# Machine Learning Exercises on 1-D Electromagnetic Inversion

Ergun Simsek, *Senior Member, IEEE*

***Abstract*— This work aims to enhance our fundamental understanding of how the measurement setup that is used to generate training and testing data sets affects the accuracy of the machine learning algorithms that attempt to solve electromagnetic inversion problems solely from data. A systematic study is carried out on a 1-D semi-inverse electromagnetic problem, which is estimating the electrical permittivity values of a planarly layered medium with fixed layer thicknesses assuming different receiver–transmitter antenna combinations in terms of location and numbers. The accuracy of the solutions obtained with four machine learning methods, including neural networks, is compared with a physics-based solver deploying the Nelder–Mead simplex method to achieve the inversion iteratively. Numerical results show that: 1) deep-learning outperforms the other machine learning techniques implemented in this study; 2) increasing the number of antennas and placing them as close as possible to the domain of interest increase inversion accuracy; 3) for neural networks, training data sets created on random grids lead to more efficient learning than the training data sets created on uniform grids; and 4) multifrequency training and testing with a few antennas can achieve more accurate inversion than single-frequency setups deploying several antennas.**

***Index Terms*— Deep learning, electromagnetic inversion, machine learning, optimization.**

## I. Introduction

In THE last decade, machine learning has received enormous interest from scientists and engineers from almost all disciplines. In addition to machine learning's potential in solving complex problems, free of charge programming languages and platforms for cloud computing and data storage, an open-source culture of the machine learning community, and affordability of graphics processing units (GPUs) are a few of the many possible factors driving this growing interest.

Machine learning has become a popular subject in the computational electromagnetics (CEM) society as well. Researchers have proposed using machine learning to solve advanced CEM problems in device design [1]–[3], material characterization [4], geophysical prospecting [5], [6], and electromagnetic inversion [3], [5], [7]–[16], which attempts to estimate the distribution of physical properties in a domain of interest from antenna measurements collected outside of that domain. Since the inversion problems are nonlinear, nonunique, and ill-posed [17], [18], electromagnetic inversion has been one of the most challenging subjects studied by the CEM society over the past decades. According to the recent studies [3], [5], [8]–[14], deep learning, a machine learning system deploying neural networks (NNs), has the potential to make substantial improvements in this area.

Even though there are some studies on 3-D electromagnetic inversion problems, the majority of the recent studies focus on 2-D electromagnetic inversion due to its relative simplicity. In most of these studies, the domain of interest is a square, and around this square are multiple antennas placed uniformly over a circle. The radius of this circle is chosen carefully so that no antenna is present very close to the domain of interest. The number of antennas is determined according to the Nyquist sampling theorem, i.e., the spatial distance between two neighboring receiver antennas is about a half wavelength or less. Another common element among these studies is that the main focus is the design of the neural network. They assume the measurement setup is ideal, probably based on the past experiences gained during traditional electromagnetic inversion solver development and the majority of them use deep learning [5], [8]–[14]. However, electromagnetic inversion can also be implemented using other machine learning techniques, such as linear regression (LR), k-nearest neighbor (kNN), and random forest. Thus, when we compare the current best practices in machine learning and electromagnetic inversion methods, we come up with some very fundamental questions. For example, can machine learning techniques other than deep learning achieve similar accuracy in electromagnetic inversion or is deep-learning significantly better than the other methods and do we have to use it for the most accurate electromagnetic inversion implementation? We know that the waveforms are extremely complex in the near-field, but the machines might learn from this complexity, so can bringing antennas closer to the search domain improve the accuracy or should we put them at least half-wavelength far from the domain of interest? How does the accuracy change when we change the number of antennas? Does multifrequency training bring any advantage compared to single-frequency training?

To find some clues that might help with answering these questions, here, we study a simplified 1-D electromagnetic inversion problem: estimating the electrical permittivity profile of a multilayered, lossless medium with fixed layer thicknesses. First, we provide an example case where the

Manuscript received July 9, 2020; revised December 28, 2020; accepted February 14, 2021. Date of publication April 7, 2021; date of current version October 6, 2021.

The author is with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250 USA (e-mail: simsek@umbc.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAP.2021.3069519.

Digital Object Identifier 10.1109/TAP.2021.3069519

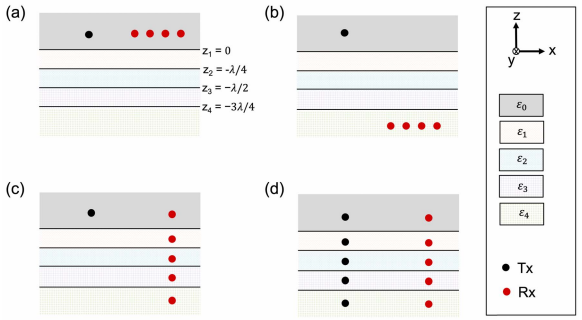


Fig. 1. Five-layer medium where the three inner layers are $\lambda/4$ thick. Top layer is air with a relative electrical permittivity of 1. Four different receiver ($R_x$)–transmitter ($T_x$) antenna combinations are considered: in the first three configurations, there is only one $T_x$ and it is in the top layer; receivers are in (a) top layer, (b) bottom layer, (c) different layers and distributed vertically; and (d) one $R_x$ and one $T_x$ in each layer.

inversion is achieved iteratively using a forward solver, and we call it a "physics-based" electromagnetic inversion method. Then, we use three main machine learning techniques (LRs, kNN, and random forest) and a simple NN to estimate the permittivity profile solely from data for various scenarios. We compare the accuracy and efficiency of the machine learning solutions and discuss how the given solutions might guide us in designing measurement setups enabling more accurate electromagnetic inversion. Then, we work on a slightly more complicated problem, which can be considered as a 1-D pixel-based electromagnetic inversion. Finally, we provide some conclusions.

## II. Forward Problem

Assume a planarly layered medium with five layers where all the interfaces are parallel to the $xy$-plane, as shown in Fig. 1(a). All the layers are homogeneous, isotropic, nonmagnetic, and lossless. The top and bottom layers are infinitely long along the $z$-axis. The top layer is air, so its relative electrical permittivity is equal to one, $\epsilon_0 = 1$. The first interface, the interface between the top and underlying layer, is at $z_1 = 0$. The thicknesses of the inner layers are all equal to $\lambda/4$, where $\lambda$ is the wavelength of the electromagnetic waves created by the transmitter antenna ($T_x$). These waves propagate along with the planarly layered medium, and we measure the electric field intensity at the receiver antennas ($R_x$s). Both transmitter and receiver antennas are dipole antennas located on the $xz$-plane (i.e., $y = 0$), pointing along the either $x$-, $y$-, or $z$-axis.

