mathematics

MDPI

Article

A Novel Inversion Method for Electrical Impedance Tomography with a Radial Basis Operator Network

Jason Kurz [1,*], Andrew Pangia [2] and Taufiquar Khan [2]

[1] Department of Mathematics, University of Waikato, Hamilton 3240, New Zealand
[2] Center for TAIMing AI, University of North Carolina, Charlotte, NC 28223, USA; tkhan13@charlotte.edu (T.K.)
* Correspondence: jason.kurz@waikato.ac.nz

Abstract

We apply a new operator neural network to solve the Electrical Impedance Tomography (EIT) inverse problem. The EIT inverse problem involves reconstructing the conductivity inside a specific body or domain, given the electric potential along the boundary of said body. Mathematically speaking, the inverse problem is known to be severely ill-posed, that is, hard to reliably solve. However, we demonstrate the efficacy of our proposed algorithm utilizing the aforementioned neural network, dubbed the Radial Basis Operator Network (RBON) in its seminal work, when applied to the EIT inverse problem.

Keywords: EIT; inverse problems; operator networks; PDEs

MSC: 35R30

1. Introduction

It is a well-known fact that partial differential equations (PDEs) do very well representing physical systems. In particular, we consider a PDE representation of the Electrical Impedance Tomography (EIT) problem. This representation has a variety of uses, including medical imaging, image identification, detection of pipe obstructions (for oil pipelines etc.), or ground water flow, to name a few. However, due to the nature of the functions involved, analytically solving the EIT representation tends to be impossible.

There are several deterministic methods to numerically solve the inverse problem. Classical EIT inversion has long relied on PDE-constrained optimization techniques such as Gauss–Newton and adjoint-based methods, often combined with regularization strategies and sparsity-promoting priors to stabilize the ill-posed inverse problem [1–4]. A Lasso regression approach, called Tikhonov regularization, combined with 'sparsity regularization' can be utilized [1]; as a nonlinear minimization, iterative methods are applied [2]. Alternatively, methods can expand into the complex plane to obtain solutions that are analytic in the complex sense (infinitely differentiable in the complex plane $\mathbb{C}$), such as factorization [5] and a method known as 'd-bar' or $\bar{\partial}$. In short, $\bar{\partial}$ is defined to be a partial differential operator which computes how far from being analytic a function is [6]; this operator is used to linearize the PDEs as seen in [7], and expounded on in [3]. Due to the nonconvexity of the loss functions being minimized, stochastic minimizations are also considered, for instance, a Metropolis–Hastings approach involving Bayesian priors was applied to solve inverse problems to reasonable effect [4]. In addition, ref. [8] considers energy-based priors, which are found to produce good results when trained on the domain and its boundary. It should be noted that [8] assumes that the potentials are known

Check for updates

Academic Editor: Jean-Guy Caputo

Received: 10 November 2025
Revised: 1 January 2026
Accepted: 14 January 2026
Published: 19 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

throughout the entire domain while our method only utilizes the boundary in accordance with the reality of the problem.

Artificial Neural Networks (ANNs) have made promising inroads on this front as another numerical approximation method, given proper weights to their penalty functions and sizes and depths of their networks, such as work done in [9,10]. By incorporating the interior and boundaries of the PDEs into the networks as loss functions [11], a hybridized Physics-Informed Neural Network (PINN) [12] can be formulated to compute the function which solves the PDE. However, even these networks have drawbacks, taking an immense time to train (that is, compute its parameters), and needing to be retrained for each new problem instance. More specifically, traditional ANNs and their physics-informed counterparts were designed to represent a single function and hence provide a solution for a single particularized PDE system. Examples of this as applied to the EIT problem are referenced here [8] and here [13], where Bayesian priors are leveraged to reduce and stabilize the training of PINNs and vice versa. Even with the reduced training time, however, each new variation of the problem still required a network to train and solve. In contrast, we shift our focus to a category of ANNs that, once trained, provide the solution across an entire class of problems.

Due to the time constraints of training traditional ANNs and PINNs, we seek to learn classes of functions instead of individual functions: Neural Operators, or Operator Networks, were designed to learn the mapping between infinite dimensional function spaces [14]. Such networks are thus designed to receive function input and produce the resulting function output at specific query points. Thus, in lieu of a single solution to a PDE system under one specific set-up, they produce solutions to the PDE system over an entire range of variable set-up conditions and parameters. Once trained, these networks then provide an almost instantaneous solution for an entire class of PDEs. The current landscape of Operator Network theory contains DeepONet, a network utilizing deep neural networks [15], Fourier Neural Operators (FNOs) [16], and Laplace Neural Operators (LNOs) [17], with applications to PINNs [18]. More recently, Cho and Son introduced Physics-Informed Deep Inverse Operator Networks (PI-DIONs), which, like DeepONet, learn solution operators—but does so in a physics-informed, data-efficient manner [19]. Operator networks have excelled in modeling a variety of physical systems such as work on weather data [20], nonlinear shallow water equations [21], multiphase CO_2 dynamics in the Earth's subsurface [22], and biological material's time-dependent thermal response to incident laser radiation [23].

To solve the exponentially ill-posed [24] inverse problem in EIT, there exists a large amount of research [25,26] in the literature including both deterministic and statistical inversion methods for better image reconstruction for nonlinear inverse problems such as EIT. For a good review of classical methods for EIT, see [27]. The deterministic approach includes variational-type methods, linearized least square, or iterative methods for the fully nonlinear models [2,28]. There also exist other classical approaches such as the factorization method [5] and the d-bar method [7]. These methods require some prior knowledge of the conductivity distribution for regularization of the ill-posed inverse problem in EIT. In addition to deterministic methods, there are also a lot of new statistical methods which exist due to the computational advancements made in last several decades [29]. There are also studies to investigate model reduction errors for unknown geometry using the Bayesian inversion method [30,31], as well as results involving uncertainty quantification of MCMC-based image reconstruction [32]. In practice, both the deterministic and statistical regularization methods provide reasonable image reconstructions; however, practical applications require much higher resolution for better imaging diagnostics for detection of tumors, for example for medical imaging. Therefore, we look to take the next step

with a cutting-edge Operator Network, the Radial Basis Operator Network (RBON) [33]. This network, constructed using radial basis functions (RBFs), has demonstrated good performance by all metrics, outperforming LNO, FNO, and DeepONet [33]. We consider the EIT problem as presented with Dirichlet and Neumann boundary conditions to demonstrate RBON's efficacy.

2. EIT Problem Description

The basics of the Electrical Impedance Tomography (EIT) problem are as follows; more details can be gleaned from [4,34]: Let $\Omega \subseteq \mathbb{R}^d$, $d \in \{2,3\}$ be an open set with an appropriately smooth boundary, $\partial\Omega$, such that $\partial\Omega \cap \Omega = \emptyset$, with $\Omega \cup \partial\Omega$ a compact domain. For the EIT problem, electrical current is applied to $\partial\Omega$ in density f resulting in boundary potentials g; then there are two functions of interest, $\sigma : \Omega \cup \partial\Omega \to \mathbb{R}$ and $u : \Omega \cup \partial\Omega \to \mathbb{R}$. Function $\sigma \in L_\infty(\Omega)$ is the conductivity of Ω; for computational purposes, it is presumed that σ is known in a neighbourhood around the boundary. At the same time, $u \in L_2(\Omega)$ measures the current potentials of Ω. There are two forms of the EIT equation, the Neumann and the Dirichlet, which have different boundary conditions.

$$\begin{aligned} -\mathrm{div}(\sigma \nabla u) &= 0 \quad \text{in } \Omega, \\ \sigma \frac{\partial u}{\partial n} &= f \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

$$\begin{aligned} -\mathrm{div}(\sigma \nabla u) &= 0 \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{2}$$

We can switch between (1) and (2) as necessary using the traces Γ_N and Γ_D, which have to be defined below. The fact that we have computable unique solutions derives from the Lax–Milgram and Riesz Representation theorems; demonstrations thereof can be found in [4,35–37].

In practice, Ω is an object (like a body part) covered in electrodes which run minor levels of current through it to generate a map of its interior; f is a finite-dimensional vector of currents being passed through a set of electrodes attached to the boundary of Ω, while g is a finite-dimensional vector of potentials at those electrodes.

2.1. Forward Problem

For the forward problem, given the conductivity, σ, we wish to compute u, the patterns of the electrical potential inside the body Ω. Analytically, there is an operator $F : L_\infty(\Omega) \times L_2(\partial\Omega) \to L_2(\Omega)$ such that $F_N(\sigma, f) = u$ is a solution to (1) and $F_D(\sigma, g) = u$ is a solution to (2). We drop the subscripts and represent both F_N and F_D as F. More precisely, we can define the space

$$\tilde{H}^1 := \tilde{H}^1(\Omega) := \left\{ u \in L_2(\Omega) : \int_\Omega \sigma(x)|u(x)|^2 dx < \infty, \int_{\partial\Omega} u(s)\, ds = 0 \right\}.$$

Then, if n is the normal vector to $\partial\Omega$, we can define the dual spaces of solution functions Y and W as follows:

$$Y = \tilde{H}^{1/2}(\partial\Omega) := \left\{ g \in L_2(\partial\Omega) : \int_\Omega \sigma |\nabla_x F(\sigma, g)(x)|^2 \, dx < \infty, \int_{\partial\Omega} g(s) ds = 0 \right\}, \tag{3}$$

$$W = \tilde{H}^{-1/2}(\partial\Omega) := \left\{ f \in L_2(\partial\Omega) : f = \sigma \frac{\partial}{\partial n} F(\sigma, g), g \in Y \right\}. \tag{4}$$

To switch between (1) and (2), we can compute $\Gamma_D F$ or $\Gamma_N F$ as necessary, where the aforementioned traces can now be defined:

$$\Gamma_D : \tilde{H}^1 \to Y, \;\; u \mapsto u\Big|_{\partial\Omega}, \qquad \Gamma_N : \tilde{H}^1 \to W, \;\; u \mapsto \sigma \frac{\partial u}{\partial n}\Big|_{\partial\Omega}.$$

Note that, in our application, we are holding f, g as known functions (notation gets tricky here because we cannot call them constant functions), while u (for the forward problem) and σ (for the backward, or inverse, problem) alternate between being the variable function to be solved for. In practice, the analytical computation of F ranges from difficult to impossible, so, instead, an approximation via Finite Element Methods (FEMs) is computed, the process for which we detail below.

Begin with, instead, a weak formulation of (1): that is, given a set of functions which we like, $v \in V \subset \tilde{H}^1$, we solve the inner product of v with phantom $F(\sigma, f)$. That is, we wish to compute

$$\int_\Omega \sigma \nabla F(\sigma, f)(x) \cdot \nabla v(x) \, dx = \int_{\partial\Omega} f(s) \Gamma_D v(s) ds \tag{5}$$

for each $v \in V$. Note that F and V reside in an infinite dimensional space, but we ignore this detail and assume there is instead a finite dimension subspace V_K made up of basis functions $v_1, v_2, \ldots v_K$, which converges to $L_2(\Omega)$ as $K \to \infty$. Note that the v_k are analogous to the unit polynomials $x^k \in C(\mathbb{R}^d)$.

