Fourier transformations in terms of the inverse or forward transformations. The complete procedure for implementing the Fourier operators as layers in TensorFlow is visualized in Fig. 4. The procedure for the inverse Fourier transformation is similar.

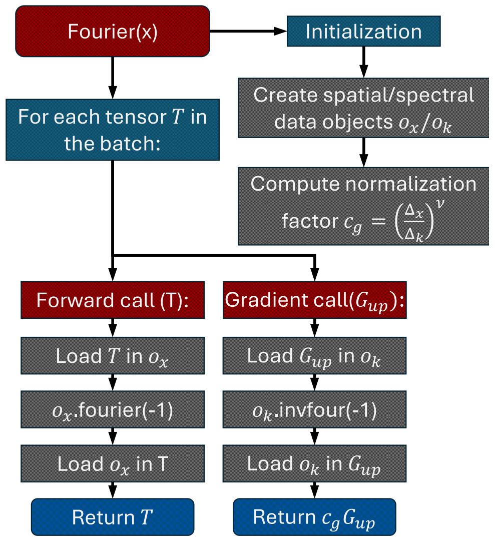


FIGURE 4. High level depiction of the discretization libraries Fourier transformation implemented as TensorFlow layer. In this process the incoming data is stored inside the spatial/spectral data objects o_x/o_k, from the discretization library, which are then used to perform the Fourier transformation. The statement $o_x \cdot$ `fourier`(-1) is part of the discretization and means that the contents of o_x are Fourier transformed over all dimensions (-1).

5. TRAINING THE NETWORK

We train the network through supervised learning using backpropagation. That is, we generate synthetic inputs and compare the output to the solutions obtained on the fine grid.

5.1. Loss Function

By design, the operator A has two input arguments: an electric field $\hat{E}$ and contrast function χ. The output of the network should be the modified contrast current density $\tilde{J}$. The final goal is to minimize the error in (10), for which we need to define a loss function that, for the purposes of gradient descent, should be continuously differentiable. We also have to take into account both the real and imaginary components. For these reasons, we decided to use a complex root mean square (RMS) function for the loss. This cost function is a vector norm, meaning $L(x, y) = \|x - y\|$, and minimizing $L(\delta_{f,c}, 0)$ is equivalent to minimizing $\|\delta_{f,c}\|$. It is given by

$$L(x, y) = \sqrt{\sum_i \frac{(x_i - y_i)(x_i - y_i)^*}{N_s}}, \tag{20}$$

where $*$ represents the complex conjugation, and N_s is the total number of elements in x and y. An important property that we will exploit later is that this loss function satisfies $L(x, y) = L(x - y, 0)$.

5.2. Data Generation

Training the network requires both input and output data. The inputs to the network are the field $\hat{E}$ in a given iteration and the contrast function χ, while the output will be a contrast current density $\tilde{J}$. The target contrast current density is calculated on the fine grid using the spatial spectral method without any learned operators. Then it is down-sampled to the coarse grid using the $R_{\hat{f},\hat{c}}$ operator. We consider simple slab scatterers for the 1D case and (a subclass of) convex polygonal scatterers for the 2D case [39], as well as a concave hourglass-shaped scatterers. For the incident waves, we use unit-amplitude plane waves. Thus, every scattering problem can be characterized by a set of scattering parameters and the angle of incidence of the wave. We randomly generated these parameters within a specified range before running the simulations.

5.3. Training Target

Our goal is to minimize the error in (10). To do this, we can calculate the loss using $L(R_{\hat{f},\hat{c}} J_{\hat{f}}, \tilde{J})$ and update the network using backpropagation. Although we have access to the reference data $R_{\hat{f},\hat{c}} J_{\hat{f}}$, we do not have $\tilde{J} = A\tilde{E}$ in (9), for which we would first need to solve the linear system in (11). Doing this for every training step would be prohibitively time consuming. Instead, we move to approximations of $\tilde{J}$.

Assume that we have a perfectly trained network that produces the linear operator A' such that $\|\delta_{f,A}\| = 0$. Then by definition

$$A'\tilde{E}' = R_{\hat{f},\hat{c}} J_{\hat{f}}, \tag{21}$$

where $\tilde{E}'$ solves the system containing A'. Furthermore,

$$\tilde{E}' = E^i_{\hat{c}} + G_{\hat{c}} A' \tilde{E}' = E^i_{\hat{c}} + G_{\hat{c}} R_{\hat{f},\hat{c}} J_{\hat{f}}, \tag{22}$$

where the right-hand side is known before training. If we now use $\tilde{E}'$ as the input to the network, we can use it to define a zeroth-order training error

$$\|\delta_0\| = \left\|R_{\hat{f},\hat{c}}J_{\hat{f}} - A\tilde{E}'\right\| = \left\|(A' - A)\tilde{E}'\right\|. \tag{23}$$