For a multilayered media with all the layers' thicknesses and permittivity values known, layered medium Green's functions (LMGFs) give us the $\eta$-component of the electric field recorded at a receiver located at $\mathbf{r}$ due to a $\zeta$-directed electrical dipole located at $\mathbf{r}'$ in a tensor form [18], [19], where $\eta$ and $\zeta$ are either $x$, $y$, or $z$, that is

$$\mathrm{G}(\mathbf{r}, \mathbf{r}', \boldsymbol{\epsilon}, \mathbf{z}, \lambda) = \begin{bmatrix} G_{xx} & G_{xy} & G_{xz} \\ G_{yx} & G_{yy} & G_{yz} \\ G_{zx} & G_{zy} & G_{zz} \end{bmatrix} \tag{1}$$

where $\boldsymbol{\epsilon} = \{\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\}$ and $\mathbf{z} = \{z_1, z_2, z_3, z_4\} = \{0, -\lambda/4, -\lambda/2, -3\lambda/4\}$, as shown in Fig. 1.

One can calculate these dyadic Green's functions by transforming the problem from the spatial domain to the spectral domain in which each layer is treated as a transmission line. The layer thicknesses become the lengths of these transmission lines whose impedances are calculated with those layers' electrical permittivity and magnetic permeability values. The voltage and current carried over these transmission lines in series can be calculated by solving the transmission-line equations recursively, corresponding to the electric and magnetic parts of electromagnetic waves propagating in that multilayered structure, respectively. Later, the electric and magnetic field intensities can be computed by transforming the problem back from the spectral domain to spatial domain by numerically evaluating the Sommerfeld integrals [18], [19].

## III. Physics-Based Electromagnetic Inversion

In an inversion problem, we are given some antenna measurements and asked to determine electric permittivity, conductivity, and/or magnetic permeability distributions over a domain of interest. In some inversion applications, pixels and cells are used to discretize the domain of interest in 2-D and 3-D implementations, respectively. There, the main goal is determining the physical parameters of each pixel or cell. Here, we first study a simpler problem: determining the electric permittivity of four layers with fixed thicknesses. This can be accomplished numerically by using an LMGF calculator as a forward solver. We first choose some random permittivity values ($\boldsymbol{\epsilon}_p$) and calculate LMGFs. The Euclidean distance between given (measured) and calculated field intensities normalized with the magnitude of the largest measurement value can be defined as a cost function, that is

$$f(\boldsymbol{\epsilon}_p) = \left|\left|\mathrm{G}_{given} - \mathrm{G}(\mathbf{r}, \mathbf{r}', \boldsymbol{\epsilon}_p, \mathbf{z}, \lambda)\right|\right|_2 / \max\{\mathrm{G}_{given}\}. \tag{2}$$

Then, we minimize this cost iteratively with an optimization algorithm without any additional regularization. When the error becomes smaller than the desired threshold value, the permittivity prediction is completed.

For the implementation, we choose the Nelder–Mead simplex algorithm [20] for the cost minimization and set our threshold to $10^{-2}$. Each iteration is allowed to have 1000 subiterations to achieve a reduction in the cost. In other words, 1000 different sets of LMGFs are calculated for each iteration. Fig. 2 shows how the cost decays as a function of iteration number for a sample case where the permittivity values are randomly selected, assuming a transmitter antenna at $(x' = 0, z' = \lambda/2)$ and 20 receiver antennas equally spaced between $x = \lambda$ and $x = 2\lambda$ at $z = \lambda/2$. The "measurement" data are created by changing the direction of transmitter and receiver antennas to $\{x, z, y, x, z\}$ and $\{x, x, y, z, z\}$, respectively. The reason behind using this set of polarizations is given as follows: since all the layers are isotropic and homogeneous, we know that $G_{xz} = G_{yz}$ and $G_{zx} = G_{zy}$; and since all the antennas are assumed to be on the $xz$-plane, $G_{xy} = G_{yx} = 0$. This means that, in our configuration, there are only five distinct components: $G_{xx}$, $G_{xz}$, $G_{yy}$, $G_{zx}$, and $G_{zz}$. From these five LMGF components (electric field intensity measurements), we determine the relative electrical permittivity of the four layers in nine iterations. The true permittivity values and the ones predicted by this physics-based iterative solver are listed in Table I.

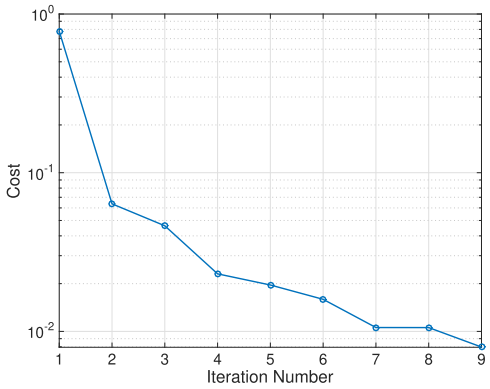


Fig. 2. Cost versus number of iterations for the physics-based inversion algorithm.

TABLE I
PERMITTIVITY VALUES PREDICTED BY THE PHYSICS-BASED SOLVER VERSUS TRUE PERMITTIVITY VALUES

| | $\epsilon_1$ | $\epsilon_2$ | $\epsilon_3$ | $\epsilon_4$ |
|---|---|---|---|---|
| True Values | 4.4164 | 7.7038 | 6.6807 | 8.3325 |
| Predicted Values | 4.4143 | 7.7057 | 6.743 | 8.4009 |

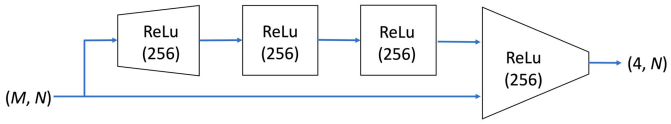


Fig. 3. Neural network implemented in this work has four layers all implemented with 256 neurons and ReLu activation functions. $N = n_s^4$ during training, and $N = 1000$ during testing. $M = 10 \times n_r \times n_t$, where $n_r$ and $n_t$ are the number of receiver and transmitter antennas.

In addition to this test, all the scenarios discussed below in Section V are also solved with this numerical solver. In all cases, absolute errors are observed to be less than 1%, meaning that the physics-based solvers can achieve inversion with very high accuracy for these kinds of simplified semi-inversion problems. Next, we use different machine learning techniques aiming to achieve inversion solely from data.

## IV. ELECTROMAGNETIC INVERSION WITH MACHINE LEARNING

We first implement a simple NN on Google Colaboratory using Keras' [22] functional application program interface (API) running on top of TensorFlow [23]. Fig. 3 shows the NN used in this work, where the first three layers transform the input data with the ReLu activation function [24] to achieve the learning. The last layer, again implemented with the ReLu activation function, accepts the output of the third layer merged with the original input. Adam is used as the optimizer [25] to minimize the mean squared error.

To answer the first question asked in the "Introduction" (is deep learning better than other machine learning methods?), we use three of the most frequently used machine learning algorithms: LR, kNN, and random forest due to their simple implementation and intuitive outcomes. The details of these algorithms are beyond the scope of this article, so the readers can kindly refer to [21]. We implement these methods with a Python module named "sklearn" along with "numpy, pandas, and matplotlib."