With $K < \infty$ chosen, we can then note that we want to solve for an approximation $u_K = \sum_{m=1}^K c_m v_m$ of $F(\sigma, f)$. So we have a system of equations, which we solve for c_m:

$$\begin{aligned} \int_\Omega \sigma \nabla u_K(x) \cdot \nabla v_k(x) \, dx &= \int_{\partial\Omega} f(s) \Gamma_D v_k(s) ds, \quad k = 1, \ldots, K \\ \sum_{m=1}^K \left(\int_\Omega \sigma \nabla v_m(x) \cdot \nabla v_k(x) \right) c_m \, dx &= \int_{\partial\Omega} f(s) \Gamma_D v_k(s) ds, \quad k = 1, \ldots, K. \end{aligned} \tag{6}$$

We can shorten (6) by defining vectors $c, \tau \in \mathbb{R}^K$ and a matrix $A \in \mathbb{R}^{K \times K}$ such that

$$A_{mk} := \int_\Omega \sigma \nabla v_m(x) \cdot \nabla v_k(x), \quad \tau_k := \int_{\partial\Omega} f(s) \Gamma_D v_k(s) ds :$$

then (6) can be rewritten as

$$Ac = \tau. \tag{7}$$

Solving this system is known as the Galerkin FEM [38,39].

2.2. Inverse Problem

Conversely, we may wish to use electrical current applied to $\partial\Omega$ to discover what its internal conductivities, σ, are; this may, for instance, help us identify the nature of cysts in the body, locate blockages in oil pipelines, or any other forms of anomaly identification. Symbolically, this would be undoing $F(\sigma, f) = u$. However, this is the inverse problem, which is hard to solve due to its ill-posedness and general instability with respect to noise.

As a result, the way that the inverse problem is solved is as follows. Take integers $I, J \in \mathbb{N}$ and a set of measured potential datapoints $g(s_j)$, $s_j \in \partial\Omega$, for all $j \in [J] := \{1, 2, \ldots, J\}$.

Denote $\hat{g}(\sigma; s) = \Gamma_D F(\sigma, f; s) \in \tilde{H}^{1/2}(\partial\Omega)$ and create a loss function

$$L_{inv}(\sigma) = \sum_{j=1}^{J} \left(\hat{g}(\sigma)(s_j) - g(s_j)\right)^2 + \lambda R(\sigma) \tag{8}$$

where $\lambda > 0$ is a regularization parameter chosen heuristically as a weight for Regularization operator $R : L_\infty(\Omega) \to \mathbb{R}$. Note that the Inverse Problem is assuming that F is known, so we are trying to minimize (8) with respect to σ; the minimizer of which will be the solution to the Inverse Problem. A summary of current methods to solve the Inverse Problem is detailed in Section 1.

2.3. Operators Approach to PDEs

Having defined two forms of the EIT problem in (1) and (2), we note that we can generate datapoints by applying forward problems, and use the resulting solutions to interpolate the potential $u = F(\sigma, f)$. Assuming that f (and g) are known, we can redefine $\mathcal{F} := F(\cdot, f)$ so that we are trying to determine $\mathcal{F}(\sigma) = u$, or, for the inverse problem where potential is known, determine $\sigma = \mathcal{F}^{-1}(u)$. This process can be generalized beyond the EIT problem.

We define a PDE in general mathematical terms to make obvious its operator aspect, but continue using the notation σ and u to emphasize the generalization from the EIT problem. Without loss of generality, let $d \in \mathbb{N}$, $C^k(\mathbb{R}^d)$ be the set of continuous, *kth*-differentiable functions over $\mathbb{R}^n$, $u \in C^k(\mathbb{R}^d)$, $\sigma \in C^k(\mathbb{R}^d)$; let the interior and boundary conditions $I, B : C^k(\mathbb{R}^d) \times \mathbb{R}^d \to C^k(\mathbb{R}^d)$. Note that, in practice, if we further define

$$\mathcal{P}(u(x), \sigma(x); x) := \begin{bmatrix} I(u(x), \sigma(x); x) \\ B(u(x), \sigma(x), x) \end{bmatrix}, \tag{9}$$

then any PDE can be represented by $\mathcal{P}(u, \sigma; x) = 0$. In normal person's terms (for a sufficiently narrow definition of normal), I and B are the interior and boundary conditions of the PDE, which apply various mathematical operations, including differentiation, to u and σ and their input, x.

Given σ, $\mathcal{P}$ defines $u := H(\sigma)$. In the case of the inverse problem, we can consider the opposite: $\sigma = G(u)$ and then approximate G using (15). To train G, we can generate data using the forward problem, which is solvable and stable, and then use the data to create a model which is predicting σ, that is, solving the inverse problem.

3. Radial Basis Operator Network

Because of the mathematical exposition required to state the algorithm we construct, we provide two convenient definitions, along with the context of the symbols we use. A brief table thereof is found in Table 1, with more elaborate explanations in Section 3.

Definition 1. *Let $N \in \mathbb{N}$. Then $[N] := \{1, 2, \ldots, N\}$.*

Definition 2. *Let $a \in \mathbb{R}^n$, $b \in \mathbb{R}^m$. Then the Kronecker Product, $\otimes$, is defined as the iterative elementwise multiplication of a with b. That is,*

$$a \otimes b := \begin{bmatrix} a_1 b \\ a_2 b \\ \vdots \\ a_n b \end{bmatrix}.$$

Note that $a \otimes b \in \mathbb{R}^{nm}$.

In order to create the network, we need the following set from which we can draw basis functions.

Definition 3. *Given space* $\mathbb{R}^d$*, the Schwartz Space,* $S(\mathbb{R}^d) \subset C^\infty(\mathbb{R}^d)$*, is the space of infinitely differentiable functions whose derivatives vanish at faster than polynomial speed.*

To more clearly define a partitioning of function spaces which appears in Section 5.3, we also provide the following definition of the angle between two vectors.

Definition 4. *Let* $a, b \in \mathbb{R}^n$*. Then the cosine similarity is*

$$\cos\theta_{ab} = \frac{a \cdot b}{||a|| \cdot ||b||}. \tag{10}$$

Table 1. A brief list of the symbols being used. Where applicable, we include what the symbols mean for the EIT problem under consideration.

Symbol	Space	Purpose	EIT Meaning
		Variables	
Ω	$\mathbb{R}^d$	compact space	body of interest
x	$K_1 := \partial\Omega \subseteq \mathbb{R}^n$	PDE forward solution input	body boundary location
y	$K_2 := \Omega \subseteq \mathbb{R}^d$	PDE inverse solution input	body internal location
$x_{idx}, idx \in [m]$	$\partial\Omega \subseteq \mathbb{R}^n$	PDE forward input sample	body boundary sample
$y_\ell, \ell \in [L]$	$\Omega \subseteq \mathbb{R}^d$	PDE inverse solution input sample	body internal sample
m, L	$\mathbb{N}$	discrete selections of x and y	
		Functions	
u	$\tilde{H}^1$	inverse input function	(boundary) electric potential
σ	$L_\infty(\Omega)$	inverse solution function; we want this	internal conductivity
$u_j, j \in [J]$	$\tilde{H}^1$	particular input	jth potential
$\sigma_j, j \in [J]$	$L_\infty(\Omega)$	particular solution	jth conductivity
J	$\mathbb{N}$	size of training set	
		Network	
$\phi(\cdot, \mu, s)$	$S(\mathbb{R}^m), S(\Omega)$	Radial Basis Functions	
μ	$\mathbb{R}^m, \Omega$	RBF centers	
s	$\mathbb{R}$	RBF spreads	
b	$(S(\mathbb{R}^m))^M$	branch network vector	
t	$(S(\Omega))^N$	trunk network vector	
$\xi_{ik}, i \in [M], k \in [N]$	$\mathbb{R}^{NM}$	scaling parameters	
M, N	$\mathbb{N}$	size of networks	
$G(u)$	$C(\tilde{H}^1)$	map of u to σ	
$G^\dagger(u)$	$C(\tilde{H}^1)$	approximate map G	

For the following general application, we keep the input function and output function the same as in the EIT. That is, let $K_1 \subseteq \mathbb{R}^n$ be the set of variables x for vector-valued function $u \in U \subseteq (C(K_1))^J$; similarly, let $K_2 \subseteq \mathbb{R}^d$ be the set of variables y for the function $\sigma := G(u) \in C(K_2)$ created by applying (unknown) operator G to u. In our application to the EIT problem, $n = 1$ while $d = 2$; that is, $K_1 := \partial\Omega$ and $K_2 := \Omega$. However, we define the RBON for general use, so refer to K_1 and K_2 in this section, but reference how to read them for the EIT problem.

To compute σ, we need a number of RBFs, $M + N$; we let M be the number of RBFs applied on the input points, $x \in K_1$, while N is the number of RBFs applied on the output points, $y \in K_2$, called query points. In the EIT context, x is coming out of $\partial\Omega$ while y is coming out of Ω. Note that J is the number of samples used to train the neural network. Then u_j will typically be a particular value function resulting in a particular solution, σ_j, to the PDE being solved for all $j \in [J]$.

For the Radial Basis Operator Network (RBON), we define two vector functions of RBFs: $b \in (S(\mathbb{R}^m))^M$, which is applied to the u functions evaluated at particular datapoints x in the domain of input functions u, and $t \in (S(\mathbb{R}^d))^N$, which is applied to datapoints y in the domain of desired general solution σ. A brief note on the nomenclature: note that b stands for 'branch' while t stands for 'trunk'; this stems from the notion that the two networks are built on top of each other to form the operator; the query points, y, are at the innermost section, so are affiliated with the internal core of a tree, the trunk, while the input function u is layered on the outside, that is, the top of the tree, the branch(es).

We choose the RBFs to be $\phi \in S(\mathbb{R}^{2n+1})$, a Gaussian (Normal) distribution defined in the following manner, using x as a general variable, which will be replaced in our network as necessary. While there are other forms of RBF, such as the multiquadric function, we found the Gaussian to be the most numerically stable. Note that, because this is a Gaussian function, we utilize the standard statistical definition of μ, but use s as the standard deviation in lieu of overloading standard deviation parameter σ:

$$\phi(x, \mu, s) := \exp\left(-\frac{||x - \mu||^2}{2s^2}\right), \tag{11}$$

where $\mu \in \mathbb{R}^n$ and $s \in \mathbb{R} \setminus \{0\}$ define the center and spread of ϕ, as the mean and standard deviation. In practice, $\phi : \mathbb{R}^{2m+1} \to \mathbb{R}$, or $\phi : \mathbb{R}^{2d+1} \to \mathbb{R}$.

Define RBF evaluation vectors b and t:

$$b(\cdot) := [\phi(\cdot, \mu_1^b, s_1^b), \phi(\cdot, \mu_2^b, s_2^b), \ldots, \phi(\cdot, \mu_M^b, s_M^b)]^T \tag{12}$$

$$t(\cdot) := [\phi(\cdot, \mu_1^t, s_1^t), \phi(\cdot, \mu_2^t, s_2^t), \ldots, \phi(\cdot, \mu_N^t, s_N^t)]^T, \tag{13}$$

where RBF centers $\mu_i^b \in \mathbb{R}^m$ and $\mu_k^t \in \mathbb{R}^d$ and spreads s_i^b and s_k^t, where $i \in [M]$, $k \in [N]$.