Minimizing $\|\delta_0\|$ would be sufficient to minimize (10) since it implies $A = A'$. However, in practice, perfect training is not possible and any remaining error in A would propagate throughout the iterative solver. Instead, we write

$$\delta_{\hat{f},A} - \delta_0 = A\left(E' - \tilde{E}\right) = AG\delta_{\hat{f},A}, \tag{24}$$

which leads to

$$\delta_{\hat{f},A} = (I - AG)^{-1}\delta_0. \tag{25}$$

Depending on A, this inverse may be approximated with a geometric series

$$(I - AG)^{-1} = \sum_{n=0}^{\infty}(AG)^n, \tag{26}$$

which we recognize as a Born series and which diverges for strong scatterers. If the series converges, we can approximate $\delta_{\hat{f},A}$ by truncating the sum to the first M terms

$$\delta_{\hat{f},A} \approx \delta_1 = \sum_{n=0}^{M-1}(AG)^n\delta_0. \tag{27}$$

In practice, we are restricted to a small value for M, since calling A multiple times in the neural network training loop requires a prohibitive amount of memory. By truncating the sum, we only need to execute A a total of $M+1$ times, including the initial invocation required to calculate δ_0.

When the Born series does not converge, this approach is not guaranteed to work. In [40], the boundary between convergence for 1D and 2D scalar EM scattering problems was quantified, and the use of Padé approximants to replace the Born series was proposed for strong scatters. In this work, we have tested both the truncated Born series and first- and second-order symmetric Padé approximants. For our use case, we noticed no significant difference in result. For this reason, we proceed with the truncated Born series for its ease of implementation.

Finally, we are only interested in the propagating part of the solution, i.e., solutions within the spectral domain $|\mathbf{k}| \leq k_0$, so we apply a window function before calculating the loss. We define

$$\hat{w}(\mathbf{k}) = \begin{cases} 1, & \text{if } \|\mathbf{k}\| \leq k_0 \\ \exp\left(-5(\|\mathbf{k}\| - k_0)^2\right), & \text{otherwise.} \end{cases} \tag{28}$$

We can now approximate the loss as

$$L\left(R_{\hat{f},A}J_{\hat{f}}, \tilde{J}\right) \approx L\left(\mathcal{D}_{\hat{c}}(\hat{w}(\mathbf{k}))\delta_1, 0\right). \tag{29}$$

Training the network to minimize this loss should yield more accurate numerical results when solving the final linear system than training on δ_0 would.

6. RESULTS

We apply the method described above to a 1D and a 2D scalar problem. We use the same general structure for both scenarios with the only major difference being the number and dimension of the convolution layers. BiCGstab [2] is used to solve linear systems with an absolute tolerance of 10^{-10}. In all tested cases, the number of iterations required for convergence was nearly identical between systems with modified and original contrast operators, having at most a difference of 1 iteration in favor of either one. For all simulations we take $\varepsilon_0 = \mu_0 = k_0 = \omega = 1$.

6.1. 1D Slab Scatterers

For the 1D case, we test the network on simple slab scatterers. These can be defined by three parameters, a, b and ε_r where a, b denote the beginning and end coordinates of the slab, respectively, and ε_r is the relative permittivity. We created the training data by uniformly sampling the simulation parameters according to Table 3. The discretization parameters are identical to those used in Table 1.

TABLE 3. Simulation parameters for the 1D simulations. Parameters are randomly sampled from a uniform distribution between Min and Max.

Parameter	Min	Max
a	-7	-3
b	2	5
ε_r	2	2

We choose to keep ε_r constant for a given training set. We do this because the final accuracy will be highly dominated by this value and using mixed permittivities will make it difficult to compare different samples. Both the A_E and the A_χ networks consist of convolutional layers. Their structure is presented in Table 4. The parameter N_c is the number of output channels for a given layer, N_k the kernel size, and N the number of modified contrast current terms as defined in (13), for the case we present here we choose $N = 4$. Note that we always end in a layer with kernel size 1 and a linear activation function. This is done so the output mappings are never constrained in their range by the activation functions (again, the imaginary channels of A_χ are an exception and instead use the swish activation function). The standard convolutional layers implemented

TABLE 4. Structure of the network. Only convolutional layers are used. Each layer uses zero-padding and has a stride of 1. The network layers are ordered from top to bottom. The final layer of the A_χ network uses a swish activation function for the imaginary channels only.