To create a training data set, we assume $N$ different sets of $\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = \{1, \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\}$ and $1 \leq \epsilon_i \leq 10$ for $i = 1, 2, 3, 4$. We calculate five LMGF components ($G_{xx}$, $G_{xz}$, $G_{yy}$, $G_{zx}$, and $G_{zz}$) representing different transmitter–receiver antenna polarization combinations for each geometry. Since LMGFs are complex numbers, their real and imaginary parts are stored separately. In the end, we obtain $10 \times n_r \times n_t$ distinct numbers for each $\boldsymbol{\epsilon}$, where $n_r$ and $n_t$ are the number of receiver and transmitter antennas, respectively. For validation and testing, we create two other data sets of LMGFs for randomly selected 1000 arrays of permittivities.

Since the layers closest to the antennas are likely to have a more direct impact on the received electromagnetic waves compared to the further layers, the prediction accuracy is expected to be a function of layer index. To verify this expectation, we carry out the following sensitivity analysis. We first choose four numbers randomly between 1 and 10 to be used as the permittivity values for layers 1 to 4, and we calculate the LMGFs assuming that the transmitter antenna is at ($x' = 0$, $z' = 0.5\lambda$) and 20 receiver antennas uniformly placed between $x = 0.5\lambda$ and $x = 1.5\lambda$ at $z = 0.5\lambda$. Then, we replace the permittivity of the first layer with another randomly selected number between 1 and 10; then, we calculate the LMGFs again. For the generation of random numbers, we use MATLAB's default random number generator algorithm. In the end, we record the difference between permittivity values of the first layer and the Euclidean distance between the LMGF sets normalized by the number of receiver antennas. We repeat this procedure 1000 times. The exact algorithm that we use is given as follows.

$[\epsilon_A, \epsilon_B, \epsilon_C, \epsilon_D, \epsilon_E] = 1 + 9 * \text{rand}(1000, 5)$
**for** $m = 1 : 1000$ **do**
  $\boldsymbol{\epsilon} = [1, \epsilon_A(m), \epsilon_C(m), \epsilon_D(m), \epsilon_E(m)]$
  $\text{G}_{1A} \leftarrow$ Calculate LMGFs
  $\boldsymbol{\epsilon} = [1, \epsilon_B(m), \epsilon_C(m), \epsilon_D(m), \epsilon_E(m)]$
  $\text{G}_{1B} \leftarrow$ Calculate LMGFs
  $\Delta\epsilon_1(m) = |\epsilon_A(m) - \epsilon_B(m)|$
  $||\text{G}_1||_2^{ave}(m) = ||\text{G}_{1A} - \text{G}_{1B}||_2/n_r$
**end for**

Then, we follow the same recipe for the other three layers. In Fig. 4, we plot how LMGFs change by changing the electrical permittivity of one layer while keeping the other three the same. The $y$-axes shrinking from (a) to (d) clearly indicate that the first layer has the highest impact on LMGFs, and the impact of other layers decreases with increasing layer number, confirming LMGFs' sensitivity to the layer index. This is why we calculate the normalized root mean square error ($\Xi_i$) for each layer separately using the following formula:

$$\Xi_i = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\frac{\hat{\epsilon}_{i,j} - \epsilon_{i,j}}{\epsilon_{i,j}}\right)^2} \tag{3}$$

to compare machine learning algorithms' prediction accuracy for each layer, where $\epsilon_{i,j}$ and $\hat{\epsilon}_{i,j}$ are the true and predicted permittivity values for layer-$i$ in the $j$th test. The overall

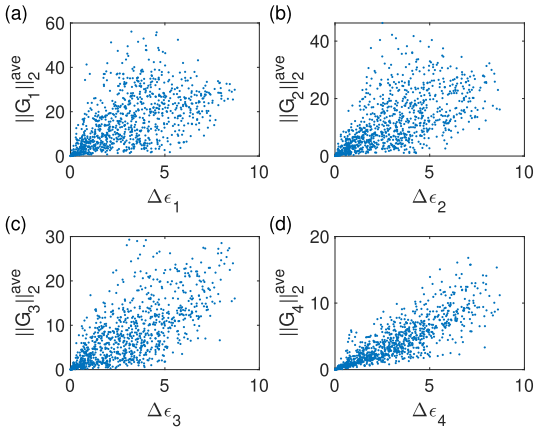


Fig. 4. Sensitivity analysis: how much LMGFs change with changing (a) $\epsilon_1$, (b) $\epsilon_2$, (c) $\epsilon_3$, and (d) $\epsilon_4$.

error ($\Xi_o$) can be measured with the square root of the average of squared errors, that is

$$\Xi_o = \frac{1}{2}\sqrt{\sum_{i=1}^{4} \Xi_i^2}. \tag{4}$$

In the NN implementations, the number of epochs is set to 1000 for all the examples presented in this work. For the training, we first take $n_s$ values linearly sampled between 1 and 10 and then assign these values as permittivity of each layer in an ordered arrangement fashion yielding $N = n_s^4$ different permutations (with repetitions). Fig. 5(a) plots how validation (during training) and test losses decay with the number of epochs for the scenario discussed in Section III where all the antennas are in the top layer and $N = n_s^4 = 18^4 = 104\,976$. These smoothly decaying and converging losses confirm that our network, indeed, learns from data. Fig. 5(b)–(e) shows predicted versus actual values for the 1000 cases used in the testing. If the accuracy was 100%, then we would have all the points on the $x = y$ line for $1 \leq x, y \leq 10$. However, what we observe is that the number of points not on the $x = y$ line and their distances to the $x = y$ line both increase from (b) to (e), which means that the prediction accuracy decreases with increasing layer index when all the antennas are in the top layer. This result also means that the prediction accuracy, indeed, is a function of layer sensitivity. When we compare the results depicted in Figs. 4 and 5, we can clearly see that the permittivity changes of the layers closer (further) to the antennas cause a larger (smaller) change in the LMGFs that lead to more (less) efficient learning, which is discussed in detail in Section V.

## V. Numerical Results and Discussion

### A. Machine Learning Versus Deep Learning

For the scenario described in § III, we use the three machine learning methods and the NN described in § IV to answer our first question: can machine learning techniques other than deep learning achieve similar or higher accuracy in an electromagnetic inversion? Fig. 6 shows the normalized root mean square error for each layer separately for the estimates obtained with four machine learning implementations trained

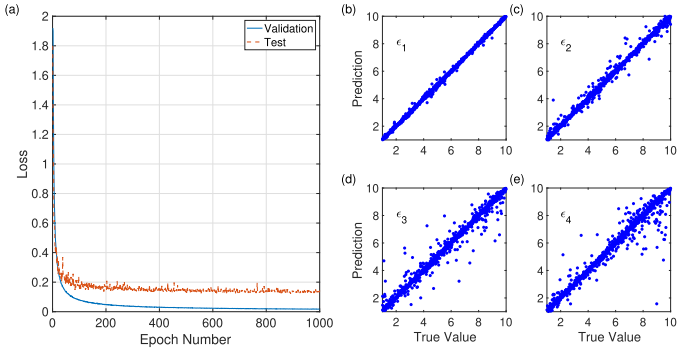


Fig. 5. (a) Validation (blue curve) and test (red dashed curve) losses versus epoch number for the NN trained with $n_s = 18$ data set. (b)–(e) True permittivity versus predicted permittivity values for the 1000 cases used in the testing.