Then our fundamental error function is

$$|G(u)(y) - G^\dagger(u)(y)| < \epsilon \tag{14}$$

where $\epsilon > 0$ is small and, if b_i, t_k are the ith and kth elements of vector functions b and t,

$$G^\dagger(u)(y) := \sum_{i=1}^{M} \sum_{k=1}^{N} \xi_{ik} b_i(u_j^m) t_k(y). \tag{15}$$

Note that, given a set of m input sample points $x_{idx} \in K_1$, we have defined the evaluation vector

$$u_j^m := (u_j(x_1), u_j(x_2), \ldots, u_j(x_m)) \in \mathbb{R}^m.$$

A graphical representation of the RBON computation of $G^\dagger$ can be found in Figure 1.

Given sufficiently large M and N, we are guaranteed that $\epsilon > 0$ can be made as small as necessary for all $y \in K_2$, that is, we are guaranteed that we have a function $G^\dagger(u)$ which acts close enough to the unknown function $\sigma = G(u)$ for all intents and purposes [40]. In practice, J is the training set.

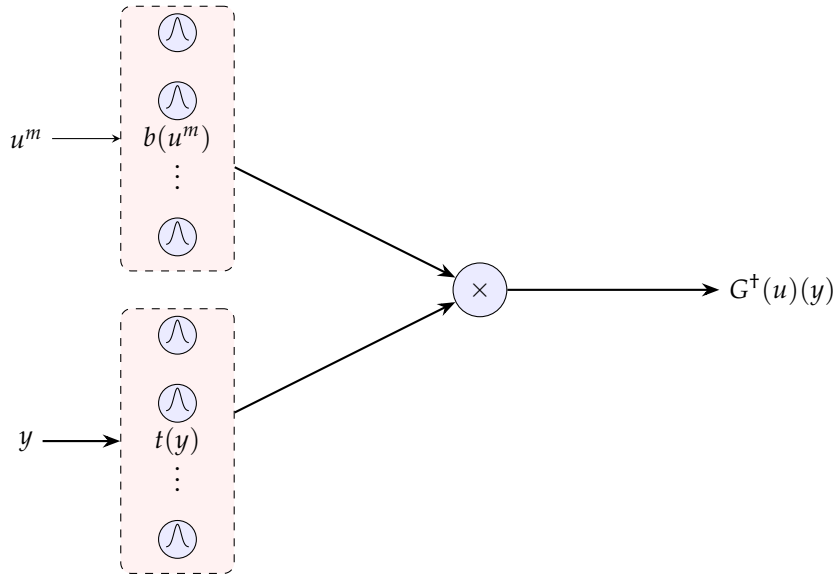


Figure 1. An illustration of the network $G^\dagger(u) \approx \sigma$.

4. Algorithm

Note that, so far, we have presented a general approximation $G^\dagger(u)(y)$. We now present the process involved in computing the parameters ξ_{ik}, μ_i^b, μ_k^t, s_i^b, s_k^t, where $i \in \{1, 2, \ldots, M\}$, $k \in \{1, 2, \ldots, N\}$ in Algorithm 1. Because the function space we are trying to predict is infinite-dimensional, we need to be able to select the RBFs correctly. As a result, in line 1 of Algorithm 1, we take the data that we have, $K_1 \times K_2$, and check to make sure that the RBFs we have properly covered that information. We use a K-means method [41,42], first for the data $x \in K_1$, to compute μ_i^b and s_i^b, and next for the data $y \in K_2$, to compute μ_k^t and s_k^t. The means μ_i^b and μ_k^t are the converged-to means from the algorithm, while the deviations s_i^b and s_k^t are the computed standard deviations from the clusters.

Now that we have the RBFs chosen, we need to compute the last parameters, ξ_{ik}. For conciseness, we denote $\xi \in \mathbb{R}^{MN}$ as the vector containing ξ_{ik} for all $i \in [M], k \in [N]$. We assume that we have available data $X \in \mathbb{R}^{n \times m}$, and $Y \in \mathbb{R}^{d \times L}$, with the rows of X acting as the x_i and the rows of Y acting as the y in (15). We also assume that we have some number $J \in \mathbb{N}$ of particular functions u_j resulting in known solutions σ_j, $j \in [J]$. We compute a matrix $V \in \mathbb{R}^L \times \mathbb{R}^J$ defined by

$$V_{\ell j} := \sigma_j(y_\ell), \tag{16}$$

for each $j \in [J]$, $\ell \in [L]$. We can then compute what ξ_{ik} is by using least squares with the standard pseudoinverse. Applying the Kronecker Product in Definition 2 to (12) and (13),

$$b(u_j^m) \otimes t(y_\ell) = \begin{bmatrix} b_1(u_j^m)t(y_\ell) \\ b_2(u_j^m)t(y_\ell) \\ \vdots \\ b_M(u_j^m)t(y_\ell) \end{bmatrix} \in \mathbb{R}^{MN}, \tag{17}$$

where b and t are defined in (12) and (13), respectively, we can define matrix

$$\Phi_j := \left[b(u_j^m) \otimes t(y_1),\ b(u_j^m) \otimes t(y_2),\ \ldots,\ b(u_j^m) \otimes t(y_L) \right] \in \mathbb{R}^{MN \times L}.$$

Now, for all $j \in [J]$, we can compute $\xi^j \in \mathbb{R}^{MN}$ using least-squares regression:

$$\Phi_j^T \xi^j = V_{:j}. \tag{18}$$

where $V_{:j} := [V_{\ell j}]_{\ell=1}^{L}$. We then average over the net of particular functions, J, to obtain

$$\xi = \frac{1}{J} \sum_{j=1}^{J} \xi^j. \tag{19}$$

Algorithm 1: Radial Basis Operator Network

Input : Dataset $(X, Y) \in \mathbb{R}^{n\times m} \times \mathbb{R}^{d\times L}$, known input functions $u_j \in C(\mathbb{R}^n)$, known output functions $v_j \in C(\mathbb{R}^d)$ number $M, N \in \mathbb{N}$ of RBFs

Output: $G^\dagger(u) : \mathbb{R}^d \to \mathbb{R}$

Step 0: Construct train/test split

Step 1: Apply K-means algorithm to $u_j(x)$, $j \in [J]$ with $K = M$ to get centers μ_i^b and spreads s_i^b, $i \in [M]$;
Apply K-means algorithm to y, with $K = N$ to get centers μ_k^t and spreads s_k^t, $t \in [N]$

Step 2: **for** *each* $\ell \in L$ **do**
  Compute ξ^ℓ via least squares from (18)

Step 3: Average over ℓ to get ξ_{ik} in (19)

5. Algorithm Adjustments

In the development of $G^\dagger$, implementation of a couple variations can improve the performance of the model significantly. Firstly, the computation of ξ can be altered by utilizing an elastic-net regularization. After the computation of $G^\dagger$, we either linearly transform it or put it through a one-layer Multilayer Perceptron (MLP) implemented via Julia. Lastly, we partition the function space in which σ resides, $L^\infty(\Omega)$, via a partition on the input function space and assumed continuity of the operator. Note that we have this assumption because the true operator G is defined entirely by the EIT problem, which is continuously stable in the forward direction.

5.1. Elastic-Net Regularization

The output for the RBON model in this context is the conductivity at a mesh node, expressed as a linear combination of a product of RBFs. Once the nonlinear kernels are fixed, this creates a linear (in terms of the parameters) system. Solving said system in the context of EIT is thus solving a potentially ill-conditioned and high-dimensional linear regression problem for the weights. For this reason, we applied a well-known regularization approach referred to as elastic-net. The following elastic-net regularization parameters, λ, α, can be used to better compute ξ in (18):

$$\min_{\xi} \left(V_\ell - \Phi_\ell \xi^\ell\right)^T \left(V_\ell - \Phi_\ell \xi^\ell\right) + \frac{\lambda(1-\alpha)}{2} \xi^T \xi + \lambda\alpha ||\xi||_1, \tag{20}$$

we pick $\lambda = 1, \alpha = 0.25$ heuristically. This choice parallels classical regularization strategies used in PDE-constrained EIT inversion, where stabilizing ill-posed linear systems through sparsity and smoothness priors is standard practice. The ℓ_1 penalty encourages sparsity in the weights so not every product of basis functions will contribute meaningfully to the construction, but favors the important products. The ℓ_2 penalty stabilizes the solution where the products of basis functions may be highly correlated. This type of regularization was chosen since EIT is an ill-posed inverse problem where small errors in the boundary data

cause huge oscillations in naive reconstructions so the elastic-net acts as a regularizer that controls variance and enforces smoother and more physically realistic conductivity fields.

Unlike conventional neural networks, RBON weights are computed in a single step via elastic-net regularized linear regression (Equation (20)) rather than iterative backpropagation; therefore, there is no convergence history. Even when an MLP output layer is added, the RBON weights remain frozen, and only the MLP parameters are trained to smooth residual patterns.

5.2. Transformations

A final output layer was applied to the representation $G^\dagger$, comparing a strictly linear transformation with a shallow MLP (one-layer). We define the Gaussian Error Linear Unit (GELU) function $GELU : \mathbb{R}^n \to \mathbb{R}^n$ as follows using a tanh approximation to the CDF of the Gaussian function:

$$GELU(x_i) = x_i \cdot \left(0.5\left(1 + \tanh\left(\sqrt{2/\pi} \cdot \left(x_i + 0.044715x_i^3\right)\right)\right)\right)$$

for all $i \in [n]$.

Then we can compute $G^\dagger$ as follows:

$$G^\dagger(u) = A \cdot GELU(G^\dagger(u)) + b \tag{21}$$

where A and b are an appropriately sized matrix and vector, respectively. This layer serves as a regularized correction map that smooths the RBON estimate further. An alternative to the output layer would be to update the centers and scaling parameters of the RBF kernels, but this introduces additional non-convexity and complicates the optimization. Appending the lightweight output layer (linear or shallow MLP) provides sufficient flexibility to capture residual structure while keeping the training process stable and computationally efficient. We can incorporate this extra output layer visually in Figure 2.