A_E	N_c	N_k	Activation
ConvComplex1D (1x)	64	50	linear
ConvComplex1D (1x)	N	1	linear
A_χ	N_c	N_k	Activation
Conv1D (15x)	32	3	tanh
ConvComplex1D (1x)	N	1	linear/swish
Number of parameters	45476		

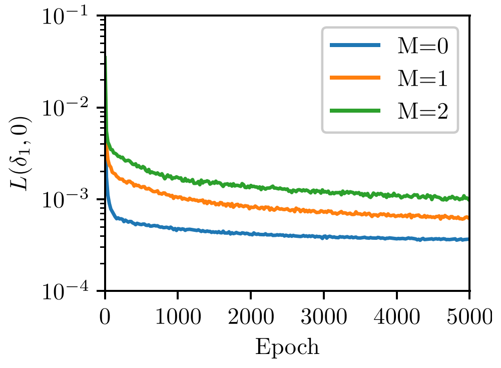


FIGURE 5. The δ_1 training loss for the 1D slab scatterer over 5000 epochs, for $M = 0, 1, 2$.

by TensorFlow are not linear in the complex numbers, since element-wise multiplication is used on the real and imaginary channels. For this reason, we implemented a `Conv1DComplex` layer that uses complex multiplication.

In Fig. 5, we show the training loss using the δ_1 approximation with $M = 1$ (27). Training was carried out in batches of 50 using an Adam optimizer [41] with a learning rate of 10^{-4}. It is clear that there are severely diminishing returns on training after a certain number of epochs.

After training, the network is tested in a validation step. We generate 50 additional scattering configurations from the same parameter space and solve (11) using the trained operator. As a reference, we also perform the system solve using the original coarse-system contrast operator. The resulting errors can be seen in Fig. 6. We observe a significant reduction in the discretization error (about a factor of 90). However, this increase in accuracy comes at the cost of computation time. The original, coarse, linear system without the modified contrast function required on average 0.042 s to solve, while the modified linear system took 0.22 s on average.

We performed the same simulation that we used to construct Fig. 2 to calculate the spectral errors for both contrast operators, see Fig. 7. We see that the trained operator counteracts the discretization error within the propagating region.

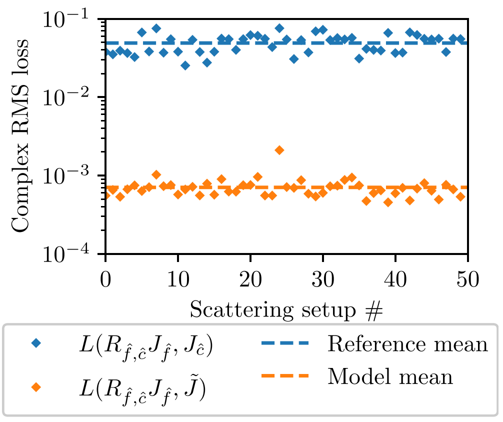


FIGURE 6. The validation losses for the 1D slab problem. Each point represents a simulation constructed from randomly sampled parameters and calculated on the coarse system for both the original (blue) and the modified (orange) contrast operators.

6.2. 2D Polygonal Scatterers

We performed the same training procedure for a 2D setup, using the same discretization parameters as in the 1D case, Table 1, but for both dimensions. For the scatterer geometry, we use convex polygonal scatterers, constructed as discussed in [39]. The polygonal scatterers are fully determined by a set of N_v vertices and the relative permittivity ε_r. To randomly generate a set of scatterers, we space the N_v vertices on a unit circle with equal angles between them. We then randomly change the radius r per vertex to create some variation and remove symmetries. For the incident field, we use a plane wave with incident angle ϕ_i and unit amplitude. The parameters we used are given in Table 5.

TABLE 5. Simulation parameters for the 2D simulations. Parameters are randomly sampled from a uniform distribution between Min and Max. The number of vertices N_v is always an integer. The radius r is sampled per vertex.

Parameter	Min	Max
x	-3	3
y	-3	3
ε_r	3	3
ϕ_i	0	2π
N_v	4	15
r	4	6

The network for the 2D case is identical in structure to the 1D case, with the exception of using 2D convolutions in the χ network. The network structure is given in Table 6, where for this 2D case we have used $N = 32$. We trained this network for over 2000 epochs using a second-order truncation in the training error term in (27), i.e. $M = 2$, since this produced better validation results than using $M = 1$ did. The resulting δ_1 training loss can be seen in Fig. 8.

TABLE 6. Structure of the 2D network. Only convolutional layers are used. Like in the 1D case, each layer uses zero-padding and has a stride of 1. We also again use the swish activation function for the imaginary output channels of χ.