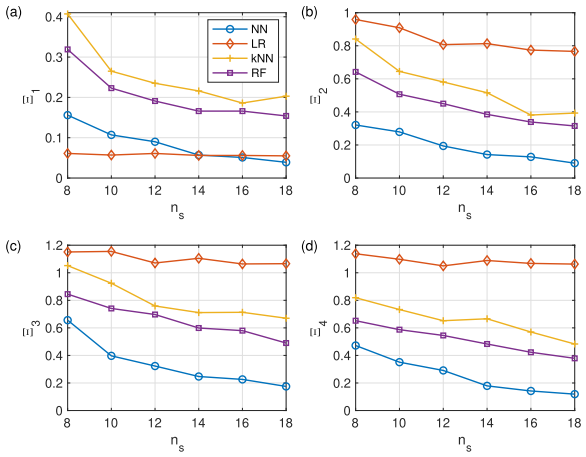


Fig. 6. (a)–(d) Normalized root mean square error ($\Xi_i$ for $i = 1, 2, 3$, and 4, respectively) versus sampling density ($n_s$) for NN, LR, kNN, and random forest implementations for the scenario shown in Fig. 1(a).

with the data sets with changing size as a function of $n_s$. Note that, when $n_s$ is increased from 8 to 18, the number of samples in our training data set increases from 4096 to 104 976. In Fig. 6(a), the red curve marked with diamonds shows that, for smaller training data sets, LR is the most accurate method to predict the permittivity of the first layer. In fact, the accuracy of the LR-based solver is almost independent of sampling density, $n_s$, for the first layer, and it is only beaten by the NN-based solver when $n_s \geq 16$. However, LR performs poorly for the lower layers. The performances of kNN and random forest are close to each other, but they are not as accurate as of the NN implementation, which we can infer from the blue curve marked with circles in Fig. 6. For the lower three layers, the NN is the most accurate one by far.

When we look at the overall accuracy ($\Xi_o$), NNs, indeed, outperform the traditional machine learning techniques in achieving electromagnetic inversion solely from data. As previously reported in [26]–[28], the use of nonlinear activation functions is the main reason behind neural networks' success in solving nonlinear problems. However, this higher accuracy usually comes with a price: longer computation time and higher memory usage. To have a comparative understanding, we plot the time spent and memory used for the training

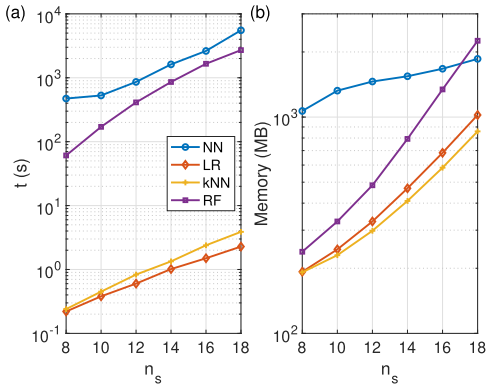


Fig. 7. (a) GPU time and (b) memory versus sampling density ($n_s$) for NN, LR, kNN, and random forest implementations.

or data fitting in the machine learning implementations as a function of sampling density in Fig. 7(a) and (b), respectively. With their relatively simple mathematical formulations, the LR and kNN require two orders of magnitude or even less time than random forest and NN implementations. Of course, we can speed up the neural network and random forest by reducing the number of epochs and estimators, respectively, or by using multiprocessors and parallelization. However, the important messages that we infer here are: 1) random forests and NNs can require substantially longer times than the LR and kNN models during the training/data fitting and 2) there is not a major time difference between NNs' training and random forests' data fitting, especially for large training data sets when we use a single GPU. In terms of memory usage, NNs have a significantly different trend than the others, i.e., they use much larger memory even for smaller training data sets because of the high number of neurons implemented in the NN. Even then, random forest's memory usage surpasses the NN's memory for $n_s \geq 18$.

At the end of this brief study, we can claim that, despite their computational cost, the use of NNs is indeed a promising approach due to their high accuracy for solving electromagnetic inversion problems solely from data. Next, we investigate how the inversion accuracy depends on the interantenna spacing.

### *B. Antenna Spacing Versus Inversion Accuracy*

As previously mentioned, in many studies, the interantenna spacing is set close to $\lambda/2$ [8]–[11], [13], [14]. According to the Nyquist sampling theorem, this distance should be sufficient for a successful inversion, but it would be useful to understand how the interantenna spacing affects the inversion accuracy. In this direction, we consider two different configurations as follows.

In the first scenario, all the receiver and transmitter antennas are located in the top layer, as shown in Fig. 1(a). We place 33 receiver antennas uniformly along $0.5\lambda \leq x \leq 2\lambda$, $z = 0.5\lambda$. The transmitter antenna is located at $(x' = 0, z' = 0.5\lambda)$. The training set has 20 736 examples, corresponding to $n_s = 12$. We first use the entire training data set and obtain an overall inversion error of 0.216 with our NN implementation, as listed in the last row of Table II. Then, we use the field

TABLE II

ERROR VERSUS INTERANTENNA SPACING: USING DIFFERENT NUMBERS OF ANTENNAS LOCATED UNIFORMLY BETWEEN $x = 0.5\lambda$ AND $x = 2\lambda$

| No. of Ant. | Spacing ($\lambda$) | $\Xi_1$ | $\Xi_2$ | $\Xi_3$ | $\Xi_4$ | $\Xi_o$ |
|---|---|---|---|---|---|---|
| 4 | 0.500 | 0.107 | 0.282 | 0.412 | 0.286 | 0.293 |
| 5 | 0.375 | 0.065 | 0.205 | 0.358 | 0.278 | 0.251 |
| 9 | 0.188 | 0.057 | 0.213 | 0.345 | 0.261 | 0.243 |
| 17 | 0.094 | 0.088 | 0.193 | 0.315 | 0.281 | 0.236 |
| 33 | 0.047 | 0.051 | 0.182 | 0.302 | 0.242 | 0.216 |

TABLE III

ERROR VERSUS INTERANTENNA SPACING: 20 ANTENNAS WITH A DIFFERENT INTERANTENNA SPACING WHERE THE FIRST RECEIVER ANTENNA IS $\lambda$ AWAY FROM THE TRANSMITTER ANTENNA

| Inter-Antenna Spacing ($\lambda$) | $\Xi_1$ | $\Xi_2$ | $\Xi_3$ | $\Xi_4$ | $\Xi_o$ |
|---|---|---|---|---|---|
| 0.5 | 0.139 | 0.286 | 0.425 | 0.484 | 0.359 |
| 0.4 | 0.142 | 0.300 | 0.412 | 0.475 | 0.356 |
| 0.3 | 0.152 | 0.242 | 0.405 | 0.445 | 0.333 |
| 0.2 | 0.105 | 0.252 | 0.378 | 0.433 | 0.318 |
| 0.1 | 0.103 | 0.232 | 0.382 | 0.307 | 0.276 |

intensities recorded at the odd-numbered antennas for the training. By doing so, we double the interantenna spacing. In this case, the overall inversion error increases to 0.236. We continue decreasing the number of receiver antennas by eliminating the even-numbered ones of the previous set till the interantenna spacing becomes $\lambda/2$, where we only have four receiver antennas. As listed in Table II, the overall error increases as we increase the distance between neighboring antennas.