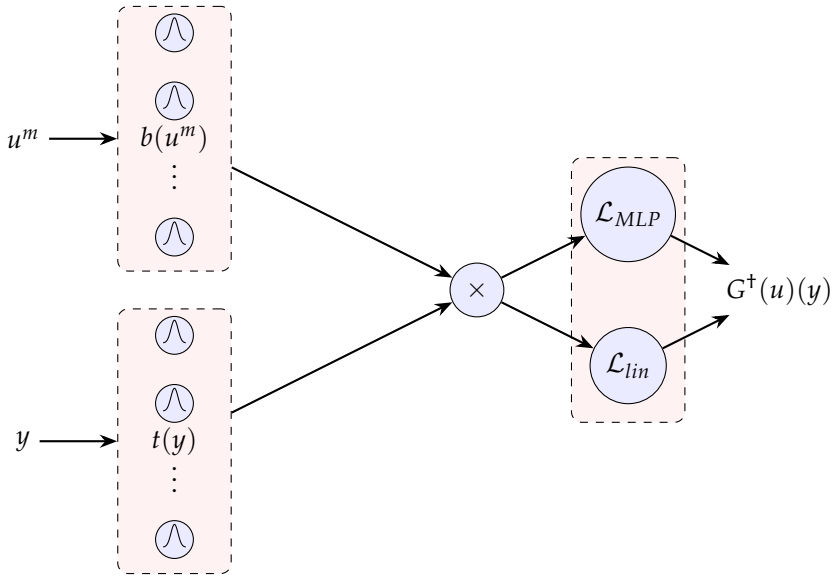
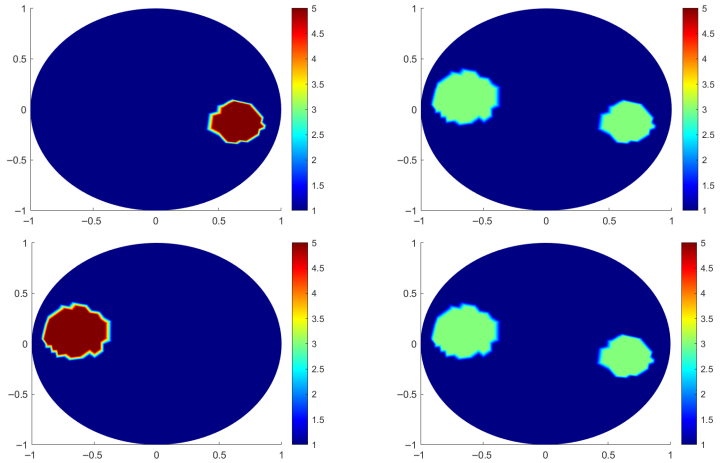


Figure 2. $G^\dagger(u)$ computed using Section 5.2.

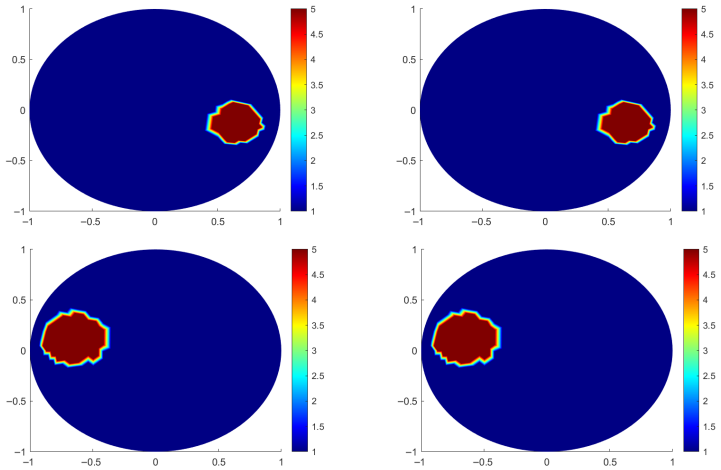
5.3. Local Operator Networks

When considering this problem as fitting a model to a data distribution, the locations of the inhomogeneity values act as the datapoints being fit. However, as with all statistical modeling, if the samples are sufficiently different, then models of insufficient complexity, in this case with N, M being too small, fail to fit them well. As a case study, take $J = 2$ and

consider the two σ_1, σ_2, in Figure 3: the high-conductivity regions are diametrically opposite each other, resulting from two very different potentials u. When fitting a model $G^\dagger$ based only on these two samples, we find it fails when $M = N = 1$. Creating one model which can predict both of these over the entire region Ω can only occur when the number of RBFs is high enough, that is, $N = M = 2$. This indicates that we simply need to have one RBF for every $j \in [J]$; however, computation time increases prohibitively with the number of RBFs.



(**a**)



(**b**)

Figure 3. Sample data showing why we need higher M and N. True data is on the left, and the prediction is on the right. (**a**) Given $J = 2$, predict using $N = M = 1$. The σ_j are on the left, while the approximations $G^\dagger(u_j)$ are on the right. Note that the model just puts both high-conductivity regions at the same time, but about half the value. (**b**) Given $J = 2$, predict using $N = M = 2$. Note that the model now fits the two datapoints perfectly.

As a workaround, we limit our M and N, but instead cluster the datapoints. For the clustering metric, we could use p-norms, but, in practice, the size of the vector discretizations renders the metric too large. As a result, we cluster based on the angle between vectors, computed using (10). Note that (10) provides θ such that $\cos\theta \in [-1, 1]$, so we can use $1 - \cos\theta \in [0, 2]$ to measure the distance between the centers of the high-conductivity locations where 0 occurs for locations close together and 2 occurs for locations far apart. Based on these clusters, we then partition the region accordingly to allow the data in those sections to remain sufficiently similar. By computing one $G^\dagger$ per section, we can keep M and N manageable sizes (around 20) and still increase accuracy.

We partition the space of input functions $\mathcal{U} \subset (\tilde{H}^1)^J \subset (C(\partial\Omega))^J$ into a number, T, of subspaces $\mathcal{U}_\tau$, $\tau \in [T]$, such that $\bigcup_{\tau\in[T]} \mathcal{U}_\tau = \mathcal{U}, \mathcal{U}_{\tau_1} \cap \mathcal{U}_{\tau_2} = \emptyset$. We apply Algorithm 2 to train T networks, $G^\dagger_\tau(u)$, in the following manner. Consider set $[J]$ and partition it into T subspaces by applying K-means to the u_j^m using $1 - \cos\theta$ from (10) as the distance. Then $G^\dagger_\tau(u)$ is computed using u_j and σ_j for $j \in J_\tau$ as seen in Sections 4 and 5.

Algorithm 2: Data Partitioning

Input : J Sample data points, u_j and σ_j, $j \in [J]$, cluster number T
Output: $G^\dagger_\tau(u) : \mathbb{R}^d \to \mathbb{R}$
Step 1: Apply K-means algorithm to $u_j(x)$, $j \in [J]$ with $K = T$ to make clusters which partition $[J]$.
Step 2: **for** *each* $j \in J$ **do**
  Compute $G^\dagger_\tau(u)$ using Algorithm 1 on $u \in \mathcal{U}_\tau$.

Then the overall operator is defined as follows:

$$G^\dagger_{loc}(u) := G^\dagger_\tau(u) \text{ where } u \text{ is closest to } \mathcal{U}_\tau$$

resulting in a partition of the output function space, a subset of $\mathcal{L}_\infty(\Omega)$. Prediction is done by sending the test input function to the 'closest' local RBON approximation based on the cosine similarity error between the input function and the stored cluster centers. Section 5.3.1 yields a guarantee that the ensemble of local operator representations can only improve the error in the approximation.

5.3.1. RBON Error for Local Approximation

We consider the error bounds on the local operator $G^\dagger_{local}$ in general. Let $\mathcal{U} \subset C(K_1)$ and let $(C(K_2), \|\cdot\|_\infty)$ be the Banach space of continuous functions on a compact $K_2 \subset \mathbb{R}^d$ with the sup-norm. Let $V \subset C(K_2)$ be compact and let $G : \mathcal{U} \to C(K_2)$ be continuous (possibly nonlinear). Let $\mathcal{F}^{-1} \subset C(\mathcal{U}, C(K_2))$ denote the hypothesis class of RBONs (considered as maps $\mathcal{U} \to C(K_2)$), endowed with the uniform norm

$$\|F^{-1}\|_{C(\mathcal{U},C(K_2))} := \sup_{u\in\mathcal{U}} \|F^{-1}(u)\|_\infty.$$

Assume the feature system is rich enough, that is, the kernels we choose can closely enough replicate the operator, so that $\mathcal{F}^{-1}$ is dense in $C(\mathcal{U}, C(K_2))$ under the topology of uniform convergence on the space of mappings. Then for every $\varepsilon > 0$ there exists $G^\dagger \in \mathcal{F}^{-1}$ such that

$$\sup_{u\in\mathcal{U}} \|G(u) - G^\dagger(u)\|_\infty < \varepsilon.$$

Shrinking the domain never worsens (i.e. improves) the worst-case error. Let $\mathcal{U}' \subset \mathcal{U}$ be nonempty. Define the (best achievable) uniform errors

$$\mathcal{E}_{\mathcal{F}^{-1}}(\mathcal{U}) := \inf_{G^\dagger(u)\in\mathcal{F}^{-1}} \sup_{u\in\mathcal{U}} \|G(u) - G^\dagger(u)\|_\infty, \qquad \mathcal{E}_{\mathcal{F}^{-1}}(\mathcal{U}') := \inf_{G^\dagger(u)\in\mathcal{F}^{-1}} \sup_{u\in\mathcal{U}'} \|G(u) - G^\dagger(u)\|_\infty.$$

Lemma 1 (Error improvement under local approximator)**.** *Assuming there exists minimizers* $G^\dagger_{local}$ *and* $G^\dagger_{global}$ *of* $\mathcal{E}_{\mathcal{F}^{-1}}(\mathcal{U}')$ *and* $\mathcal{E}_{\mathcal{F}^{-1}}(\mathcal{U})$*, respectively, then*

$$\mathcal{E}_{\mathcal{F}^{-1}}(\mathcal{U}') \le \sup_{u\in\mathcal{U}'} \|G(u) - G^\dagger_{global}(u)\|_\infty.$$

Proof. This follows immediately by the definition of the infimum and since $G^{\dagger}_{\text{global}}$ is one possible choice in $\mathcal{F}^{-1}$. □

Hence, given the existence of such an approximation, the local approximator yields more flexibility in representing the data on the localized set so that the error will be no worse than a global approximation when restricted to the same local set. By extension, a model made up of local models applied to a partition of the feature space $\mathcal{U}$ functions no worse than a single global model. We note that, in general, the infimum may not be attained in $\mathcal{F}^{-1}$. However, under the density assumption, there is an approximator that can be chosen so that the error is arbitrarily close to best achievable.

5.3.2. Runtime Complexity

An additional consideration is the computation time involved in decreasing the error. Note that the dominant steps involved in computing G are the K-means clustering to obtain the kernel RBFs and the matrix inversions involved in the elastic-net computation of ξ. However, the clustering is linear with respect to M and N, while the matrix $\Phi \in \mathbb{R}^{MN \times L}$ dictates the computation time of elastic-net, via its matrix inversion. However, matrix inversion can be assumed to be more than $O(M^2N^2)$. Even at $O(M^2N^2)$, though, any increases to M and N result in a quadratic increase in computation time while an increase in T will be a linear increase in computation time. Therefore it is more beneficial to utilize multiple local models with fewer kernels than vice versa.

6. Numerical Results

For our numerical work, we let Ω be the disc of radius one. We define $A \subset \Omega$ to have high conductivity, while $\Omega \setminus A$ has low conductivity. Define conductivity $\sigma : \Omega \to \mathbb{R}$ as follows:

$$\sigma(\omega) := \begin{cases} \sigma_{hi} & \text{if } \omega \in A \\ \sigma_{lo} & \text{otherwise} \end{cases} \tag{22}$$

The guarantee in [43] is for applying operators to continuous functions, which we do not have here; however, it is well-known that piece-wise discontinuous functions can be approximated by smooth functions; thus, we can proceed to learn the smooth function representation.