A_E	N_c	N_k	Activation
ConvComplex1D (1x)	64	50×1	linear
ConvComplex1D (1x)	64	1×50	linear
ConvComplex2D (1x)	N	1	linear
A_χ	N_c	N_k	Activation
Conv2D (15x)	32	3×3	tanh
ConvComplex2D (1x)	N	1×1	linear/swish
Number of parameters	268928		

We again generated 50 new simulations for validation. We performed a full system solve on each of these and the resulting losses are shown in Fig. 9. Compared to the 1D case, we do not see the same improvement over the original coarse system. Instead, the discretization error only reduces by a factor of 5. The average computation time per simulation are 1.8 s and 4.9 s, for the original and the modified system, respectively. Unlike the 1D case, this difference in time cost is small enough to make

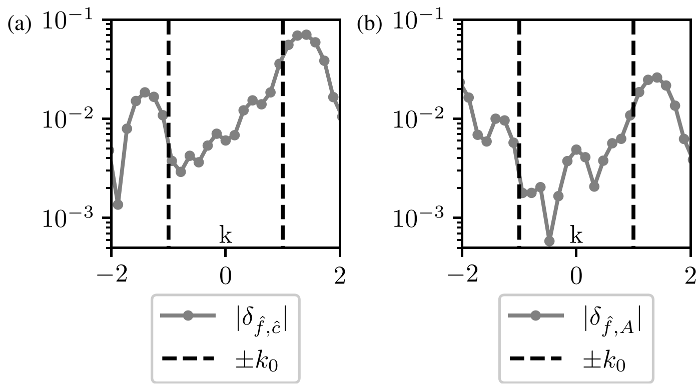


FIGURE 7. Spectral contrast current density errors for (a) the coarse and (b) the modified contrast operator.

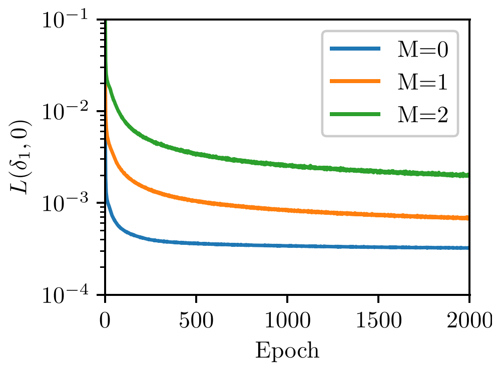


FIGURE 8. The δ_1 training loss for the 2D polyonal scatterers using $M = 0, 1, 2$.

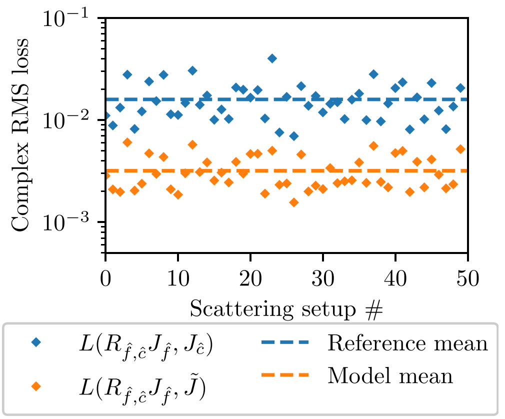


FIGURE 9. The validation losses for the 2D polygonal scattering problem. Each point on the horizontal axis represents a unique simulation that was solved using both the original and the modified contrast function.

this approach worthwhile since even a twofold increase in the discretization density of the original would surpass the modified system in computation time.

This network has 32 modified χ maps that all have some similarity. As an illustration, we show 4 of them in Fig. 10. From these samples, it can be seen that the mappings are trained into a structure with similar features to the Gibbs-phenomena around the scatterer boundaries, with different strengths in different directions. These can be interpreted as direction-dependent filters that, combined with the E field mappings, act to reduce the effects of the spectral truncation.

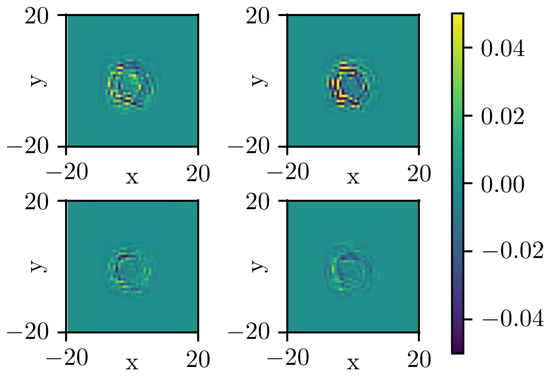


FIGURE 10. Examples of χ_n maps produced by the $A_\chi(\chi_c)$ operation for a random scattering 2D setup in the validation set. Visually, all maps look similar.

6.3. Extrapolation outside the Training Set

The modified contrast operator is trained on a constrained training set and might not function as expected for simulations outside these sets. To get an idea of the limits of the operator, we ran the validation step for several scenarios outside the parameter range of the training step. For this we used the network trained on the lossless convex polygonal scatterers. The cases we tested are:

1. Bigger radius and more vertices: $7 \leq r \leq 9$, $15 \leq N_v \leq 25$.
2. Two scatterers with $-7 \leq x \leq -5$ for the first and $5 \leq x \leq 7$ for the second. Both scatterers have $4 \leq r \leq 5$ to avoid overlap.
3. Cylindrical scatterer, centered at (x_c, y_c) and radius r, with $-3 \leq x_c \leq 3$, $-3 \leq y_c \leq 3$, $4 \leq r \leq 6$.
4. Higher contrast scatterer 1: $\varepsilon_r = 3.5$.
5. Higher contrast scatterer 2: $\varepsilon_r = 4$.