One might claim that changing the antenna spacing while keeping coverage the same actually means reducing the amount of information used for training, and hence, the increasing error is inevitable, which makes the investigation unfair. This is why, as a next step, we consider a slightly different configuration: now, there are 20 receiver antennas located at $(x = \lambda + [m-1] \times d, z = \lambda/2)$, where $m$ is an integer representing the antenna number from 1 to 20 and $d$ is the interantenna spacing. The transmitter antenna's location is the same as before $(x = 0, z = \lambda/2)$. The training data set has 20 736 samples. As listed in Table III, the prediction error of our NN implementation decreases as we decrease $d$ from $0.5\lambda$ to $0.1\lambda$ at the steps of $0.1\lambda$.

When we further increase the number of antennas in the first configuration or decrease the interantenna spacing in the second configuration, we observe almost no improvement in the overall accuracy. In the light of these observations, we might claim that: 1) increasing the number of antennas helps with reducing the error but this reduction is not linear and 2) similar to suggested mesh sampling density in CEM, use of 20–30 antennas per wavelength might be the ideal case for the highest accuracy, but further increase in the number of receiver antennas is not helpful.

Notice that, even though we have the same interantenna spacing ($d = 0.5\lambda$) in the first rows of Tables II and III, the former has a smaller error. There are mainly two reasons behind this difference (location of the antennas and distance between transmitter and receiver antennas), which we discuss in Section V-C.

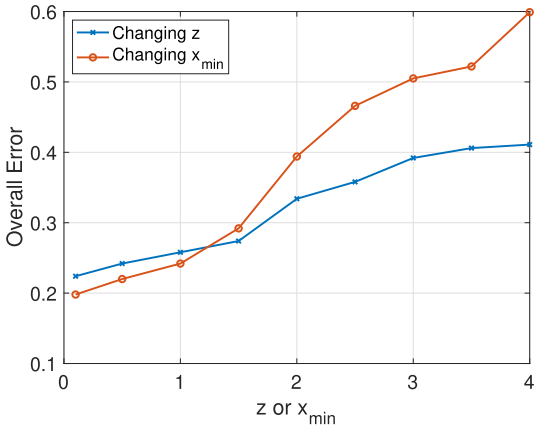


Fig. 8. For the configuration shown in Fig. 1(a), blue and red curves show how the overall error ($\Xi_o$) changes as a function of the location of the receiver antennas along the $z$- and $x$-axes, respectively.

### C. Location of the Antennas Versus Inversion Accuracy

First, let us examine how the overall error changes when we change the location of these 20 receiver antennas vertically or horizontally in the top layer. The configuration is shown in Fig. 1(a). The transmitter antenna is located at $(x' = 0, z' = \lambda/2)$. Training and testing data sets have 20 736 and 1000 samples, respectively. When all the 20 receiver antennas, which are placed uniformly between $x = \lambda$ and $x = 2\lambda$, and are moved away from the top interface in the $\hat{z}$-direction from $z = 0.1\lambda$ to $z = 4\lambda$, the overall error increases from 0.224 to 0.411, as shown with the blue curve in Fig. 8. Similarly, when we shift the receiver antennas along the $\hat{x}$-direction from $x_{min} = 0.1\lambda$ to $x_{min} = 4\lambda$, while $z = 0.5\lambda$, the overall error increases from 0.198 to 0.599, where $x_{min}$ is the x coordinate of the receiver antenna closest to the transmitter antenna. These results mean that the accuracy increases when the transmitter and receiver antennas get closer to each other and to the domain of interest.

Thus, all the cases that we have discussed have receiver and transmitter antennas placed in the top layer, and the minimum error is achieved for the prediction of the first layer's relative electrical permittivity. Next, we place antennas in different layers and try to understand how the location of antennas affects the inversion accuracy. Training and testing data sets have 4096 and 1000 samples, respectively.

*Case-A:* This is the reference case, whose results are already provided in Fig. 6: The transmitter antenna is located at $(x = 0, z = \lambda/2)$; 20 receiver antennas are in the top layer $(\lambda \leq x \leq 2\lambda, z = \lambda/2)$. Table IV lists all the individual errors and the overall error both for this and the following cases.

*Case-B:* 20 receiver antennas are placed in the bottom layer $(\lambda \leq x \leq 2\lambda, z = -\lambda)$, as shown in Fig. 1(b). The transmitter antenna location is the same as Case-A. Not surprisingly, in Case-B, the smallest error is obtained for the prediction of the bottom layer's permittivity. However, here, the errors in predicting the inner layers' permittivities are lower than the Case-A. This can be explained as follows: in Case-A, the primary field term (the direct interaction between $R_x$ and $T_x$ [19]) is dominant, and as the waves propagate toward lower layers, they get weaker (much smaller than the primary field term), but, in case B, there is no primary field term, and the electromagnetic waves have to propagate through all the layers and interfaces to reach the receiver antennas placed in the bottom layer. The recorded field intensities form a more balanced data set in terms of magnitudes, and this leads to more efficient learning.

TABLE IV
ERROR VERSUS LOCATION OF THE ANTENNAS. $n_s = 8$

| Case | $\Xi_1$ | $\Xi_2$ | $\Xi_3$ | $\Xi_4$ | $\Xi_o$ |
|---|---|---|---|---|---|
| A | 0.156 | 0.321 | 0.656 | 0.472 | 0.442 |
| B | 0.587 | 0.094 | 0.399 | 0.252 | 0.380 |
| C | 0.029 | 0.045 | 0.033 | 0.027 | 0.035 |
| D | 0.107 | 0.270 | 0.162 | 0.144 | 0.181 |
| E | 0.062 | 0.050 | 0.044 | 0.067 | 0.057 |

*Case-C:* As shown in Fig. 1(c), now, there are 20 receivers uniformly distributed along the vertical line $(x = \lambda, -\lambda \leq z \leq \lambda/4)$. Since, here, we have antennas in the domain of our interest, this is not a valid electromagnetic inversion configuration, but it still might be helpful. As listed in the fourth row of Table IV, when we have multiple receiver antennas in each layer, all the errors decrease dramatically. Recognizing that it is obtained solely from data without any physics, this result can still be considered as a success of deep learning.