For this experiment let $\sigma_{lo} = 1$ and $\sigma_{hi} = 5$; we generate A in three different ways: we first create a disk of randomized radius in a random location within Ω; next, we create either one or two disks of randomized radius; lastly, we create two disks of randomized radius. To generate the data, we use Matlab's Partial Differential Equations Toolbox, *pde*, to apply Delaunay Triangulation to Ω. With the mesh generated thereby, we solve the forward problem for the potentials u on the boundary $\partial\Omega$ using *assempde*. To generate the data, the mesh resolution is 5249 points, the number of measures on the boundary is 256 points, and the current–voltage pattern is one source at

$$\sin(\arctan(y/x) + ((\text{sign}(x) - 1)/2) \cdot \pi).$$

Note that this approach is not necessarily limited by the number or shape of inhomogeneities. We also consider $A = A'$ defined by generating squares instead of disks as out-of-distribution data. Worth noting is that we train this network on disks and test on the squares.

6.1. Experiment Setup

Given Ω, we generate a dataset of 5000 disks with randomized radius and centers, using $J = J_{train} := 4000$ to train the model and reserving $J = J_{test} = 1000$ for the test set. We save σ_j, u_j, and the query points, $y_\ell \in \Omega$ (which remain constant for all $j \in [J]$), to a *.mat* file and apply Algorithm 2 using Julia. Note that σ_j are known, u_j are computed, and y are the node points from the mesh. We find heuristically that $N = M = 21$ works well. We consider $T \in \{1, 5, 10, 15\}$. As mentioned in Section 5.1, we also incorporate a standard elastic-net regression scheme into the loss function to help smooth out the resulting function, using $\lambda = 1, \alpha = 0.25$ found heuristically via cross-validation.

6.2. Results

We present the mean absolute error and the mean squared error (MSE) for conductivity reconstructions σ_j, $j \in [J_{test}]$ in Table 2. Since the conductivity values $\sigma(y)$ are in $[1, 5]$, we follow [8] and compute MSE where outputs are normalized to $[0, 1]$. We do not provide scaled MAE, as this would add little interpretive value while making the presentation unnecessarily busy.

Table 2. MAE and MSE for different local model sizes using linear and MLP output layers. Each RBON learner uses 21 branch RBFs and 21 trunk RBFs with regularization parameters $\alpha = 0.25$ and $\lambda = 1$. A visualization of this table can be found in Figure 4. Computation times represent fit times for RBON ensembles with linear output layers.

# Regions	# Learners (T)	Linear Output			MLP Output			Time (s)
		MAE	MSE	Scaled MSE	MAE	MSE	Scaled MSE	
1	1	0.2389	0.2957	0.0739	0.2003	0.1979	0.0495	365.79
	5	0.1557	0.1997	0.0499	0.1349	0.1266	0.0317	358.40
	10	0.1364	0.1836	0.0459	0.1212	0.1097	0.0274	498.55
	15	0.1236	0.1803	0.0451	0.1115	0.1161	0.0290	546.77
1–2	1	0.3649	0.5175	0.1294	0.3363	0.4268	0.1067	404.04
	5	0.2981	0.4329	0.1082	0.2601	0.3094	0.0774	421.77
	10	0.2803	0.4179	0.1045	0.2528	0.3080	0.0770	504.54
	15	0.2734	0.4214	0.1054	0.2581	0.3360	0.0840	572.20
2	1	0.4454	0.6574	0.1644	0.4183	0.5611	0.1403	466.67
	5	0.3859	0.5761	0.1440	0.3433	0.4412	0.1103	475.57
	10	0.3680	0.5604	0.1401	0.3312	0.4248	0.1062	588.07
	15	0.3574	0.5667	0.1417	0.3289	0.4352	0.1088	612.37

To contextualize RBON's performance, we compare its single-learner configuration (linear output) against two networks: DeepONet [15] as a benchmark and state-of-the-art PI-DION [19]. Both baselines were trained on high-end GPUs (NVIDIA Quadro GV100 GPU and GeForce RTX 2080 Ti) using CUDA-enabled PyTorch 2.8.0, with computation capped at approximately 100 hours and the best checkpoint selected for testing. The network layer widths for DeepONet and PI-DION ranged from 31 to 100 neurons in order to achieve results that were reasonably comparable. In contrast, the smaller RBON (single hidden layer with 21 neurons) network was trained on a local CPU (Intel Core i7-12700H, 12th generation; 32 GB RAM) using Julia. Table 3 summarizes the MSE and MAE across single region, mixed cases of 1–2 regions, and 2 region scenarios. RBON consistently achieves lower errors across all cases despite its significantly lighter computational setup.

Table 3. MSE and MAE comparison across three scenarios for RBON, PI-DION, and DeepONet. The best performance is bold.

Scenario	RBON (Linear, T=1)	PI-DION	DeepONet
1 region	**MSE 0.2957/MAE 0.2389**	MSE 0.5617/MAE 0.3121	MSE 0.6859/MAE 0.3832
1–2 regions	**MSE 0.5175/MAE 0.3649**	MSE 0.8585/MAE 0.4541	MSE 1.0343/MAE 0.5313
2 regions	**MSE 0.6574/MAE 0.4454**	MSE 1.1091/MAE 0.5991	MSE 1.3389/MAE 0.7238

We observe that the model performs best on one region, and increases its error linearly with the number of regions. To get a better look at how the errors compare, we also consider the graph representation of Table 2 portrayed in Figure 4. Two noteworthy factors of the model serve to improve the error: the number of local models T, and the type of post-processing layer we choose, linear or MLP; we address these separately.

We find that the error improves as we increase the number of local models T. However, we note that, going from $T = 10$ to $T = 15$, the MSE actually increases. This can be explained by the fact that we are not increasing J as we increase T: when $T = 1$, we have one model trained on $J = 4000$, but, when $T = 10$, we have an average of $J_\tau = 400$ (assuming equal distribution of samples). As a result, the improvements we see occur in spite of the deficit of training data. Based on initial exploratory testing, we would ideally have $500 \leq J_\tau \leq 1000$ for all $\tau \in [T]$. However, in practice, training data takes time and energy to obtain, so we limit how much we allow ourselves for our numerical tests.

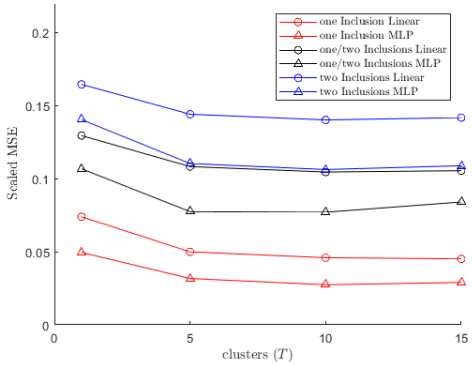


Figure 4. A visualization of the errors in Table 2.

As we expect, the inclusion of a GELU-based MLP refinement layer also improves the performance of the model with respect to the scaled MSE when compared to a simpler linear layer. This is especially manifest when we see that the MLP error for two regions is only slightly worse than the linear error for the mixture of one and two regions. Interestingly, though, this improvement seems to occur independently from the number of high-conductivity regions: there is roughly a flat 0.03 improvement across each region. The flatness of this improvement suggests that the nonlinearity is capturing a systematic residual pattern left by the RBON rather than interacting strongly with the number of regions. This is consistent with our design choice to fix the RBF locations and scales: the refinement layer provides expressivity without complicating optimization by making kernel parameters updateable.

A further comparison across models trained on the different numbers of regions while holding $T = 15$ highlights the effect of training distribution on reconstruction quality. The model trained solely on one high-conductivity region consistently achieves the lowest errors, as seen in Figures 5 and 6, where most predictions capture the sharp jump around the border of the region with good accuracy. The excellent performance in test observations is also due to sufficient size in the training data set, which captures a large enough variety

of sizes and locations for a single high-conductivity region, which has a more limited range of possibilities than multiple regions.

In contrast, we see the prediction accuracy begin to drop when we consider Figures 7 and 8. This exacerbates further with the two-region model (Figure 9), which demonstrates a higher error; while it roughly predicts the size and location correctly, the two-region model has more difficulty in maintaining a sharp transition around the borders of the high-conductivity regions. A larger training set would certainly reduce the error around the borders, but this is the nature of including more regions of sharp contrast in the model output.

When the training set includes a mixture of one and two regions in the output, the model has an overall lower error than the model trained to predict two regions as the smaller error on the single region predictions reduces the overall error. Interestingly, the error on the test observations that predict a single region of high conductivity are similar to the model trained to predict only one region; similarly, the error on the test observations with two regions of high conductivity has error close to the test error for the model trained solely on output with two high-conductivity regions. This results in a higher variance of errors for the mixed 1–2 region model (0.118), compared to 0.018 for the model trained to predict one region and 0.067 for two regions.

As for the out-of-distribution (OOD) results, we let $T = 15$ and consider the model that trained on one and two regions. We then apply it to several randomly generated squares of high conductivity; the results are found in Figure 10. We note that, surprisingly, the MSEs remain comparable to the results for in-distribution with one region. However, while the model finds the location of the high-conductivity regions, it struggles to attain the shape of the regions. This fits expectations given the Gaussian nature of the RBFs when compared to the non-Gaussian nature of the examples; locations are captured correctly, but shapes are smoothed.

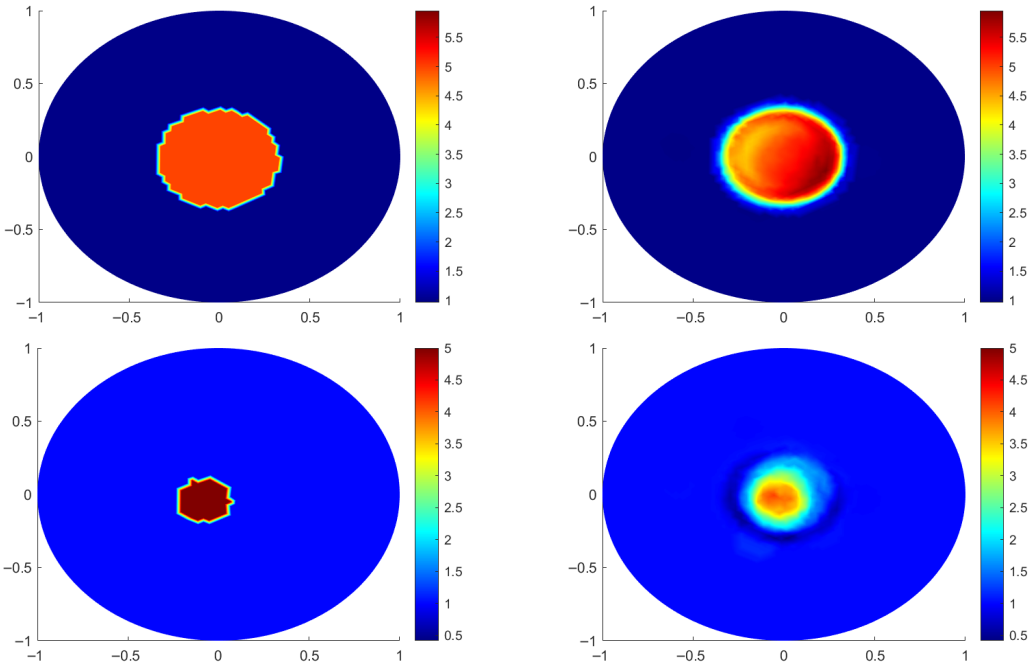


Figure 5. True (**left**) versus predicted (**right**) conductivity as produced by the model trained on a single high-conductivity region. Selected samples demonstrate a wide range of location and sizes.