Parameters that are not mentioned are identical to those in the training set. We briefly list all the results in Table 7. From these

TABLE 7. Validation errors for several 2D scattering setups with parameters outside the training set. For every scenario 50 simulations with randomly sampled parameters were performed and their loss was averaged. All these scenarios consist of 50 simulations with the error values being averaged.

Case	$\lvert\delta_{\hat{f},\hat{c}}\rvert$	$\lvert\delta_{\hat{f},A}\rvert$
1	1.3×10^{-2}	4.8×10^{-3}
2	2.2×10^{-2}	1.2×10^{-2}
3	1.4×10^{-2}	5.1×10^{-3}
4	2.2×10^{-2}	8.4×10^{-3}
5	5.0×10^{-2}	3.1×10^{-2}

results, we see that the trained operator still results in improved accuracy for all scenarios. Although none of the results are as good as the results achieved on setups sampled from the training distributions, they are at least stable. This leaves us with the conclusion that, as long as we use similar scatters, we can use the trained operator with a reasonable expectation of proper behavior.

6.4. Use of Modified Contrast Operator on a Larger Computational Domain

The network was trained on a computational domain with 51 samples in each direction and a sample distance of $\Delta_x = 0.78$, see Table 3. Due to the simple CNN structure of the network we expect the trained operator to be spatially local. This means that we should be able to use the trained network on larger computational domains as well. To test this, we consider a new validation case. We use the same discretization, but this time with $N = 101$ samples in both x and y directions. We use the trained network trained in the previous subsection to solve the scattering problem on this new domain. We use the simulation parameters from Table 8 where compared to the previous case, we allow the scatterers a larger x and y translation.

TABLE 8. Simulation parameters for the validation set of the larger computational domain, using the network trained on convex polygons.

Parameter	Min	Max
x	-15	15
y	-15	15
ε_r	3	3
ϕ_i	0	2π
N_v	4	15
r	4	6

We show the results in Fig. 11. The results for these simulations are nearly as good as those for the original scattering problem in terms of achieved accuracy, demonstrating the generalizability of this operator to larger grid sizes.

6.5. Lossy Dielectrics

Thus far we have considered only dielectric scatterers without any losses. These scattering problems are generally more difficult than problems with lossy scatters. To verify that this also holds for the modified contrast-operator approach, we will compare networks trained on lossy and lossless materials. Therefore, we compare three different models

1. Model 1 is trained on lossless polygonal scatterers with $\varepsilon_r = 3$.
2. Model 2 is trained on lossy polygonal scatterers with a constant permittivity of $\varepsilon_r = 3 + 0.5i$.
3. Model 3 is trained on lossy polygonal scatters with variable losses where $\text{Re}(\varepsilon_r) = 3$ and $0 \leq \text{Im}(\varepsilon_r) \leq 1.5$.

Apart from the permittivity, all parameters are taken from Table 5. The models are trained with 1000 training samples, using the truncated Born series with $M = 2$ for 2000 epochs.

We compare the models on three corresponding validation sets. The first set contains lossless scatterers. The second set has scatterers with a constant complex permittivity of $\varepsilon_r = 3 + 0.5i$. The final set has a randomly sampled permittivity with $\text{Re}(\varepsilon_r) = 3$ and $0 \leq \text{Im}(\varepsilon_r) \leq 1.5$. We show the results in Fig. 12.

As expected, the models perform best on the validation sets closest to their training data. However, all three models perform better than the reference on all three validation sets. There are also no notable differences in macroscopic behavior between the lossy and lossless networks. From this we conclude that the addition of losses adds no additional challenges to the modified-contrast operator approach.

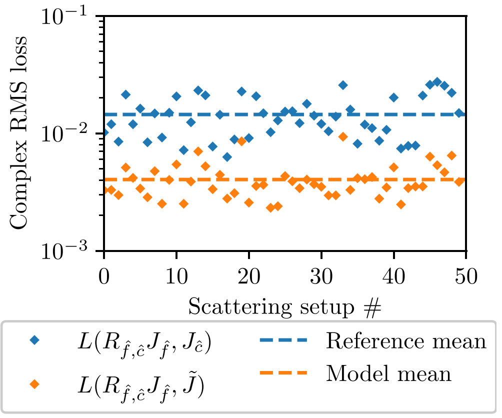
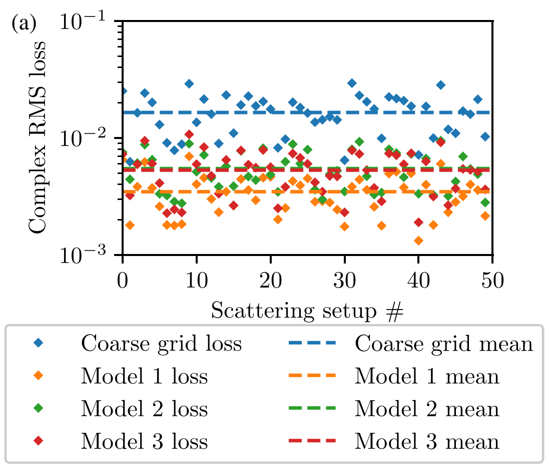