*Case-D:* This is a slightly modified version of the configuration discussed in Case-C. To investigate how the error changes with increasing distance between receiver and transmitter antennas, we move all the receiver antennas used in Case-C horizontally from $x = \lambda$ to $x = 10\lambda$. We observe that all the errors increase three times or more compared to Case-C. This result confirms that, when the distance between receiver and transmitter antennas increases, the accuracy drops. Considering LMGFs' oscillatory behavior in the far-field [19], this is not a surprising result.

*Case-E:* As shown in Fig. 1(d), this time, we have one receiver $(x = 0)$ and one transmitter antenna $(x = \lambda)$ in each layer. The $z$-coordinates of these antennas are $z = \{-\lambda, -5\lambda/8, -3\lambda/8, -\lambda/8, \lambda/4\}$. Here, we do not have multiple receiver antennas in each layer, but, by having $R_x$-$T_x$ pairs in each layer, we provide a direct interaction between $R_x$ and $T_x$ in each layer. The error values are very close to the Case-C results but not better in terms of magnitude.

Based on the results of the last three cases, we can suggest that, when we have antennas, which can receive waves immediately reflected from or transmitted through the objects within the domain of interest, neural networks have the potential to identify those objects with higher accuracy compared to the cases where antennas receive waves after propagating through more complicated paths (transmission through one interface first, then reflection from the bottom interface, and then another reflection from the upper interface, and so on).

### D. What Else Can Be Done to Increase the Accuracy?

In real-life applications, we might not always have antennas that can receive waves immediately reflected from or transmitted through the objects within the domain of interest. So what

else can we do to increase the accuracy? Let us revisit the configuration shown in Fig. 1(a) in which normalized root mean square errors are 0.039, 0.09, 0.175, and 0.118 with $n_s = 18$ training data set.

First, we can design more complex NNs by increasing the number of hidden layers [8]–[11], [13], [14], which can especially help with problems dealing with several objects, but it is not the case here. Second, we can build NNs with complex activation functions to improve the accuracy [29], but, since they are computationally expensive, they might slow down the training process significantly, especially, if the network is implemented with a programming language, which is not specifically designed for scientific computing, such as Python. Third, simple postprocessing methods, such as taking the average of the predicted values in the last few epochs [4], and advanced methods, such as the NN-based image processing algorithm implemented in [14], can increase the accuracy by balancing over and underestimates and increasing the contrast among multiple objects, respectively. Another method for slightly higher accuracy is using a nonuniform grid for the training. Similar to their success in hyperparameter optimization [30], training sets created on nonuniform grids can lead to better learning by providing both smaller and larger contrasts compared to a training data set created on a uniform grid. In Fig. 9, we show how much improvement that we can get by using a training set created on a nonuniform grid, which has $n_s^4$ randomly sampled permittivity values between 1 and 10 for each layer using MATLAB's default random number generator, i.e., $\boldsymbol{\epsilon}_p(n_s^4, 1:4) = 1 + 9 * \text{rand}(n_s^4, 4)$. We observe that the training data set created on a nonuniform grid yields a more accurate inversion than the uniform grid data set. The final normalized root mean square errors are 0.029, 0.05, 0.081, and 0.072 for the $n_s = 18$ case, which are almost half of the errors obtained with uniform grid. These are more promising numbers than the originals but still not as impressive as the ones obtained in Case-C (i.e., multiple receiver antennas passing through all layers). There is another possibility that we can investigate: a multifrequency (broadband) excitation that is our next test.

### *E. Spectral Versus Spatial Distribution*

Considering the important role of color in image recognition applications of deep learning, the use of broadband excitations should increase the accuracy of electromagnetic inversion applications of deep learning too. Very recent studies indeed confirm that training and testing a learning system with broadband excitations (either continuous or discrete) outperform the learning systems deploying single-frequency data sets [6], [16]. Here, we would like to investigate a slightly different research question. Which one is better: $n_r$ receiver antennas operating at a single frequency ($f_0$) or one receiver antenna operating at $n_r$ different frequencies ($f_m$) consecutively, where $f_m \leq f_0$ for $m = 1, 2, \cdots, n_r$? The criterion described for the frequency sampling is crucial because the use of higher frequencies, in other words, the use of shorter wavelengths, would provide an unfair advantage to the multifrequency system by increasing the resolution.

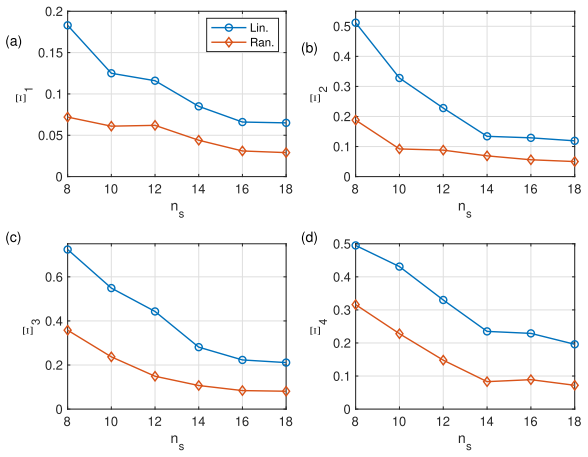


Fig. 9. (a)–(d) $\Xi_i$ versus $n_s$ for neural-networks trained by a dataset generated on uniform grids (blue curves with circle markers) and on non-uniform grids (red curves with diamond markers), for $i = 1, 2, 3,$ and 4, respectively.

TABLE V
ERROR VERSUS LOCATION OF THE ANTENNAS: MULTIFREQUENCY EXCITATION. TRAINING SET $n_s = 8$

| Case | $\Xi_1$ | $\Xi_2$ | $\Xi_3$ | $\Xi_4$ | $\Xi_o$ |
|---|---|---|---|---|---|
| F | 0.013 | 0.028 | 0.046 | 0.049 | 0.037 |
| G | 0.104 | 0.072 | 0.121 | 0.022 | 0.088 |

To test this idea while having a fair comparison, we create another group of training and testing data sets using only one transmitter and one receiver antenna operating at 20 distinct frequencies uniformly sampled from $c_0/2\lambda$ to $c_0/\lambda$, where $c_0$ is the speed of electromagnetic waves in vacuum. Transmitter antenna is located at $(x' = 0, z' = \lambda/2)$. We first place the receiver antenna in the top layer at $(x = \lambda, z = \lambda/2)$; then, we place the receiver antenna in the bottom layer at $(x = \lambda, z = -\lambda)$. The training set has 4096 samples. These two cases are labeled as Case-F and Case-G, respectively, and individual and overall errors are listed in Table V. Compared to Case-A and Case-B, where we have 20 antennas spatially distributed from $x = \lambda$ to $x = 2\lambda$ in the top and bottom layers, respectively, here, we have much smaller errors. Especially, for the cases with all the antennas in the top layer, there is more than an order of magnitude reduction in overall error (from 0.442 to 0.037). This result tells us that the minimum number of antennas required for accurate electromagnetic inversion can be lower than the one calculated with the Nyquist sampling theorem when multifrequency excitation is used for training and testing. By having fever antennas, transmission lines, and other required circuitry, this approach can reduce the volume and cost of the real electromagnetic inversion devices significantly.