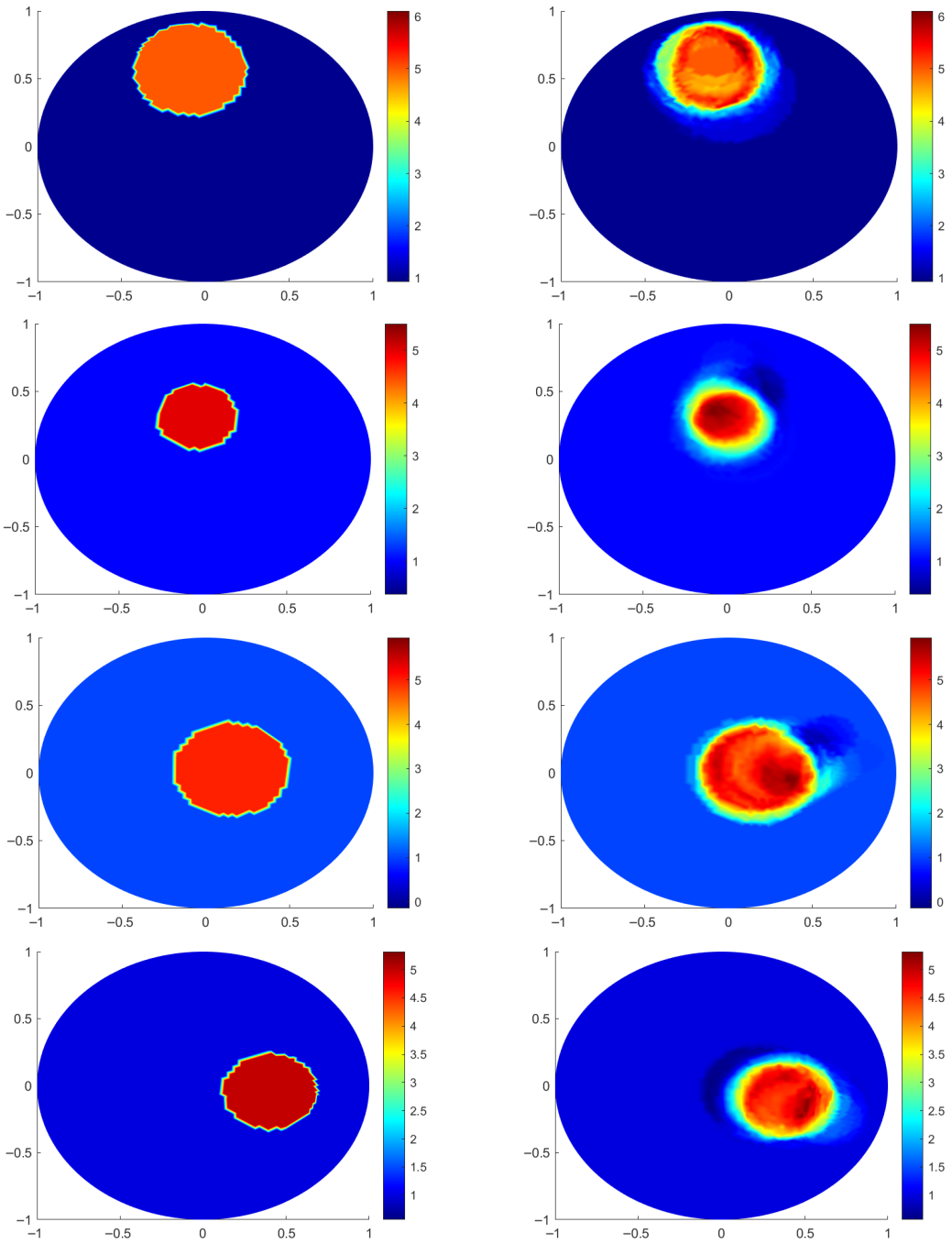


Figure 6. True (**left**) versus predicted (**right**) conductivity as produced by the model trained on a single high-conductivity region. Selected samples demonstrate variety across the predictions.

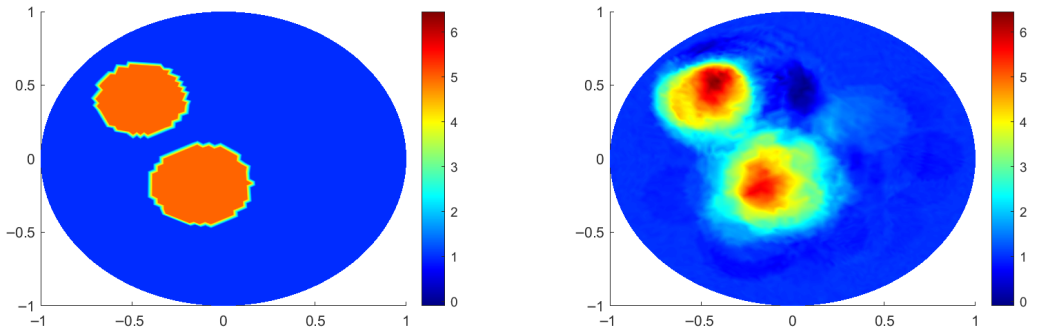


Figure 7. *Cont.*

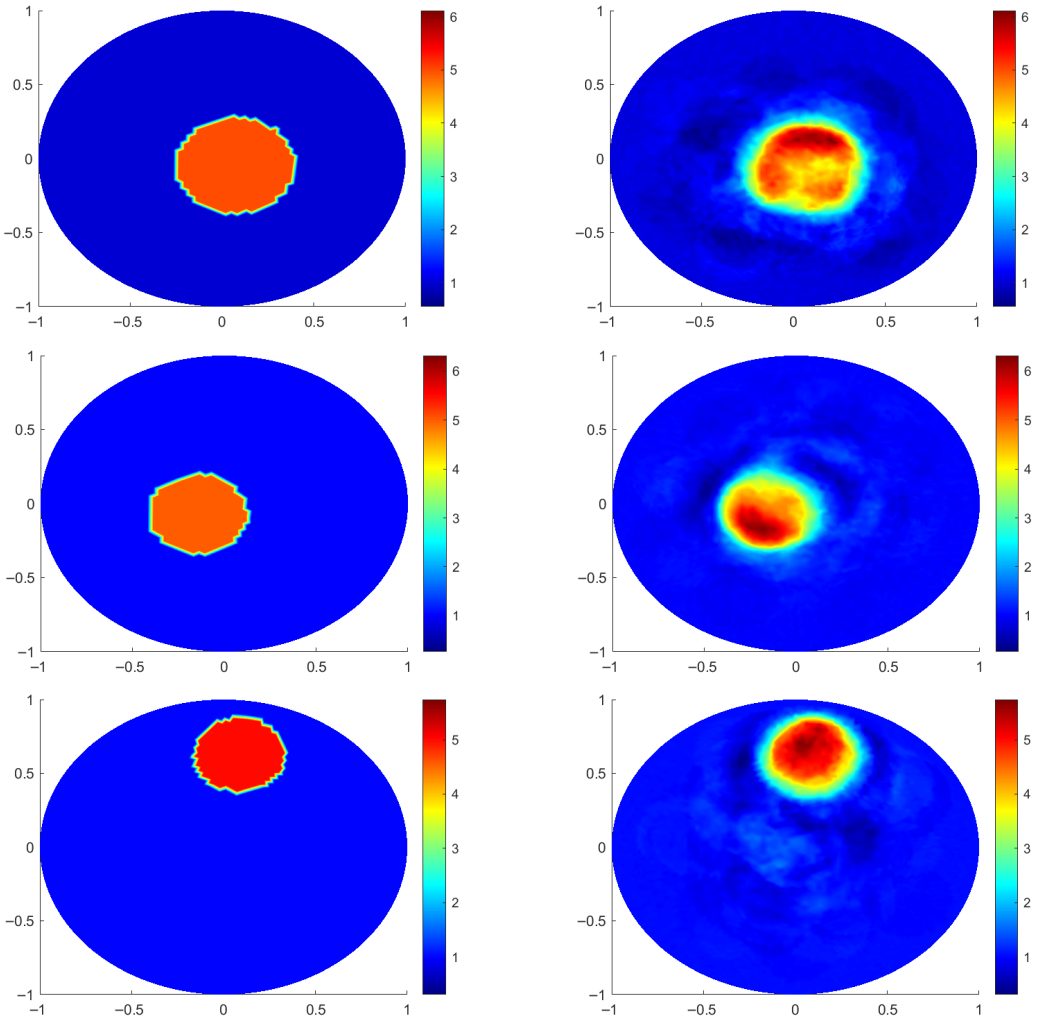


Figure 7. True (**left**) versus predicted (**right**) conductivity as produced by the model trained on a mixture of 1–2 high-conductivity regions. Selected samples demonstrate variety across the predictions.