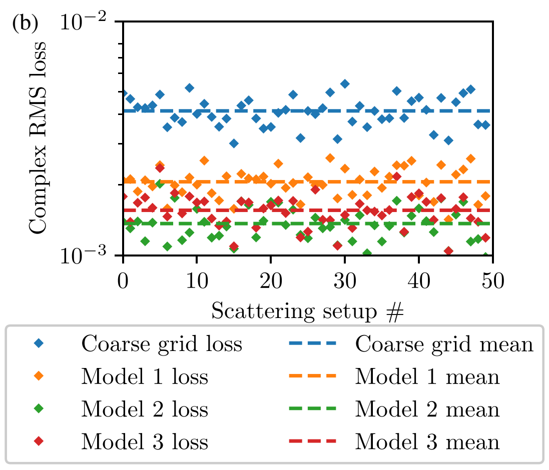
FIGURE 11. The validation losses for the original trained network, tested on a bigger computational domain. Each point on the horizontal axis represents a unique simulation that was solved using both the original and the modified contrast function.

6.6. 2D Hourglass Scatterers

In addition to the 2D convex polygons, we also train the network on concave hourglass-shaped scatterers. In Fig. 13, we show a depiction of such a scatterer. We trained the same network including these scatterers, this time using 500 polygonal scatterers from the same parameter space used previously and 500 (concave) scatterers from the hourglass distribution given in Table 9.

We compare the results for both the model trained on convex polygons and the model trained on this mixed set. We test both





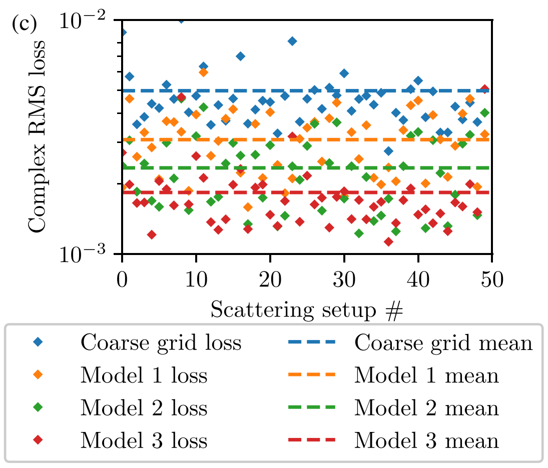


FIGURE 12. Validation losses for the three models for three different validation sets. (a) The first containing scatteres without losses, (b) the second containing scatteres with constant losses and (c) the third containing scatterers with variable losses.

models on both a convex polygonal validation set and an hourglass validation set, both with 50 samples. We show the results in Fig. 14. We see that both networks perform well for both validation sets. However, the network trained on the hourglass scatterers performs significantly better on the hourglass validation set, while having a nearly identical performance on the convex polygon validation set.

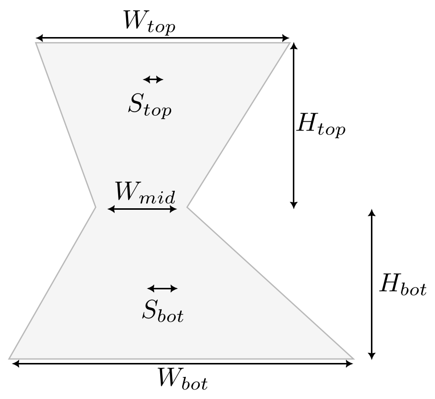


FIGURE 13. Schematic depiction of an hourglass scatterer. The W and H parameters refer to widths and heights, while S is the slant, or the offset, as compared to the center.

TABLE 9. Simulation parameters for the 2D hourglass simulations. Parameters are randomly sampled from a uniform distribution between Min and Max. The W parameters concern the different widths, the H parameters the height, and the S parameters the slant of the top and bottom.