### *F. Slightly More Complicated Problem: 1-D Pixel-Based Electromagnetic Inversion*

As for the last exercise, we slightly change the problem as follows. We divide the region between $z = 0$ and $z = -\lambda$ into 240 thin layers, and we place three objects with random thicknesses changing between $\lambda/240$ and $\lambda/2$ and permittivity

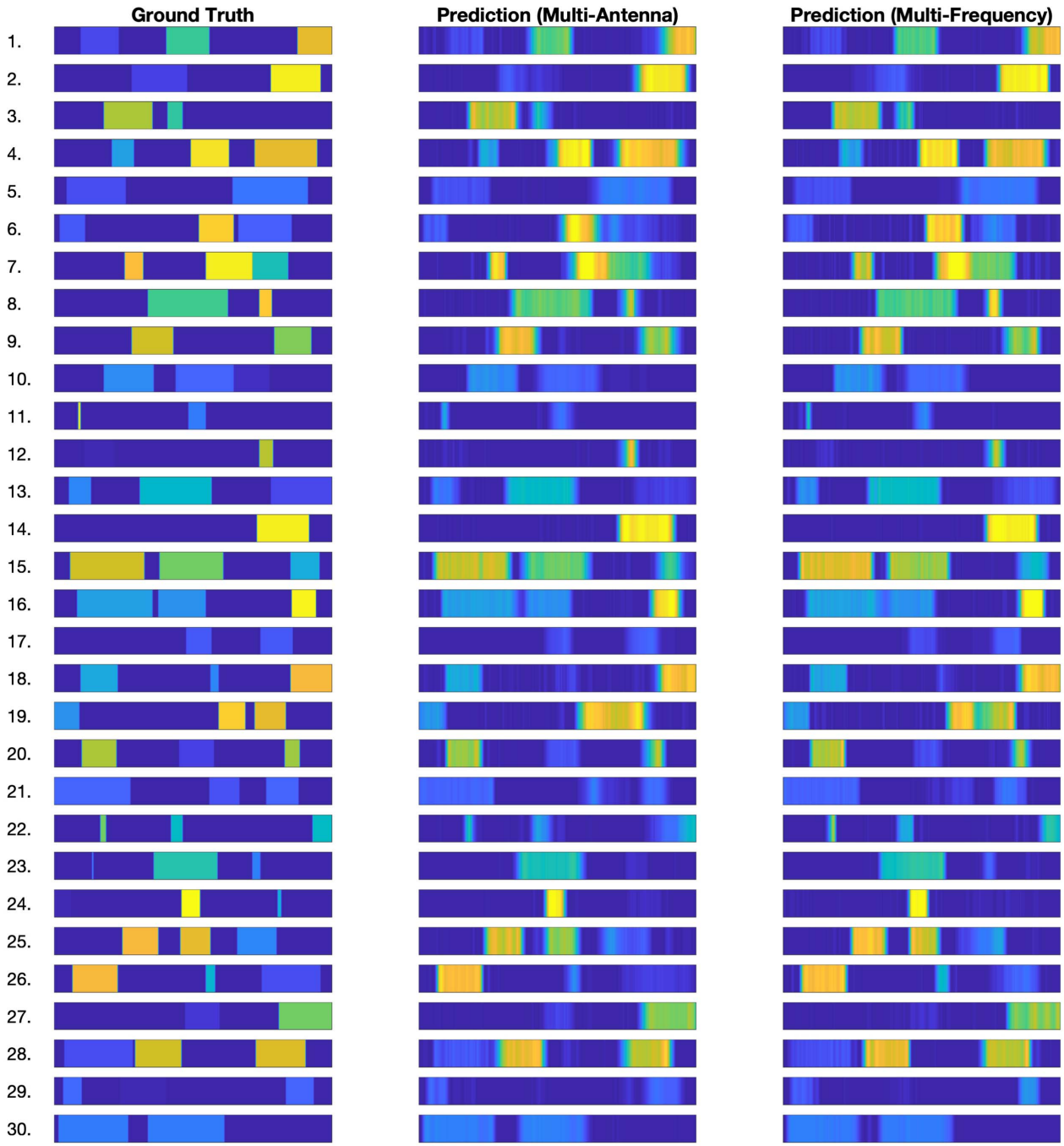


Fig. 10. Region between $z = 0$ and $z = -\lambda$ is divided into 240 layers. Three objects with a random thickness and relative permittivity are formed in this region. Left column: permittivity profiles of the first 30 examples of the 1000 cases studied. Middle and right columns: permittivity profiles predicted by the NN system with multiantennas/single frequency and singel antenna/multifrequency, respectively.

values randomly selected between 1 and 4. The left column of Fig. 10 shows 30 sample permittivity profiles.

The source is located at $(x' = 0,\ z' = 0.1\lambda)$. For the multiantenna case, we assume 20 receiver antennas uniformly placed along $0.5\lambda \leq x \leq 1.5\lambda, z = 0.1\lambda$. For the multifrequency case, the receiver antenna is at $(x = 0.5\lambda,\ z = 0.1\lambda)$, and we calculate LMGF sets at 20 frequency values from $c_0/2\lambda$ to $c_0/\lambda$. Training and testing data sets have 100 000 and 1000 samples, respectively, in each case. A visual comparison of the middle and right columns of Fig. 10 with the permittivity profiles shown in the left column indicates that both approaches provide a successful inversion such that both thin and thick objects with small and large contrasts are detected. The overall accuracy of the NN implementation for the multiantenna case turns out to be 92.6%, while the multifrequency implementation's accuracy

is 95.35%. When we shift all the antennas $0.4\lambda$ along the $z-$axis, these numbers reduce to 88.1% and 93.9%, respectively. The outcomes of this brief study align with the observations shared previously: 1) multifrequency systems outperform the single-frequency systems and 2) bringing antennas closer to the domain of interest increases the inversion accuracy.

## VI. Conclusion

LR, kNN, random forest, and NN-based learning systems are trained to solve a simplified 1-D electromagnetic inversion problem. In terms of accuracy, NN-based learning systems outperform the other machine learning methods implemented in this work. Numerical results show that using more antennas that are placed around the domain of interest as close as possible and training data sets created with random grids rather than uniform grids can increase the inversion accuracy, especially for the single-frequency applications. Multifrequency systems can provide more efficient learning while requiring a lower number of antennas than single-frequency systems.

## Acknowledgment

The author would like to express his gratitude to the editor, anonymous reviewers, and Thaiyalnayagi Karthik for their useful comments on earlier versions of this article. All the data sets generated for this work and all the notebooks where the machine learning algorithms are implemented are shared through the author's Github account at https://github.com/simsekergun/1D_EMI.