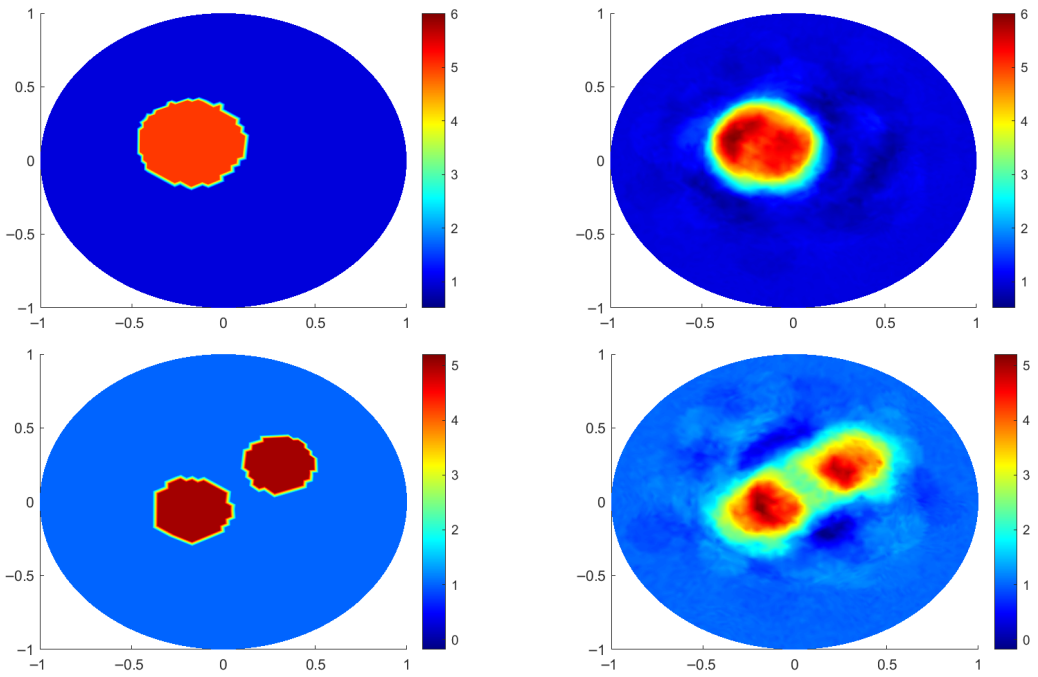


Figure 8. *Cont.*

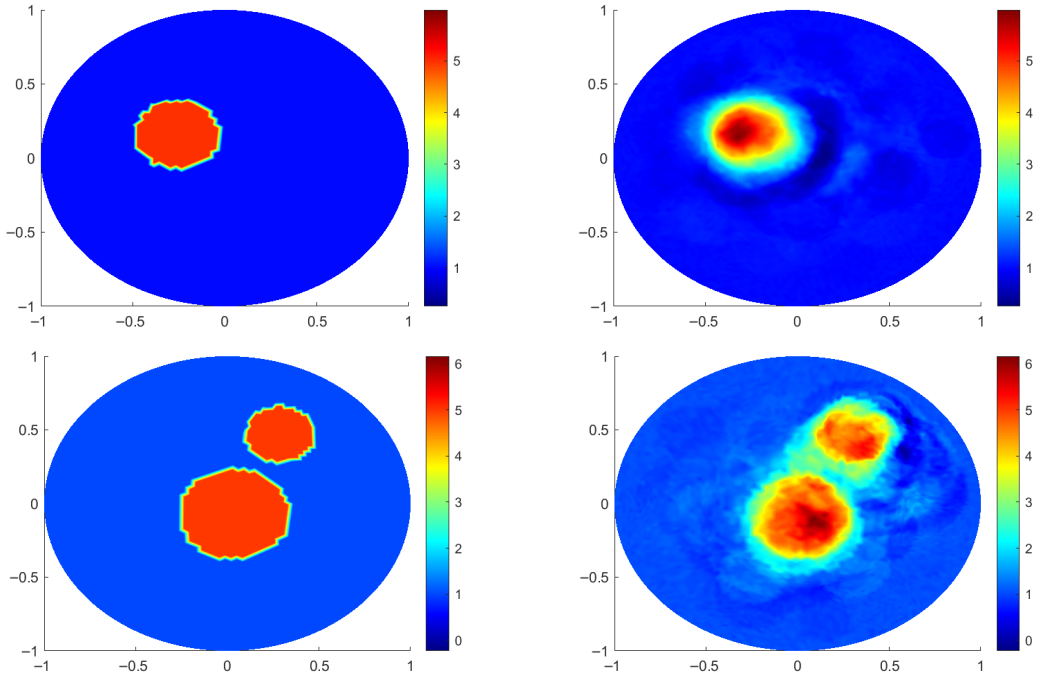


Figure 8. True (**left**) versus predicted (**right**) conductivity as produced by the model trained on a mixture of 1–2 high-conductivity regions. Selected samples demonstrate variety across the predictions.

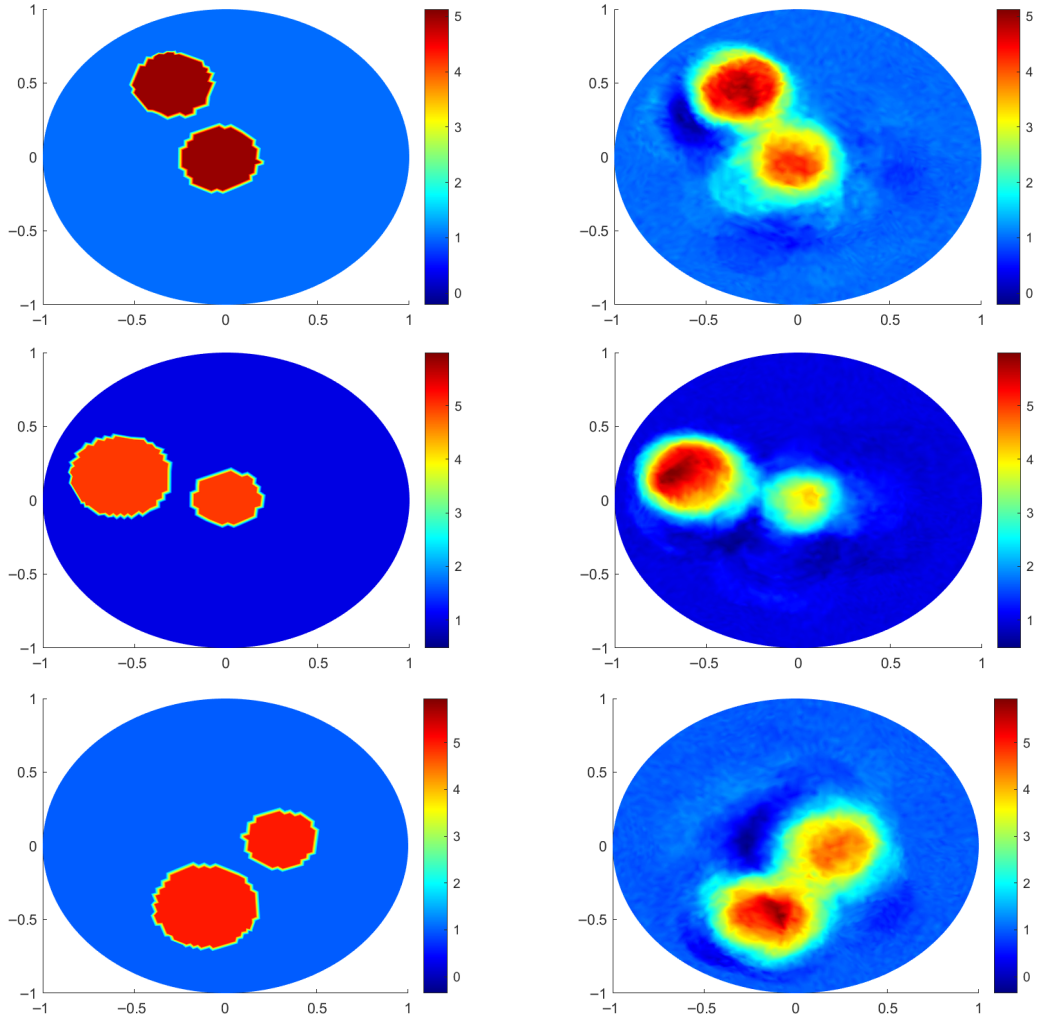


Figure 9. *Cont.*

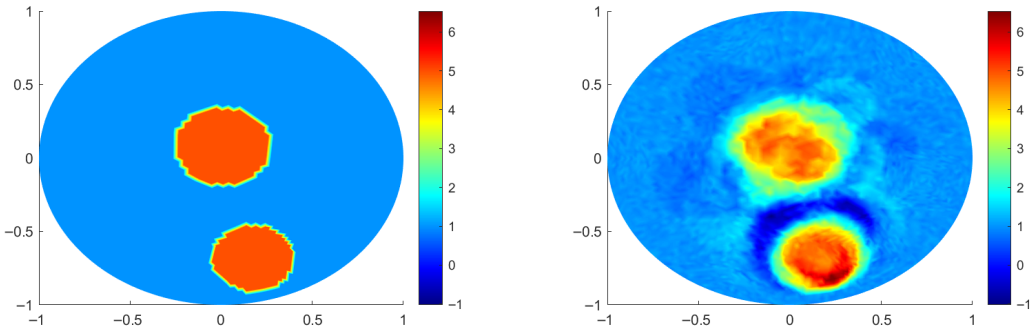
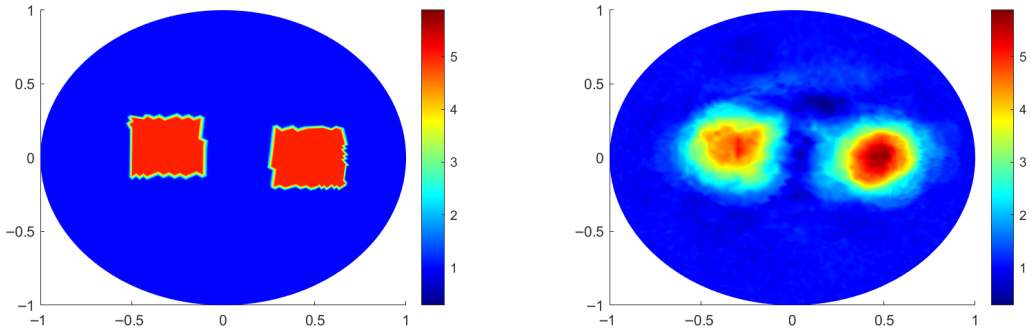
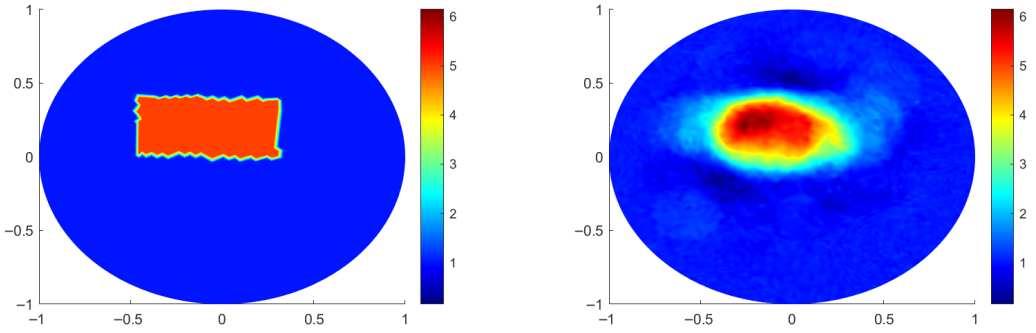


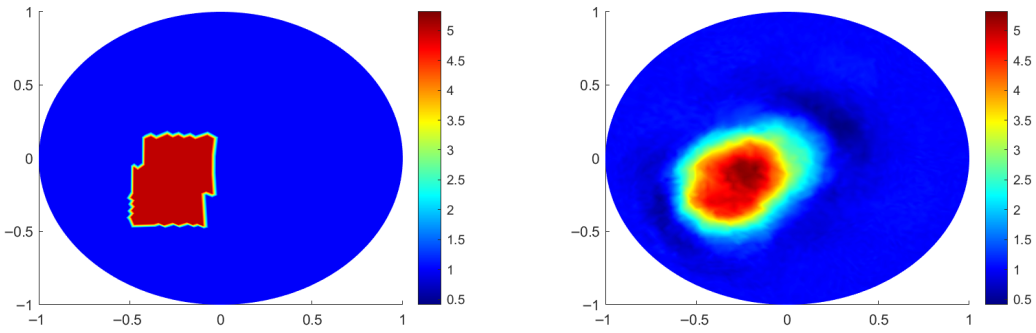
Figure 9. True (**left**) versus predicted (**right**) conductivity as produced by the model trained on 2 high conductivity regions. Selected samples demonstrate variety across the predictions.