Parameter	Min	Max
x	-3	3
y	-3	3
ε_r	3	3
ϕ_i	0	2π
W_{mid}	3	4
W_{top}	4	5
W_{bot}	5	7
S_{top}	-1	1
S_{bot}	-1	1
H_{top}	4	5
H_{bot}	4	5

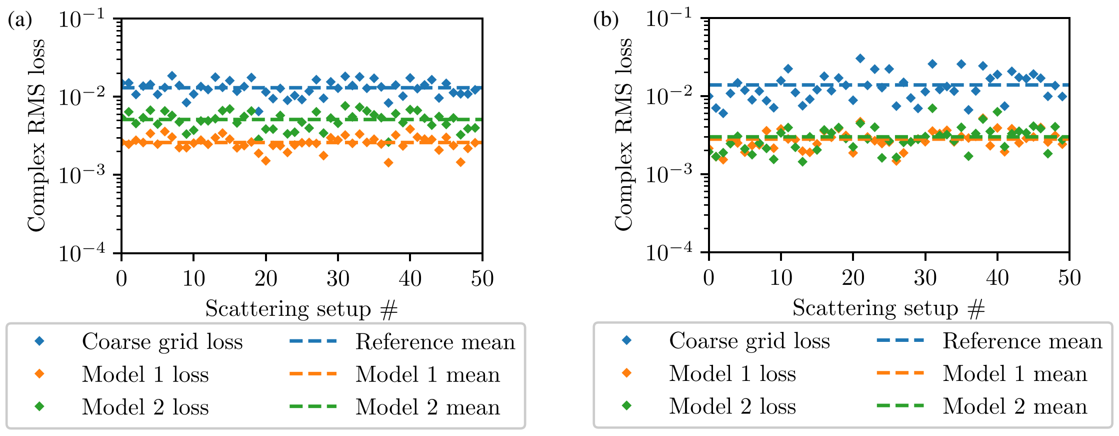


FIGURE 14. Results for both the network trained on hourglass scatterers (Model 1) and convex polygonal scatterers (Model 2) for (a) 50 hourglass scattering problems and (b) 50 convex polygon scattering problems.

7. CONCLUSION

We introduced a trainable operator that acts as a modified dielectric contrast operator on a coarse grid discretization of a domain integral equation. This operator reduces the discretization error for small wavenumbers, which means that the far-field accuracy is increased. It does this while still incurring a lower computational cost than the unmodified operator on a fine discretization grid. The operator is constructed as a combination of two convolutional neural networks, one acting linearly on the electric field while the other acts non-linearly on the contrast function. The network has the same general structure for the one-dimensional and two-dimensional case, thus providing a uniform interface regardless of the problem's dimension. Solving the altered system results in a significant reduction in the discretization error compared to the standard operator on a coarse grid. The operator is trained using standard backpropagation provided by TensorFlow.

The approach is flexible, being nearly identical in one and two dimensions without change in formulation and independent of the specifics of the scatterer's geometry. This provides a general, hands-off procedure to reduce the number of unknowns in forward-scattering problems.

REFERENCES

[1] Gibson, W. C., *The Method of Moments in Electromagnetics*, CRC Press, 2021.

[2] Van der Vorst, H. A., "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 13, No. 2, 631–644, 1992.

[3] Saad, Y. and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 856–869, 1986.

[4] Zwamborn, P. and P. M. van den Berg, "The three dimensional weak form of the conjugate gradient FFT method for solving scattering problems," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 40, No. 9, 1757–1766, 1992.

[5] Dilz, R. J. and M. C. van Beurden, "An efficient spatial spectral integral-equation method for EM scattering from finite objects in layered media," in *2016 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, 509–511, Cairns, Australia, Sep. 2016.

[6] Dilz, R. J. and M. C. van Beurden, "A domain integral equation approach for simulating two dimensional transverse electric scattering in a layered medium with a Gabor frame discretization," *Journal of Computational Physics*, Vol. 345, 528–542, 2017.

[7] Dilz, R. J., M. G. M. M. van Kraaij, and M. C. van Beurden, "A 3D spatial spectral integral equation method for electromagnetic scattering from finite objects in a layered medium," *Optical and Quantum Electronics*, Vol. 50, No. 5, 206, 2018.

[8] Eijsvogel, S., R. J. Dilz, R. Bojanić, and M. C. van Beurden, "Phaseless inverse scattering with a parametrized spatial spectral volume integral equation for finite scatterers in the soft X-ray regime," *Journal of the Optical Society of America A*, Vol. 41, No. 11, 2076–2089, 2024.

[9] Ergul, O. and L. Gurel, *The Multilevel Fast Multipole Algorithm (MLFMA) for Solving Large-Scale Computational Electromagnetics Problems*, John Wiley & Sons, 2014.

[10] Hu, B., W. C. Chew, E. Michielssen, and J. Zhao, "Fast inhomogeneous plane wave algorithm for the fast analysis of two-dimensional scattering problems," *Radio Science*, Vol. 34, No. 4, 759–772, 1999.

[11] Chen, Y., J. Hu, Z. Nie, and L. Lei, "A FAFFA-FIPWA algorithm for two-dimensional electromagnetic scattering problem," in *2005 IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, Vol. 1, 479–482, Beijing, China, 2005.