## References

[1] R. K. Mishra and A. Patnaik, "Neural network-based CAD model for the design of square-patch antennas," *IEEE Trans. Antennas Propag.*, vol. 46, no. 12, pp. 1890–1891, Dec. 1998.

[2] S. Molesky, Z. Lin, A. Y. Piggott, W. Jin, J. Vuckovic, and A. W. Rodriguez, "Inverse design in nanophotonics," *Nature Photon.*, vol. 12, no. 11, pp. 659–670, Nov. 2018.

[3] A. Massa, G. Oliveri, M. Salucci, N. Anselmi, and P. Rocca, "Learning-by-examples techniques as applied to electromagnetics," *J. Electromagn. Waves Appl.*, vol. 32, no. 4, pp. 516–541, Mar. 2018.

[4] E. Simsek, "Determining optical constants of 2D materials with neural networks from multi-angle reflectometry data," *Mach. Learn., Sci. Technol.*, vol. 1, no. 1, Feb. 2020, Art. no. 01LT01.

[5] V. Puzyrev, "Deep learning electromagnetic inversion with convolutional neural networks," *Geophys. J. Int.*, vol. 218, no. 2, pp. 817–832, Aug. 2019.

[6] R. Guo, M. Li, F. Yang, S. Xu, and A. Abubakar, "Application of supervised descent method for 2D magnetotelluric data inversion," *Geophysics*, vol. 85, no. 4, pp. WA53–WA65, Jul. 2020.

[7] H. M. Yao, M. Li, and L. Jiang, "Applying deep learning approach to the far-field subwavelength imaging based on near-field resonant metalens at microwave frequencies," *IEEE Access*, vol. 7, pp. 63801–63808, 2019.

[8] Z. Wei and X. Chen, "Deep-learning schemes for full-wave nonlinear inverse scattering problems," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 1849–1860, Apr. 2019.

[9] L. Li, L. G. Wang, F. L. Teixeira, C. Liu, A. Nehorai, and T. J. Cui, "DeepNIS: Deep neural network for nonlinear electromagnetic inverse scattering," *IEEE Trans. Antennas Propag.*, vol. 67, no. 3, pp. 1819–1825, Mar. 2019.

[10] Z. Wei and X. Chen, "Physics-inspired convolutional neural network for solving full-wave inverse scattering problems," *IEEE Trans. Antennas Propag.*, vol. 67, no. 9, pp. 6138–6148, Sep. 2019.

[11] H. M. Yao, W. E. I. Sha, and L. Jiang, "Two-step enhanced deep learning approach for electromagnetic inverse scattering problems," *IEEE Antennas Wireless Propag. Lett.*, vol. 18, no. 11, pp. 2254–2258, Nov. 2019.

[12] H. M. Yao, L. Jiang, and W. E. I. Sha, "Enhanced deep learning approach based on the deep convolutional encoder–decoder architecture for electromagnetic inverse scattering problems," *IEEE Antennas Wireless Propag. Lett.*, vol. 19, no. 7, pp. 1211–1215, Jul. 2020.

[13] L. Li, L. G. Wang, and F. L. Teixeira, "Performance analysis and dynamic evolution of deep convolutional neural network for electromagnetic inverse scattering," *IEEE Antennas Wireless Propag. Lett.*, vol. 18, no. 11, pp. 2259–2263, Nov. 2019.

[14] L.-Y. Xiao, J. Li, F. Han, W. Shao, and Q. H. Liu, "Dual-module NMM-IEM machine learning for fast electromagnetic inversion of inhomogeneous scatterers with high contrasts and large electrical dimensions," *IEEE Trans. Antennas Propag.*, vol. 68, no. 8, pp. 6245–6255, Aug. 2020, doi: 10.1109/TAP.2020.2990222.

[15] R. Guo, X. Song, M. Li, F. Yang, S. Xu, and A. Abubakar, "Supervised descent learning technique for 2-D microwave imaging," *IEEE Trans. Antennas Propag.*, vol. 67, no. 5, pp. 3550–3554, May 2019.

[16] R. Guo *et al.*, "Pixel- and model-based microwave inversion with supervised descent method for dielectric targets," *IEEE Trans. Antennas Propag.*, vol. 68, no. 12, pp. 8114–8126, Dec. 2020, doi: 10.1109/TAP.2020.2999741.

[17] B. Wei, E. Simsek, C. Yu, and Q. H. Liu, "Three-dimensional electromagnetic nonlinear inversion in layered media by a hybrid diagonal tensor approximation: Stabilized biconjugate gradient fast Fourier transform method," *Waves Random Complex Media*, vol. 17, no. 2, pp. 129–147, Apr. 2007.

[18] W. C. Chew, *Waves and Fields in Inhomogeneous Media*. Piscataway, NJ, USA: IEEE Press, 1995.

[19] E. Simsek, Q. H. Liu, and B. Wei, "Singularity subtraction for evaluation of Green's functions for multilayer media," *IEEE Trans. Microw. Theory Techn.*, vol. 54, no. 1, pp. 216–225, Jan. 2006.

[20] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM J. Optim.*, vol. 9, no. 1, pp. 112–147, Jan. 1998.

[21] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. New York, NY, USA: Springer, 2009.

[22] F. Chollet *et al. 2015 Keras*. Accessed: Jun. 1, 2020. [Online]. Available: https://github.com/keras-team/keras

[23] M. Abadi *et al. 2015 TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Accessed: Jun. 1, 2020. [Online]. Available: https://tensorflow.org

[24] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," 2017, *arXiv:1706.02515*. [Online]. Available: http://arxiv.org/abs/1706.02515

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[26] S. Chen, S. A. Billings, and P. M. Grant, "Non-linear system identification using neural networks," *Int. J. Control*, vol. 51, no. 6, pp. 1191–1214, Jan. 1990.

[27] S. Lek, M. Delacoste, P. Baran, I. Dimopoulos, J. Lauga, and S. Aulagnier, "Application of neural networks to modelling nonlinear relationships in ecology," *Ecol. Model.*, vol. 90, no. 1, pp. 39–52, Sep. 1996.

[28] S. Jiao, Y. Gao, J. Feng, T. Lei, and X. Yuan, "Does deep learning always outperform simple linear regression in optical imaging?" *Opt. Exp.*, vol. 28, no. 3, pp. 3717–3731, 2020.

[29] T. Kim and T. Adali, "Fully complex multi-layer perceptron network for nonlinear signal processing," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 32, nos. 1–2, pp. 29–43, 2002.

[30] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

**Ergun Simsek** (Senior Member, IEEE) received the B.S. degree from Bilkent University, Ankara, Turkey, in 2001, the M.S. degree from the University of Massachusetts, Dartmouth, MA, USA, in 2003, and the Ph.D. degree from Duke University, Durham, NC, USA, in 2006, all in electrical engineering.

He is currently working as the Director of the UMBC's Graduate Data Science Programs, Baltimore, MD, USA. His research interests include scientific computing for different applications in electromagnetics, photonics, geophysics, material science, and data science.