(**a**)



(**b**)



(**c**)

Figure 10. Out-of-distribution test samples; true data is on the left, and the prediction is on the right. (**a**) MAE: 0.2761; MSE: 0.3111; Scaled MSE: 0.0778. (**b**) MAE: 0.2148; MSE: 0.2413; Scaled MSE: 0.0601. (**c**) MAE: 0.1872; MSE: 0.2194; Scaled MSE: 0.0549.

7. Conclusions

We have demonstrated a state-of-the-art operator network, the RBON network, on the Electrical Impedance Tomography (EIT) medical imaging problem. Given a domain and set of training functions defining electric potential and conductivity, we find that it performs very well in the inverse problem, that is, finding the conductivity of the domain based only on the potentials of the boundaries. This is in comparison to other later work [8], which has obtained similar results by assuming that the potential is known over the entire domain. Though the number of high-conductivity regions inside the domain decrease the MSE performance of the algorithm, we find that, image-wise, the model is still able to identify the location of the conductivity inside the domain. Frequently, this is all that is necessary in practical applications. Importantly, this model is able to perform for high-conductivity regions of shapes outside the distribution of its basis functions, which further increases its value. In addition, we compared the performance of the RBON to that of DeepONet [15] and a fellow cutting-edge network, PI-DION [19], where we learned that RBON performs more accurately and faster. As a result, this network, especially when overlaid with a simple MLP layer, has the potential to revolutionize imaging technology.

The RBON framework handles conductivity discontinuities by fitting to their continuous approximations. This is consistent with other forms of operator networks and more classical PDE solution methods. As we have demonstrated, the RBON excels where there is sufficient data to use to train, being faster and more accurate than other methods at a fraction of the computation power. In addition, we demonstrated the ability of the RBON to fit to conductivities lying outside its kernel distributions (see Figure 10). However, like other statistical models, if the data drifts sufficiently, then RBON will struggle: for instance, training only on two high-conductivity regions results in an inability to spot three.

With the cutting-edge nature of this research, there are several potential directions we can proceed. From an optimization standpoint, we have so far relied on relatively simple regularization schemes such as elastic-net. However, more advanced strategies such as efficient and adaptive global optimization that includes updating the kernel parameters would be worth investigating. This includes, but is not limited to, the consideration of more adaptive optimization strategies, second-order optimization, or even proximal methods. Such techniques may further stabilize training and allow for larger models without incurring variance penalties.

Beyond EIT, the operator learning paradigm embodied by the RBON is well positioned to address other classes of PDE-based inverse problems. For instance, fluid flow reconstruction, subsurface geophysics, or even nonlinear material modeling could benefit from a mesh-free, data-efficient operator network that generalizes across functions rather than individual inputs. Moreover, integrating RBONs with PINNs may provide a principled way to embed governing equations into the architecture, combining the expressivity of operator learning with the strong inductive bias of physical constraints. Taken together, these directions suggest that the RBON framework is not only effective for EIT but also represents a promising step toward more general-purpose operator learning in scientific and medical imaging.

Author Contributions: J.K.: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing—Review and Editing. A.P.: Methodology, Software, Validation, Formal analysis, Writing—Original Draft, Writing—Review and Editing. T.K.: Conceptualization, Methodology, Supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The raw data and source code supporting the conclusions of this article are available at: https://github.com/jkurz119/EnsembleRBON (accessed on 13 January 2026). The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jin, B.; Khan, T.; Maass, P. A reconstruction algorithm for electrical impedance tomography based on sparsity regularization. *Int. J. Numer. Methods Eng.* **2012**, *89*, 337–353. [CrossRef]
2. Jin, B.; Maass, P. Sparsity regularization for parameter identification problems. *Inverse Probl.* **2012**, *28*, 123001. [CrossRef]
3. Mueller, J.L.; Siltanen, S. The D-bar method for electrical impedance tomography—demystified. *Inverse Probl.* **2020**, *36*, 093001. [CrossRef]
4. Strauss, T.; Khan, T. Statistical inversion in electrical impedance tomography using mixed total variation and non-convex ℓ_p regularization prior. *J. Inverse Ill-Posed Probl.* **2015**, *23*, 529–542. [CrossRef]
5. Kirsch, A.; Grinberg, N. *The Factorization Method for Inverse Problems*; OUP: Oxford, UK, 2007; Volume 36.
6. Beals, R.; Coifman, R.R. Linear spectral problems, non-linear equations and the δ-method. *Inverse Probl.* **1989**, *5*, 87. [CrossRef]
7. Isaacson, D.; Mueller, J.L.; Newell, J.C.; Siltanen, S. Reconstructions of chest phantoms by the D-bar method for electrical impedance tomography. *IEEE Trans. Med. Imaging* **2004**, *23*, 821–828. [CrossRef]
8. Pokkunuru, A.; Rooshenas, P.; Strauss, T.; Abhishek, A.; Khan, T. Improved training of physics-informed neural networks using energy-based priors: A study on electrical impedance tomography. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
9. Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv* **2020**, arXiv:2003.03485. [CrossRef]
10. Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Stuart, A.; Bhattacharya, K.; Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6755–6766.
11. Lagaris, I.; Likas, A.; Fotiadis, D. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **1998**, *9*, 987–1000. [CrossRef]
12. Raissi, M.; Perdikaris, P.; Karniadakis, G. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv* **2017**, arXiv:1711.10561. [CrossRef]
13. Majee, S.; Abhishek, A.; Strauss, T.; Khan, T. MCMC-Net: Accelerating Markov Chain Monte Carlo with Neural Networks for Inverse Problems. *arXiv* **2024**, arXiv:2412.16883. [CrossRef]
14. Li, Z.; Kovachki, K.; Azizzadenesheli, B.; Liu, K.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Neural operator: Learning maps between function spaces. *arXiv* **2021**, arXiv:2010.08895.
15. Lu, L.; Jin, P.; Pang, G.; Zhang, Z.; Karniadakis, G.E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **2021**, *3*, 218–229. [CrossRef]
16. Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Fourier Neural Operator for Parametric Partial Differential Equations. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
17. Cao, Q.; Goswami, S.; Karniadakis, G.E. Laplace neural operator for solving differential equations. *Nat. Mach. Intell.* **2024**, *6*, 631–640. [CrossRef]
18. Li, Z.; Zheng, H.; Kovachki, N.; Jin, D.; Chen, H.; Liu, B.; Azizzadenesheli, K.; Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM/IMS J. Data Sci.* **2024**, *1*, 1–27. [CrossRef]
19. Cho, S.W.; Son, H. Physics-Informed Deep Inverse Operator Networks for Solving PDE Inverse Problems. In Proceedings of the Thirteenth International Conference on Learning Representations, Singapore, 24–28 April 2025.
20. Pathak, J.; Subramanian, S.; Harrington, P.; Raja, S.; Chattopadhyay, A.; Mardani, M.; Kurth, T.; Hall, D.; Li, Z.; Azizzadenesheli, K.; et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv* **2022**, arXiv:2202.11214.
21. Rosofsky, S.G.; Al Majed, H.; Huerta, E.A. Applications of physics informed neural operators. *Mach. Learn. Sci. Technol.* **2023**, *4*, 025022. [CrossRef]
22. Grady, T.J.; Khan, R.; Louboutin, M.; Yin, Z.; Witte, P.A.; Chandra, R.; Hewett, R.J.; Herrmann, F.J. Model-parallel Fourier neural operators as learned surrogates for large-scale parametric PDEs. *Comput. Geosci.* **2023**, *178*, 105402. [CrossRef]
23. Kurz, J.; Bowman, B.; Seman, M.; Oian, C.; Khan, T. A physics-informed kernel approach to learning the operator for parametric PDEs. *Neural Comput. Appl.* **2024**, *36*, 22773–22787. [CrossRef]
24. Palamodov, V.P. Gabor analysis of the continuum model for impedance tomography. *Ark. För Mat.* **2002**, *40*, 169–187. [CrossRef]

25. Banks, H.T.; Kunisch, K. *Estimation Techniques for Distributed Parameter Systems*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
26. Banks, H.T. *A Functional Analysis Framework for Modeling, Estimation and Control in Science and Engineering*; CRC Press: Boca Raton, FL, USA, 2012.
27. Borcea, L. Electrical impedance tomography. *Inverse Probl.* **2002**, *18*, R99. [CrossRef]
28. Jin, B.; Maass, P. An analysis of electrical impedance tomography with applications to Tikhonov regularization. *ESAIM Control. Optim. Calc. Var.* **2012**, *18*, 1027–1048. [CrossRef]
29. Kaipio, J.P.; Kolehmainen, V.; Somersalo, E.; Vauhkonen, M. Statistical inversion and Monte Carlo sampling methods in electrical impedance tomography. *Inverse Probl.* **2000**, *16*, 1487. [CrossRef]
30. Nissinen, A.; Heikkinen, L.; Kaipio, J. Approximation errors in electrical impedance tomography—An experimental study. In Proceedings of the 5th World Congress in Industrial Process Tomography, Bergen, Norway, 3–6 September 2007; pp. 858–864.
31. Nissinen, A.; Kolehmainen, V.; Kaipio, J. Compensation of errors due to incorrect model geometry in electrical impedance tomography. *J. Phys. Conf. Ser.* **2010**, *224*, 012050. [CrossRef]
32. Smith, R.C. *Uncertainty Quantification: Theory, Implementation, and Applications*; SIAM: Bangkok, Thailand, 2013; Volume 12.
33. Kurz, J.; Oughton, S.; Liu, S. Radial Basis Operator Networks. *arXiv* **2024**, arXiv:2410.04639. [CrossRef]
34. Kupis, S. Methods for the Electrical Impedance Tomography Inverse Problem: Deep Learning and Regularization with Wavelets. Master's thesis, Clemson University, Clemson, SC, USA, 2021.
35. Lax, P.D.; Milgram, A.N. Contributions to the Theory of Partial Differential Equations. *Parabol. Equ.* **1954**, *9*, 167.
36. Daubechies, I. *Ten Lectures on Wavelets*; SIAM: Bangkok, Thailand, 1992.
37. Schechter, M. *Principles of Functional Analysis*; American Mathematical Soc.: Providence, RI, USA, 2001.
38. Ascher, U.M.; Greif, C. *A First Course on Numerical Methods*; SIAM: Bangkok, Thailand, 2011.
39. Brenner, S.C.; Scott, L.R. *The Mathematical Theory of Finite Element Methods*; Springer: Berlin/Heidelberg, Germany, 2008.
40. Chen, T.; Chen, H. Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks. *IEEE Trans. Neural Netw.* **1995**, *6*, 904–910. [CrossRef]
41. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [CrossRef]
42. Forgy, E.W. Cluster analysis of multivariate data : Efficiency versus interpretability of classifications. *Biometrics* **1965**, *21*, 768–769.
43. Chen, T.; Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Netw.* **1995**, *6*, 911–917. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.