[12] Weiss, T., G. Granet, N. A. Gippius, S. G. Tikhodeev, and H. Giessen, "Matched coordinates and adaptive spatial resolution in the Fourier modal method," *Optics Express*, Vol. 17, No. 10, 8051–8061, 2009.

[13] Krizhevsky, A., I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, Vol. 25, 2012.

[14] He, K., X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[15] Ronneberger, O., P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *arXiv preprint arXiv:1505.04597*, 2015.

[16] Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, Vol. 29, No. 6, 82–97, 2012.

[17] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[18] Raissi, M., P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[19] Guo, L., M. Li, S. Xu, and F. Yang, "Study on a recurrent convolutional neural network based FDTD method," in *2019 International Applied Computational Electromagnetics Society Symposium — China (ACES)*, Vol. 1, 1–2, Nanjing, China, 2019.

[20] Yao, H. M. and L. J. Jiang, "Machine learning based neural network solving methods for the FDTD method," in *2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, 2321–2322, Boston, MA, USA, 2018.

[21] Yee, K., "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, Vol. 14, No. 3, 302–307, 1966.

[22] LeCun, Y., Y. Bengio, and G. Hinton, "Deep learning," *Nature*, Vol. 521, No. 7553, 436–444, 2015.

[23] Rumelhart, D. E. and J. L. McClelland, *Learning Internal Representations by Error Propagation*, 318–362, MIT Press, 1987.

[24] Sak, H., A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proceedings of the Annual Conference of the International Speech Communication Association*, 338–342, Singapore, 2014.

[25] Yao, H. M. and L. Jiang, "Machine-learning-based PML for the FDTD method," *IEEE Antennas and Wireless Propagation Letters*, Vol. 18, No. 1, 192–196, 2018.

[26] Key, C. and B. M. Notaroš, "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*, Vol. 19, No. 4, 626–630, 2020.

[27] Jin, J.-M., *The Finite Element Method in Electromagnetics*, John Wiley & Sons, 2002.

[28] Key, C. and B. M. Notaroš, "Predicting macro basis functions for method of moments scattering problems using deep neural networks," *IEEE Antennas and Wireless Propagation Letters*, Vol. 20, No. 7, 1200–1204, 2021.

[29] Guo, R., T. Shan, X. Song, M. Li, F. Yang, S. Xu, and A. Abubakar, "Physics embedded deep neural network for solving volume integral equation: 2-D case," *IEEE Transactions on Antennas and Propagation*, Vol. 70, No. 8, 6135–6147, 2022.

[30] Guo, R., Z. Lin, T. Shan, X. Song, M. Li, F. Yang, S. Xu, and A. Abubakar, "Physics embedded deep neural network for solving full-wave inverse scattering problems," *IEEE Transactions on Antennas and Propagation*, Vol. 70, No. 8, 6148–6159, 2022.

[31] Xiao, L.-Y., J.-N. Yi, Y. Mao, X.-Y. Qi, R. Hong, and Q. H. Liu, "A novel optical proximity correction (OPC) system based on deep learning method for the extreme ultraviolet (EUV) lithography," *Progress In Electromagnetics Research*, Vol. 176, 95–108, 2023.

[32] Xu, F. and S. Fu, "Modeling EM problem with deep neural networks," in *2018 IEEE International Conference on Computational Electromagnetics (ICCEM)*, 1–2, Chengdu, China, 2018.

[33] Van den Hof, D., M. C. van Beurden, and R. Dilz, "Flexible discretization of singular Green functions using a composite spectral integration path," *Progress In Electromagnetics Research B*, Vol. 107, 77–90, 2024.

[34] Abadi, M., A. Agarwal, P. Barham, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," *arXiv:1603.04467*, 2016.

[35] Dilz, R., "The windowed fourier series with a complex-plane path deformation as an efficient discretization for spatial spectral integral equations," in *AES 2024 Rome-Italy: The 10th International Conference on Antennas and Electromagnetic Systems*, 324–325, Rome, Italy, Jun. 2024.

[36] Ramachandran, P., B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[37] Harris, C. R., K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, *et al.*, "Array programming with NumPy," *Nature*, Vol. 585, No. 7825, 357–362, Sep. 2020.

[38] Hunter, J. D., "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, Vol. 9, No. 3, 90–95, 2007.

[39] Eijsvogel, S., L. Sun, F. Sepehripour, R. J. Dilz, and M. C. van Beurden, "Describing discontinuous finite 3D scattering objects in Gabor coefficients: Fast and accurate methods," *Journal of the Optical Society of America A*, Vol. 39, No. 1, 86–97, 2022.

[40] Van Der Sijs, T. A., O. El Gawhary, and H. P. Urbach, "Electromagnetic scattering beyond the weak regime: Solving the problem of divergent Born perturbation series by Padé approximants," *Physical Review Research*, Vol. 2, No. 1, 013308, 2020.

[41] Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, Vol. 1412, No. 6, 2014.